Gnus Manual

by Lars Magne Ingebrigtsen

Copyright © 1995--2026 Free Software Foundation, Inc.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, with the Front-Cover Texts being ‘‘A GNU Manual”, and with the Back-Cover Texts as in (a) below. A copy of the license is included in the section entitled ‘‘GNU Free Documentation License”.

この文書を、フリーソフトウェア財団発行の GNU フリー文書利用許諾契約書第 1.3 版またはそれ以降の版が定める条件の下で複製、配布、あるいは変更することを許可します。変更不可部分は指定しません。“A GNU Manual”は表表紙テキスト、以下の(a) は裏表紙テキストです。この利用許諾契約書の複写は「GNU フリー文書利用許諾契約書」という章に含まれています。

(a) The FSF’s Back-Cover Text is: ‘‘You have the freedom to copy and modify this GNU manual.”

(a) FSF の裏表紙テキスト:「あなたにはこの GNU Manual を複製したり変更する自由があります。」

簡単な目次

1 あわてないで 1
2 Gnus の起動 3
3 グループバッファー 12
4 概略バッファー 47
5 記事バッファー 125
6 メッセージの作成 135
7 選択方法 146
8 スコア 233
9 検索 257
10 いろいろ 271
11 終わり 327
12 付録 328
13 GNU フリー文書利用許諾契約書 413
14 Index 421
15 Key Index 442

目次

1 あわてないで 1

2 Gnus の起動 3

2.1 ニュースを見つける 3
2.2 サーバーが落ちている 4
2.3 子どもの Gnus 4
2.4 新しいグループ 5
2.4.1 新しいグループを調べる 5
2.4.2 購読方法 6
2.4.3 新しいグループを選別する 7
2.5 サーバーを換える 7
2.6 起動ファイル 8
2.7 自動保存 9
2.8 アクティブファイル 10
2.9 起動変数 11

3 グループバッファー 12

3.1 グループバッファーの形式 12
3.1.1 グループ行の仕様 12
3.1.2 グループモード行の仕様 14
3.1.3 グループのハイライト 14
3.2 グループ操作 15
3.3 グループの選択 16
3.4 購読制御コマンド 18
3.5 グループデータ 18
3.6 グループレベル 19
3.7 グループのスコア 20
3.8 グループへの印 21
3.9 外部グループ 21
3.10 グループパラメーター 23
3.11 グループの一覧表示 31
3.12 グループの並べ替え 32
3.13 グループの管理 34
3.14 外部サーバーの閲覧 34
3.15 Gnus の終了 35
3.16 トピック 35
3.16.1 トピック命令 35
3.16.2 トピック変数 38
3.16.3 トピックの並べ替え 39
3.16.4 トピックの位相構造 39
3.16.5 トピックパラメーター 40
3.17 英字以外の名前のグループへのアクセス 41

3.18 その他のグループ関連 ... 43
3.18.1 新着メッセージを探す ... 44
3.18.2 グループ情報 ... 44
3.18.3 グループの日付 ... 45
3.18.4 ファイル命令 ... 45
3.18.5 Sieve コマンド ... 45

4 概略バッファー ... 47
4.1 概略バッファーの様式 ... 47
4.1.1 概略バッファーの行 ... 47
4.1.2 To From Newsgroups ... 50
4.1.3 概略バッファーのモード行 ... 52
4.1.4 概略のハイライト ... 52
4.2 概略間の移動 ... 53
4.3 記事の選択 ... 54
4.3.1 選択命令 ... 54
4.3.2 選ぶための変数 ... 55
4.4 記事のスクロール ... 56
4.5 返答、フォローアップ、投稿 ... 56
4.5.1 概略でのメールの命令 ... 57
4.5.2 概略の投稿命令 ... 59
4.5.3 概略メッセージ命令 ... 60
4.5.4 記事を取り消す ... 60
4.6 遅延記事 ... 61
4.7 記事に印を付ける ... 62
4.7.1 未読記事 ... 62
4.7.2 既読記事 ... 63
4.7.3 他の印 ... 63
4.7.4 印を付ける ... 64
4.7.5 一般的な印を付けるコマンド ... 65
4.7.6 プロセス印を付ける ... 66
4.8 制限をする ... 67
4.9 スレッド ... 69
4.9.1 スレッドをカスタマイズする ... 70
4.9.1.1 無束縛スレッド ... 70
4.9.1.2 スレッドを埋める ... 72
4.9.1.3 もっとスレッドを ... 73
4.9.1.4 低レベルにおけるスレッド作成 ... 74
4.9.2 スレッドの命令 ... 75
4.10 並べ替え ... 76
4.11 非同期記事取得 ... 77
4.12 記事のキャッシュ ... 79
4.13 永続記事 ... 80
4.14 粘着記事 ... 80
4.15 記事のバックログ ... 81
4.16 記事の保存 ... 81

4.17 記事のデコード 86
4.17.1 uuencode された記事 86
4.17.2 シェルアーカイブ 87
4.17.3 ポストスクリプトファイル 87
4.17.4 他のファイル 87
4.17.5 デコードのための変数 87
4.17.5.1 規則変数 87
4.17.5.2 他のデコードのための変数 88
4.17.5.3 uuencode と投稿 89
4.17.6 ファイルの表示 89
4.18 記事のトリートメント 90
4.18.1 記事のハイライト 90
4.18.2 記事中の文の強調表示 91
4.18.3 記事を隠す 92
4.18.4 記事の洗濯 94
4.18.5 記事ヘッダー 98
4.18.6 記事のボタン 98
4.18.6.1 関連する変数と関数 99
4.18.7 記事ボタンのレベル 100
4.18.8 記事の日付 100
4.18.9 記事に表示するもの 101
4.18.10 記事の署名 102
4.18.11 記事いろいろ 102
4.19 MIME コマンド 103
4.20 文字セット 106
4.21 記事命令 107
4.22 概略の並べ替え 108
4.23 親記事を探す 109
4.24 代替手段 110
4.24.1 選んで読む 111
4.24.2 バイナリーグループ 112
4.25 木表示 112
4.26 メールグループ命令 114
4.27 概略のいろいろなもの 116
4.27.1 概略グループ情報 118
4.27.2 記事を探す 118
4.27.3 概略生成命令 118
4.27.4 本当にいろいろな概略命令 118
4.28 概略バッファーを抜ける 119
4.29 クロスポストの扱い 121
4.30 重複の抑制 121
4.31 セキュリティー 122
4.32 メーリングリスト 124

5 記事バッファー 125
5.1 余分なヘッダーを隠す 125

5.2 MIME を使う 126
5.3 HTML 128
5.4 記事のカスタマイズ 129
5.5 記事のキーマップ 132
5.6 記事のその他 133

6 メッセージの作成 135

6.1 メール 135
6.2 投稿するサーバー 135
6.3 POP before SMTP 136
6.4 メールと投稿 137
6.5 メッセージの保管 137
6.6 投稿様式 140
6.7 下書き 143
6.8 拒否された記事 144
6.9 署名と暗号化 144

7 選択方法 146

7.1 サーバーバッファー 146
7.1.1 サーバーバッファーの表示様式 146
7.1.2 サーバー命令 147
7.1.3 方法の例 148
7.1.4 仮想サーバーを作成する 149
7.1.5 サーバー変数 149
7.1.6 サーバーと選択方法 150
7.1.7 使用不可能なサーバー 150
7.2 ニュースの取得 151
7.2.1 NNTP 151
7.2.1.1 直接接続するための関数 154
7.2.1.2 間接的に接続するための関数 156
7.2.1.3 共通の変数 157
7.2.2 ニューススプール 158
7.3 IMAP を使う 159
7.3.1 IMAP サーバーに接続する 159
7.3.2 IMAP 接続をカスタマイズする 159
7.3.3 クライアント側での IMAP 分割 161
7.3.4 IMAP 拡張のサポート 162
7.4 メール取得 162
7.4.1 ニュースリーダーでメール 162
7.4.2 メールを読むことを始める 163
7.4.3 メールの分割 164
7.4.4 メールソース 166
7.4.4.1 メールソース指示子 166
7.4.4.2 関数インターフェース 172
7.4.4.3 メールソースのカスタマイズ 172
7.4.4.4 メールの取得 173

7.4.5 メールバックエンド変数 174
7.4.6 特級メール分割 175
7.4.7 グループメール分割 178
7.4.8 古いメールを取り込む 180
7.4.9 メールの期限切れ消去 181
7.4.10 メール洗濯 184
7.4.11 重複 186
7.4.12 メールを読むのではない 187
7.4.13 メールバックエンドを選ぶ 187
7.4.13.1 Unix メールボックス 187
7.4.13.2 Babyl 187
7.4.13.3 メールスプール 188
7.4.13.4 MH スプール 189
7.4.13.5 Maildir 190
7.4.13.6 グループパラメーター 191
7.4.13.7 記事の識別 193
7.4.13.8 NOV データ 193
7.4.13.9 記事の印 194
7.4.13.10 メールフォルダー 194
7.4.13.11 メールバックエンドの比較 195
7.5 ウェブの閲覧 198
7.5.1 ウェブ検索 198
7.5.2 RSS 199
7.5.3 Atom 202
7.6 その他のグループ源 202
7.6.1 ディレクトリーグループ 202
7.6.2 なんでもグループ 203
7.6.3 文書グループ 204
7.6.3.1 文書サーバーの内部 205
7.6.4 メールからニュースへのゲートウェイ 207
7.6.5 空っぽのバックエンド 208
7.7 仮想グループ 209
7.7.1 選択グループ 209
7.7.2 合併グループ 210
7.8 電子メールによる日程管理 212
7.8.1 NNDiary バックエンド 212
7.8.1.1 日程メッセージ 212
7.8.1.2 NNDiary を動かす 213
7.8.1.3 NNDiary のカスタマイズ 214
7.8.2 Gnus Diary ライブラリー 214
7.8.2.1 日程の概略行仕様 215
7.8.2.2 日程記事の並べ替え 215
7.8.2.3 日程ヘッダーの生成 215
7.8.2.4 日程グループのパラメーター 216
7.9 Gnus の切り離し 216
7.9.1 エージェントの基礎 216
7.9.2 エージェント分類 218

7.9.2.1 分類の文法 218
7.9.2.2 分類バッファー 222
7.9.2.3 分類変数 223
7.9.3 エージェント命令 224
7.9.3.1 グループエージェント命令 224
7.9.3.2 概略エージェント命令 224
7.9.3.3 サーバーエージェント命令 225
7.9.4 エージェントの視覚効果 225
7.9.5 キャッシュとしてのエージェント 226
7.9.6 エージェント期限切れ消去 226
7.9.7 エージェントを作り直す 227
7.9.8 エージェントとフラグ 227
7.9.9 エージェントを IMAP で使う方法 228
7.9.10 差出用メッセージ 228
7.9.11 エージェント変数 228
7.9.12 設定例 231
7.9.13 一括エージェント処理 232
7.9.14 エージェントの問題点 232

8 スコア 233
8.1 概略スコア命令 233
8.2 グループスコア命令 236
8.3 スコア変数 236
8.4 スコアファイル様式 239
8.5 スコアファイルの編集 244
8.6 適応スコア付け 244
8.7 ホームスコアファイル 247
8.8 自分自身へのフォローアップ 248
8.9 他のヘッダーにスコアを付ける 248
8.10 スコア付けの奥義 249
8.11 逆スコア 250
8.12 グローバルスコアファイル 250
8.13 消去ファイル 251
8.14 消去ファイルの変換 252
8.15 上級スコア付け 253
8.15.1 上級スコア付け構文 253
8.15.2 上級スコア付けの例 253
8.15.3 上級スコアのちょっとした秘訣 254
8.16 スコアを減衰させる 255

9 検索 257
9.1 検索エンジン 257
9.2 検索グループを作る 259
9.3 検索構文 260
9.3.1 日付の値の解析 261
9.4 nnmairix 261

9.4.1 メール検索エンジン mairix について 261
9.4.2 nnmairix を使うために必要なこと 262
9.4.3 nnmairix は実際に何をするのか? 262
9.4.4 mairix の設定 263
9.4.5 nnmairix バックエンドの設定 264
9.4.6 nnmairix で利用できるショートカットキー 265
9.4.7 nnmairix グループの印を伝搬させる方法 266
9.4.8 nnmairix のヒント、こつ、およびいくつかの例 268
9.4.9 nnmairix でさらに知っておく必要があること 269
9.5 nnir からの移行 270

10 いろいろ 271

10.1 プロセス/接頭引数 271
10.2 利用者との相互作用 271
10.3 シンボルの接頭引数 272
10.4 書法仕様変数 272
10.4.1 書法仕様の基本 273
10.4.2 モード行書法仕様 273
10.4.3 上級書法仕様 273
10.4.4 利用者定義の指定 274
10.4.5 書法仕様フォント 275
10.4.6 ポイントの移動 276
10.4.7 整列 276
10.5 ウィンドウの配置 276
10.5.1 ウィンドウ配置の名称 280
10.5.2 ウィンドウ配置の例 281
10.6 Tab によるインターフェース 281
10.7 フェースとフォント 282
10.8 モード行 282
10.9 ハイライトとメニュー 282
10.10 デーモン 284
10.11 通知 285
10.12 やり直し 285
10.13 述語指示子 286
10.14 司会役 286
10.15 グループを取得する 286
10.16 画像の拡張 287
10.16.1 X-Face 287
10.16.2 Face 288
10.16.3 スマイリー 289
10.16.4 Picons 289
10.16.5 Gravatars 290
10.16.5.1 ツール・バー 291
10.17 ファジーな一致 291
10.18 spam メールの裏をかく 291
10.18.1 Spam の問題 292

10.18.2 Spam 退治の基礎 293
10.18.3 SpamAssassin, Vipul's Razor, DCC, etc 294
10.19 Spam パッケージ 295
10.19.1 Spam パッケージ序説 295
10.19.2 やって来るメールの濾過 297
10.19.3 グループにおける spam の検出 298
10.19.4 Spam と Ham プロセッサー 299
10.19.5 Spam パッケージの設定例 301
10.19.6 Spam バックエンド 304
10.19.6.1 ブラックリストとホワイトリスト 304
10.19.6.2 BBDB ホワイトリスト 306
10.19.6.3 Gmane Spam 報告 306
10.19.6.4 ブラックホール 307
10.19.6.5 正規表現によるヘッダーの合致検査 307
10.19.6.6 Bogofilter 308
10.19.6.7 SpamAssassin back end 309
10.19.6.8 ifile による spam の濾過 310
10.19.6.9 Spam 統計濾過 310
10.19.6.10 Gnus で SpamOracle を使うには 311
10.19.7 Spam パッケージの拡張 312
10.19.8 Spam 統計パッケージ 314
10.19.8.1 spam-統計(spam-stat) 辞書を作る 315
10.19.8.2 spam-統計(spam-stat) を使ってメールを分割する .. 316
10.19.8.3 spam-統計(spam-stat) 辞書への低階層インターフェース 317
10.20 Gnus レジストリー 318
10.20.1 Gnus レジストリーの設定 319
10.20.2 `Message-ID` に基づいてレジストリーで記事を取得する . 320
10.20.3 親への特級分割 321
10.20.4 独自のフラグとキーワードの記録 322
10.20.5 任意のデータの記録 322
10.21 Gnus クラウド 323
10.21.1 Gnus クラウドの設定 323
10.21.2 Gnus クラウドの使い方 323
10.22 システムスリープの統合 324
10.23 他のモードとの相互作用 324
10.23.1 Dired 324
10.24 いろいろのいろいろ 325

11 終わり 327

12 付録 328
12.1 歴史 328
12.1.1 Gnus Versions 328
12.1.2 なぜ? 329
12.1.3 標準への準拠 329

12.1.4 貢献者 330
12.1.5 新しい機能 332
12.1.5.1 (ding) Gnus 332
12.1.5.2 September Gnus 334
12.1.5.3 Red Gnus 336
12.1.5.4 Quassia Gnus 338
12.1.5.5 Pterodactyl Gnus 339
12.1.5.6 Oort Gnus 340
12.1.5.7 No Gnus 348
12.1.5.8 Ma Gnus 353
12.2 マニュアル 354
12.3 マニュアルを書く 354
12.4 用語 355
12.5 カスタマイズ 359
12.5.1 遅くて高価な接続 359
12.5.2 遅いターミナル接続 359
12.5.3 少ないディスク容量 360
12.5.4 遅いマシン 360
12.6 問題解決 361
12.7 Gnus リファレンスガイド 363
12.7.1 Gnus の有用な関数 363
12.7.2 バックエンドインターフェース 364
12.7.2.1 必須バックエンド関数 366
12.7.2.2 任意バックエンド関数 369
12.7.2.3 エラーメッセージの発行 372
12.7.2.4 新しいバックエンドを書く 373
12.7.2.5 新しいバックエンドを Gnus に繋げる 376
12.7.2.6 メール風バックエンド 376
12.7.2.7 Web フィードのバックエンド 377
12.7.3 スコアファイルの構文 377
12.7.4 ヘッダー 379
12.7.5 範囲 379
12.7.6 グループ情報 380
12.7.7 対話形式の拡張 382
12.7.8 いろいろなファイル様式 382
12.7.8.1 アクテイブファイルの様式 382
12.7.8.2 ニュースグループファイルの様式 383
12.8 異教徒への Emacs 384
12.8.1 打鍵 384
12.8.2 Emacs Lisp 384
12.9 よく尋ねられる質問 386
12.9.1 序 386
12.9.2 インストールに関する FAQ 386
12.9.3 起動/ グループバッファー 386
12.9.4 メッセージの取得 388
12.9.5 メッセージを読む 392
12.9.6 メッセージの作成 399

12.9.7 古いメッセージ 405
12.9.8 ダイアルアップ環境で Gnus を使う 407
12.9.9 助けを得る 409
12.9.10 Gnus をチューンする 410
12.9.11 用語集 411

13 GNU フリー文書利用許諾契約書 413

14 Index 421

15 Key Index 442

1 あわてないで

ようこそ Gnus ニュースリーダーと電子メールクライアントへ。Gnus はほとんどのクライアントとは異なります。ある面ではその無限の構成可能性のため、別の観点ではその歴史的な起源のためです。現在 Gnus はフル機能の電子メールクライアントですが、生まれは Usenet スタイルのニュースリーダーであり、その遺伝子はいまだニュースリーダーの遺伝子です。したがって、ほとんどのメールクライアントとは少し異なる振る舞いをします。

ニュースリーダーというものは一般に次のように考えられています:

1. サーバーは、さまざまな主題に関して潜在的に膨大な数のニュースグループを提供します。ユーザーはそれらのグループのほんの一部にだけ興味があるかもしれないし、いくつかのグループに他より関心を持っている場合もあるでしょう。
2. 多くのグループには大量の記事があって、ユーザーはそれらすべてを読みたいわけではありません。興味がある記事を前に出して、興味のないものは表に出さないためのメカニズムが必要です。
3. グループがスキャンされてユーザーが操作した後は、新しい記事が来るまではそのグループに関心を持つことはまずありません。

これらの考えは Gnus の特定のディフォルトの動作を導きます:

1. 関心があるすべてのグループが同じくらいに興味深いわけではないので、グループにはさまざまな程度の「購読度」があり、それに応じて変化する振る舞いがあります。
2. 記事を採点する、並べ替える、あるいは見なかったことにするためのコマンドやツールはたくさんあります。
3. Gnus は未読の記事や印が付けられた記事があるグループだけを表示します。新しい記事がないグループは表示しません。
4. グループに入ると未読または印が付けられた記事のみが表示され、他のすべての記事は表示されません。

もしこれが厳格すぎると思うなら、自動の Inbox Zero と考えてください。これが Gnus のディフォルトのやり方です。メールクライアントのように機能させる(常に既読グループと既読記事を表示させる) こともできますが、ユーザー側で多少の努力が必要です。

あなたを離陸させるには、以下の簡単な導入で十分なはずです。

サーバー、グループ、記事の基本

Gnus の基本的な構成要素は*servers* と*groups* と*articles* です。サーバーはローカルでもリモートでも構いません。それぞれのサーバーはグループのリストを保持していて、それらのグループには記事があります。Gnus は多種多様なサーバーへの統一されたインターフェースを提供するため、語彙は必ずしも整頓されていません(より完全な用語集についてはsee Section 12.9.11 [FAQ - Glossary], p. 411)。したがってローカルの Maildir は Usenet や IMAP サーバーと同じ「サーバー」(see Section 2.1 [Finding the News], p. 3) と呼ばれます。「グループ」(see Chapter 3 [Group Buffer], p. 12) は NNTP グループ、IMAP フォルダー、またはローカルメールディレクトリーを意味する場合があります。また「記事」(see Chapter 4 [Summary Buffer], p. 47) はメッセージまたは電子メールと呼ばれることもあります。Gnus はこれらすべてに統一された用語を採用しています。

サーバーは 2 つの一般的なカテゴリに分類されます。「ニュースふう」: 記事がパブリックアーカイブの一部であり、ユーザーが操作できないことを意味します。「メールふう」: 記事はユーザーが所有していて、自由に編集、移動、削除できることを意味します。

通常数百または数千のグループを提供する「ニュースふう」サーバーの場合は、それらのグループの一部だけを購読できることが重要です。「メールふう」サーバーの場合、ユーザーは通常すべてのグループが自動的に購読可能になります(ただし、例えば IMAP は選択的な購読もできます)。グループの購読を変更するには、サーバーバッファーに(^ で) 入って、目的のサーバーで`RET` を押します。その場所で Gnus はグループの購読を変更または切り替えるコマンドを提供します(see Section 3.14 [Browse Foreign Server], p. 34)。

基本的に Gnus の実装は、1 つ以上のサーバーのリスト、それらのサーバーからユーザーが購読するグループ、およびそれらのグループの記事です。

サーバーは 2 つの場所で追加と構成ができます: gnus.el 起動ファイルの中で`gnus-select-method` と`gnus-secondary-select-methods` オプションを使うことによって、または Gnus 自身のサーバーバッファーで対話的なコマンドを使うことによって。詳細はSee Section 2.1 [Finding the News], p. 3.

メールの取得

新しいメールはどこからかやって来なければなりません。NNTP や IMAP のようないくつかのサーバーは、それ自体が新着記事の取得を担当します。Maildir や mbox サーバーなどの他のものは記事を保存するだけで、どこからも取得しません。

後者の場合 Gnus は`mail sources`、すなわち新しいメールを取得する場所を提供します。メールソースは、ローカルスプール、リモート POP サーバー、またはその他の新着記事のソースです。メールソースは通常グローバルに設定しますが、グループごとに指定することもできます(より詳しくはsee Section 7.4.4 [Mail Sources], p. 166)。

新しいメールの取得に関する詳細はSee Section 3.18.1 [Scanning New Messages], p. 44.

メールの閲覧

Gnus のグループバッファーは、ディフォルトでは未読記事のあるグループだけを表示します。一時的にすべてのグループを表示することが`L` でできますし、いくつかのグループは常に表示するように Gnus を構成することも可能です(see Section 3.11 [Listing Groups], p. 31)。

グループへの入り方はSee Section 3.3 [Selecting a Group], p. 16. そこに行ったら何をすべきかについてはSee Chapter 4 [Summary Buffer], p. 47.

メールの送信

新しいメッセージの作成をグループバッファーから始めることができます(see Section 3.18 [Misc Group Stuff], p. 43)。概略バッファーにいる場合は、新しいメッセージを始めるだけでなく、返信や電子メールを転送することができます。詳しくはSee Section 4.5.1 [Summary Mail Commands], p. 57.

メッセージの作成を始めるとどうなるかについてはSee Chapter 6 [Composing Messages], p. 135, を参照してください。特にSMTP サーバーの設定についてはSee Section “メール変数” in *Message manual*.

2 Gnus の起動

Gnus を使う以前にあまり Emacs を使っていないのならば、最初にSection 12.8 [Emacs for Heathens], p. 384, を読んでください。

システム管理者が適切な設定をしていたならば、Gnus を起動してニュースを読むのは非常に簡単です。そう、Emacs で*M-x gnus* と打つだけです。さもなければ、変数`gnus-select-method` をカスタマイズしなければなりません。これはSection 2.1 [Finding the News], p. 3, で説明されています。また、投稿するための最低限の設定を行なうために、変数`user-full-name` および`user-mail-address` もカスタマイズしなければなりません。

別のフレーム(frame) で Gnus を起動したいときは、*M-x gnus-other-frame* 命令を使うことができます。

開始時に何かがうまくいかないときは`~/.gnus.el` ファイルの中で変数をいくつかいじくりまわさなければならないでしょう。このファイルは`~/.emacs` と似ていますが、こちらは Gnus が起動するときに読み込まれます。

この説明書でよくわからない用語がでてきたときは、用語の章(Section 12.4 [Terminology], p. 355) を参照してください。

2.1 ニュースを見つける

第一に、Gnus が認知しているすべてのサーバーを列挙している`*Server*` という特別なバッファーがあることを知っていなければなりません。それを見るにはグループバッファーで`^` を押してください。サーバーバッファーでは、定義されているあるサーバーで`RET` を押すことによって、それが取り扱っているすべてのグループを(購読しているかどうかにかかわらず!) 見ることができます。そこでサーバーを追加または削除したり、外部(foreign) サーバーの定義を編集したり、サーバーをエージェント化したり解除したり、さらに他の多くのことを手際良く行なうことができます。See Section 7.1 [Server Buffer], p. 146. See Section 3.9 [Foreign Groups], p. 21. See Section 7.9.1 [Agent Basics], p. 216.

変数`gnus-select-method` は Gnus がどこでニュースを探すべきかを示します。この変数ははじめの要素が「方法」、二番目の要素が「場所」を表すリストである必要があります。この方法はあなたの基本方法(native method) になります。この方法で取ってこないグループはすべて第二の(secondary) 選択方法で取得するグループか、または外部(foreign) グループです。

例えばNNTP サーバー‘`news.somewhere.edu`’ から毎日(薬のように) 一定の量のニュースを摂取したいのであれば、

```
(setq gnus-select-method '(nntp "news.somewhere.edu"))
```

のようにすることができます。

ローカル・スプールのディレクトリーを読み込みたい場合は、

```
(setq gnus-select-method '(nnspool ""))
```

のようにできます。

ローカルのスプールを使えるのであれば、かなりの確率でその方がずっと速いでしょうし、それを使うべきでしょう。でも、もしあなたのサーバーが Leafnode (それは簡単な個人用のニュースサーバーです) であるならばローカルスプールを使ってはいけません。この場合は`(nntp "localhost")` にしましょう。

もしこの変数が設定されていなければ、Gnus は`NNTPSERVER` 環境変数を読みにいきます。もしその変数が設定されていなければ、Gnus は`gnus-nntpserver-file` (設定されて

いない場合は/etc/nntpserver) がこの件に関して何かを言っていないかを調べます。もしそれも失敗したなら、Gnus は Emacs が動作しているサーバーをNNTP サーバーとして使おうとします。随分な当て推量ですけどね。

しかし、普段日常的には一つのNNTP サーバーを使い、違ったサーバーには興味のあるグループが少ししかない場合、グループバッファーで*B* 命令を使うことの方が良いでしょう。それは、選択可能なグループを表示し、その中からどれでも好きなものを購読することができます。これは.newsrc の保持をずっとやりやすくします。See Section 3.9 [Foreign Groups], p. 21.

外部グループに対する少し違ったやり方は、変数gnus-secondary-select-methods を設定する方法です。この変数に表されている選択方法は、多くの点でgnus-select-method サーバーの選択方法と同じように扱われます。起動中にアクティブファイルを探しにいき(もし要求されていれば)、これらのサーバー上にできた新しいニュースグループは元々のグループと同じように購読されます(もしくは、されません)。

例えばメールを読むためにnnmbox バックエンド(back end) を使いたいときは、普通この変数を、

```
(setq gnus-secondary-select-methods '((nnmbox "")))
```

と設定します。

2.2 サーバーが落ちている

ディフォルトのサーバーが落ちているときは、当然 Gnus の起動にいくつかの問題が発生します。しかし、ニュースグループの他にいくつかメールのグループがあるのならば、それにもかかわらず Gnus を起動する必要があるかもしれません。

信頼できるプログラムの一つである Gnus は、サーバーと接続できないときは基本選択方法なしで続けるかどうかを尋ねます。これは実際にはサーバーが存在しないとき(例えば、アドレスを間違えた場合) やサーバーが何らかの理由で一時的に調子がおかしくなっているときに起こります。もしそのまま続行することにして、外部グループが一つも無い場合、実はグループバッファーではほとんど何もできないということに気が付くでしょう。でも、ねぇ、それはあなたの問題です。ブブーッ!

サーバーが完全に落ちているのを知っているか、サーバーでわずらうことなくメールだけを読みたいときは、Gnus を起動するのにgnus-no-server 命令を使うことができます。急いでいるときにもぴったりでしょう。この命令は本来のサーバーには接続しません---その代わりに、レベル 1 と 2 にあるすべてのグループを活動状態にします(基本グループでないグループはその二つのレベルにしておくのが望ましいでしょう)。Section 3.6 [Group Levels], p. 19, も参照してください(訳注: gnus-no-server はgnus-group-use-permanent-levels 変数の値を 2 に設定することに注意してください)。

2.3 子どもの Gnus

あなたには二つ以上の Gnus をそれぞれ別の Emacs 上で同時に動かす必要が生じるかもしれません。違った.newsrc ファイルを使っているなら(例えば、二つの違ったサーバーから読み込むために、二つの違った Gnus を動作させている場合)、まったく問題はありません。それを行なえば良いだけです。

問題は、同じ.newsrc ファイルを使う二つの Gnus を動かそうとしたときに起こります。

この問題に対処するために Gnus タワーのシンクタンクにいる私たちは、新しい概念にたどりつきました。「親」と「子」です。

とにかく、*M-x gnus* (もしくは普段やっている方法) で Gnus を普通に起動します。それに続く子どもの Gnus はそれぞれ*M-x gnus-child* で起動します。子どもは普通の.newsrc ファイルは保存しませんが、代わりに「子どものファイル」に、子どもが動作しているときに購読したグループの情報だけを保存します。親の Gnus は、起動するときにそれらの子どものファイルを読み込み(そして消し)、すべての情報を取り込みます。(子どものファイルは、最終的な変更が優先されるようにそれらが作られた順番で読まれます。)

もちろん、子どものファイルからの情報は普通の(すなわち親の) .newsrc ファイルよりも優先されます。

子どもが起動するときにもし親の.newsrc* ファイル群がセーブされていなかったら、自動保存されたファイルを読むかどうかを尋ねられるでしょう。“yes” と答えると、親にセーブされていない変更は子どもに組み込まれます。“no” と答えると、親で読まれたいくつかの記事が、子どもでは未読であると見なされる場合があります。

2.4 新しいグループ

新しいニュースグループをまったく見なくても満足ならば、gnus-check-new-newsgroups をnil に設定することができます。これを設定した場合、起動にかかる時間が短くなります。この変数がnil に設定されていても、グループバッファーで*U* を押せばいつでも新しいグループを購読することができます(see Section 3.13 [Group Maintenance], p. 34)。ディフォルトではこの変数はask-server です。この変数がalways に設定されていると、*g* 命令を実行したときでも Gnus はバックエンドに新しいグループを探すことを求めます(see Section 3.18.1 [Scanning New Messages], p. 44)。

2.4.1 新しいグループを調べる

Gnus は、普通はグループが新しいかどうかを、購読しているグループと削除されているグループのリストとアクティブファイルを比較することにより判定しています。この方法は特に速いというわけではありません。gnus-check-new-newsgroups がask-server であると、Gnus はサーバーに、最後に接続してから新しいグループができているかどうかを尋ねます。この方法は速いし、安上がりです。これにより、削除されたグループ(see Section 3.6 [Group Levels], p. 19) のリストを保持しておくことからから完全に開放されます。ですから、gnus-save-killed-list をnil にすることができるでしょう。そうすれば、起動、終了の両方、そして全体にわたって時間を節約できます。ディスク消費量も少なくなります。それなら、どうしてこれがディフォルトではないのでしょう? 残念ながら、すべてのサーバーがこの命令を理解するわけではないのです。

私は今あなたが何を考えているかを当てられます。どうすればサーバーがask-server を理解するかがわかるのでしょう? え、違うのですか? あぁ、良かった。というのは、確実な答は存在しないのです。私に言えることは、この変数をask-server に設定して、数日間新しいグループが現れるかどうかを調べてください、ということだけです。もしいくつかのグループが現れたなら、それで動作しています。一つも現れなければ、それは動作していません。私は、Gnus にサーバーがask-server を理解するかどうかを推量させる関数を書くこともできますが、それは単に推量しているにすぎません。ですから、その関数を書くことはないでしょう。他の方法としては、サーバーにtelnet をして、HELP と打ち、サーバーが理解するコマンドの中に‘NEWGROUPS’ があるかどうかを調べることもできます。もしあれば、おそらく動作するでしょう(しかし、適切に機能を提供することなく‘NEWGROUPS’ をリストに含めるサーバーもあります)。

この変数は、選択方法のリストであることもできます。そのときは、Gnus はask-server命令をそれぞれの選択方法に対して実行し、普通の方法で購読します(もしくは、しません)。これの副作用は、起動にかなり時間がかかるので、待っている間に瞑想できることです。永久の幸福を達成するために、マントラ“dingnusdingnusdingnus”を使ってください。

2.4.2 購読方法

新しいグループに遭遇したときに Gnus が何をするかは、変数gnus-subscribe-newsgroup-method によって決定されます。

この変数は関数を含んでいる必要があります。この関数は新しいグループの名前を唯一の引数として呼ばれます。

いくつかの手軽なプレハブ関数は、以下のようになっています。

gnus-subscribe-zombies
: すべての新しいグループをゾンビ(zombie) にします(see Section 3.6 [Group Levels], p. 19)。これがディフォルトになっています。後でゾンビを(*A z* によって) 概観したり、(*S z* によって) 適切にすべてを削除したり、(*u* によって) 購読したりできます。

gnus-subscribe-randomly
: 任意の順番ですべての新しいグループを購読します。実際には、すべての新しいグループはグループバッファーの『一番上』に加えられます。

gnus-subscribe-alphabetically
: すべての新しいグループをアルファベット順に購読します。

gnus-subscribe-hierarchically
: すべての新しいグループを階層的に購読します。この関数とgnus-subscribe-alphabetically の違いは少ししかありません。gnus-subscribe-alphabetically は新しいグループを厳密にアルファベット順にならべますが、この関数はグループをその階層の中に入れます。ですから、‘rec’ の階層を‘comp’ の階層の前に持ってきたい場合、この関数はその配置をぐちゃぐちゃにはしません。もしくは、そのようなものです。

gnus-subscribe-interactively
: 新しいグループを対話的に購読します。これは Gnus が**すべて** のグループに対して尋ねることを意味しています。購読するグループは階層的に購読されます。

gnus-subscribe-killed
: すべての新しいグループを削除します。

gnus-subscribe-topics
: グループを、それに合致するsubscribe トピックパラメーターを持っているグループに入れます(see Section 3.16.5 [Topic Parameters], p. 40)。例えば、以下のようなsubscribe パラメーター

```
"nnml"
```

は、その正規表現に合致するすべてのグループはそのトピックの下で購読されるということです。

グループに合致するトピックが無い場合、グループは最上位のトピックで購読されます。

上の変数と密接に関係する変数は、gnus-subscribe-hierarchical-interactive です。この変数がnil でないと、Gnus は階層的な方法で新しいグループを購読するかどうかを尋ねます。Gnus はそれぞれの階層で、それを下に降りるかどうかを尋ねます。

よくある間違いは、数段落前の(gnus-subscribe-newsgroup-method) 変数をgnus-subscribe-herarchical-interactive に設定することです。これは誤りです。これは動作しません。これはおめでたい人のすることです。ですから、絶対にしないでください。

2.4.3 新しいグループを選別する

どの新しいグループが購読(もしくは、無視) されるべきかを管理する快適で手軽な方法は、.newsrc ファイルの先頭に*options* 行を挿入することです。次は、例です。

```
options -n !alt.all !rec.all sci.all
```

この行は、明らかにまじめで理知的で科学的な人間(あるいは彼女はどこにでもいる単につまらない人かもしれないけれど) が書いたものです。なぜなら、これは‘alt’ と‘rec’ で始まる名前を持つグループはすべて無視され、‘sci’ で始まる名前を持つグループはすべて購読する、ということを表しているからです。Gnus はこれらのグループを購読するのに普通の購読方法を使いません。代わりにgnus-subscribe-options-newsgroup-method が使われます。この変数はディフォルトでgnus-subscribe-alphabetically になります。

“options -n” の形式は単純過ぎます。上記の構文はサポートしているもののすべてです: つまり、あるニュース階層を強いて購読することもできるし、あるニュース階層を購読対象から除外することもできるということです。

.newsrc ファイルをいじりたくない場合は、gnus-options-subscribe とgnus-options-not-subscribe の二つの変数だけを設定することもできます。この二つの変数は.newsrc ファイルの‘optinos -n’ 行とまったく同じことをします。どちらの変数も正規表現で、新しいグループは前者に合致すれば無条件に購読され、後者に合致すると無視されます。

さらにここでおせっかいをする変数は、gnus-auto-subscribed-groups です。それはgnus-options-subscribe とまったく同じように動作するので、本当は余分なものです。しかし、私はこの二つがあった方が良いと思いました。もう一方の変数は利用者がいじくるのに使われるのに対して、この変数はいくつかの基本的な規則を設定するためのものです。ディフォルトではこの変数はメールバックエンド(nnml, nnbabyl, nnfolder, nnmbox, nnmh, nnimap およびnnmaildir) からできるすべての新しいグループを購読するようになっています。それが嫌であれば、この変数をnil に設定してください。

これだけでも十分ですが、さらにgnus-auto-subscribed-categories は、それらの選択方法が属するカテゴリーに基づいて購読されるべきであることを明示できるようにします。ディフォルトは‘(mail post-mail)’ で、メールのようなバックエンドに基づく新しいグループは、すべて自動的に購読されるはずです。

これらの変数に合致する新しいグループはgnus-subscribe-options-newsgroup-method を使って購読されます。

2.5 サーバーを換える

ときどき、あるNNTP サーバーから別のサーバーへ移動しなければならないことがあります。このようなことはめったにおきませんが、おそらくあなたが仕事を変えたり、使っているサーバーがとても不安定で、別のものに乗り換えたいというときに必要になるでしょう。

サーバーを変更するのはとても簡単ですよね? gnus-select-method を新しいサーバーを指し示すように変更すればいいだけですね?

違います*!*

記事の番号は違ったNNTP サーバーでも(どうにかして) 同じにしてあるということはありません。そして、Gnus がどの記事を読んだかを記録する唯一の方法は、記事番号を記録することです。ですからgnus-select-method を変更したときは、.newsrc ファイルは役に立たなくなります。

M-x gnus-group-clear-data-on-native-groups コマンドを使って、基本グループに関するデータをすべて消去することができます。注意して使ってください。

gnus-group-clear-data コマンドは現在のグループのすべてのデータをクリアします---印と既読記事のリストを消し去ります。

サーバーを変更した後で、キャッシュ階層を移動させなければなりません。というのは、キャッシュ記事は間違った記事番号になっており、それは Gnus がどの記事を読んだとみなすかに影響します。gnus-group-clear-data-on-native-groups はそれを自動で行なってしまうかどうかを尋ねます。gnus-group-clear-data では*M-x gnus-cache-move-cache* が使えます(でも気を付けて、それはすべてのグループのキャッシュを移動してしまいますから)。

2.6 起動ファイル

最もありふれた Unix のニュースリーダーは、.newsrc と呼ばれる共用の起動ファイルを使います。このファイルは、購読しているグループと、それらのグループにおいてどの記事が読まれたかの、すべての情報を持っています。

GNUS ではものごとが少々複雑になっています。.newsrc ファイルを最新のものにするだけではなく、.newsrc ファイルには合わない情報を保存しておくために.newsrc.el と呼ばれるファイルを使います。(実際は.newsrc ファイルのすべての情報を複製して保持しています。) GNUS はこれらの中で一番最後に保存されたものを使います。これをすることにより、GNUS と他のニュースリーダーを切り替えて使うことができます。

これはちょっと間が抜けているので、Gnus はもっと良い方法を編み出しました。.newsrc と.newsrc.el ファイルに加えて、Gnus は.newsrc.eld と呼ばれるファイルも持っています。Gnus はこれらの中で一番新しいファイルを読みますが、.newsrc.el ファイルに書き込むことはありません。.newsrc.eld ファイルは絶対に消すべきではありません。---それは.newsrc ファイルにはないたくさんの情報を保持しています。

gnus-save-newsrc-file をnil にすることによって.newsrc ファイルに書き込むのを止めることができます。そうすれば、そのファイルを削除することができ、ディスク容量を節約することができ、Gnus の終了が速くなります。しかし、そうすると他のニュースリーダーを使えなくなります。でも、ちょっと、誰かそうしたい人がいるでしょうか。同じようにgnus-read-newsrc-file をnil にすることによって、Gnus は.newsrc ファイルとすべての.newsrc-SERVER ファイルを無視するようになります。そのことは、あなたが時々違うニュースリーダーを使ったり、利用可能なグループの異なるサブセットをそれらのニュースリーダーで読みたい場合に、便利なことがあります。

gnus-save-killed-list (ディフォルトはt) がnil であると、Gnus は削除されたグループを起動ファイルに保存しません。これは(起動時と終了時の) 時間と、(ディスクの) 容量を節約します。こうすると Gnus がどのグループが新しいかの記録を持っていないことになるので、新しいグループの自動購読方法は意味が無くなります。この変数をnil にしたときは、gnus-check-new-newsgroups を常にnil かask-server にしておくべきでしょう(see Section 2.4 [New Groups], p. 5)。この変数は正規表現であることもできます。その

ような場合は、ファイルを保存する直前にその正規表現に合致しないすべてのグループを消去します。これは、すべてのサーバーが`ask-server` を理解するわけではない、といったような、いくらかあいまいな状況のときに役に立つでしょう。

変数`gnus-startup-file` は起動ファイルがどこにあるかを指定します。ディフォルト値は`~/.newsrc` で、それがどのようなものであれ、末尾に‘`.eld`’ を付けたものが Gnus (El Dingo) の起動ファイルになります。このファイルの番号付きバックアップを残しておきたいときは`gnus-backup-startup-file` をセットしてください。それは`version-control` 変数と同じ値を取ります。

`gnus-save-newsrc-hook` は各種の newsrc ファイルのどれかを保存する前に実行されるのに対し、`gnus-save-quick-newsrc-hook` は`.newsrc.eld` ファイルを保存する前に実行され、`gnus-save-standard-newsrc-hook` は`.newsrc` ファイルを保存する前に実行されます。後の二つは普通はバージョン制御を on/off するのに使われます。ディフォルトでは、起動ファイルを保存するときにバージョン制御が行なわれます。バックアップファイルの作成を止めたいときは、次のようにしてください。

```
(defun turn-off-backup ()
  (set (make-local-variable 'backup-inhibited) t))

(add-hook 'gnus-save-quick-newsrc-hook 'turn-off-backup)
(add-hook 'gnus-save-standard-newsrc-hook 'turn-off-backup)
```

Gnus が起動すると、`gnus-site-init-file` (ディフォルトで`.../site-lisp/gnus-init`) と`gnus-init-file` (ディフォルトで`~/.gnus`) のファイルを読み込みます。これらは普通の Emacs Lisp ファイルで、`~/.emacs` や`site-init` ファイルを Gnus 関係のもので乱雑にしないようにするために使うことができます。Gnus はこれらと同じ名前のファイルに、接尾語`.elc` と`.el` が付いているものも調べます。言い換えれば、`gnus-init-file` を`~/.gnus` に設定すると、Gnus は`~/.gnus.elc`, `~/.gnus.el` を探し、最後に`~/.gnus` を(この順番に) 探します。`-q` または`--no-init-file` オプション(see Section “Initial Options” in *The Emacs Editor*) が指定されて Emacs が起動された場合、Gnus は`gnus-init-file` を読み込みません。

2.7 自動保存

何か Gnus のデータを変更すること(記事を読む、印を付ける、グループを削除または購読する) をしたとき、変更は特別な「ドリブルバッファー」(dribble buffer) に書き込まれます。このバッファーは Emacs が普通するように自動保存されます。`.newsrc` ファイルを保存する前に Emacs が落ちたときは、すべての変更をこのファイルから回復することができるでしょう。

起動時に Gnus がこのファイルの存在を発見すると、Gnus はそれを読み込むかどうかを利用者に尋ねます。本当の起動ファイルが保存されれば、自動保存ファイルは削除されます。

`gnus-use-dribble-file` が`nil` であると、Gnus はドリブルバッファーを作ったり、維持したりしません。ディフォルトは`t` です。

Gnus はドリブルファイルを`gnus-dribble-directory` に置きます。ディフォルトではそのようになっていますが、この変数が`nil` であると、Gnus は`.newsrc` ファイルの置かれているディレクトリー(これは普通は利用者のホームディレクトリーです) に入っていってドリブルファイルを作ります。ドリブルファイルは`.newsrc` と同じ許可属性を与えられます。

もしgnus-always-read-dribble-file がnil でなければ、Gnus は利用者に尋ねること無く、ドリブルファイルを起動時に読み込みます。

2.8 アクティブファイル

Gnus は起動したときや、実際に新しい記事が到着しているかを判定しようとするときに、アクティブファイルを読み込みます。これはとても大きなファイルで、そのサーバーの活動中のグループと記事のすべてのリストが入っています。

アクティブファイルを検査する前に、Gnus は正規表現gnus-ignored-newsgroups に合うすべての行を削除します。これは主に偽の名前を持つグループを排除するために使われてきましたが、興味の無いグループの階層を無視するために使うこともできます。しかし、これはお勧めできません。本当のことを言うと、まったく賛成できません。代わりに、そのような用途に用いられる変数の概略を知るために、Section 2.4 [New Groups], p. 5, を参照してください。

アクティブファイルは比較的大きくなる傾向があるので、遅い回線を使っているときは、アクティブファイルを読み込まないようにgnus-read-active-file をnil に設定することができます。この変数はディフォルトではsome です。

そのような時は、Gnus は実際に購読されているグループに関する情報だけを得てやっていこうとします。

気を付けてほしいのは、あなたが山ほどのたくさんのグループを購読しているときにこの変数をnil に設定すると、Gnus は速くなるどころか遅くなってしまうということです。現状では、ニュースを 2400bps 以上のモデムを通して読んでいるのでない限り、Gnus の速度はかなり遅くなるでしょう。

この変数はsome という値も取ることができます。その時は、Gnus は購読しているグループに関する情報をだけを得ようとします。いくつかのサーバー(LIST ACTIVE group 命令を使うことのできる、最新鋭の INN サーバー) では、非常に早くなるでしょうが、他のサーバーでは速くはありません。どのようにせよ、遅い回線ではsome はnil よりも速く、それはもちろんt よりも速くなります。

いくつかのニュースサーバー(例えば古い Leafnode や古い INN) にはLIST ACTIVE group 命令がありません。そういうサーバーにはnil をこの変数の値に設定するのが、おそらくもっとも有効でしょう。

もしこの変数がnil であると、Gnus は完全にがんじがらめの方法でグループの情報を得ようとします。そして、これはあまり速くありません。もしそれがsome でNNTP サーバーを使っているときは、Gnus はできるだけ速く命令を出し、一撃ですべての返答を読み込みます。この方が普通はより良い結果をもたらしますが、サーバーがLIST ACTIVE group 命令を理解しないなら、サーバーにとってはあまり良いとは言えません。

Gnus の起動にあまりに時間がかかると思ったなら、この変数にこれらの三つの違った値を試してみて、どれが一番良いかを探してください。

some かnil を使うのであれば、どちらにしろ速度を上げるためにすべての興味の無いグループを必ず削除するべきでしょう。

この変数は第二の(secondary) 選択方法のアクティブファイル取得にも影響することに気を付けてください。

2.9 起動変数

`gnus-before-startup-hook`
: Gnus が起動するとき、最初に呼ばれるフックです。

`gnus-before-resume-hook`
: 一時停止させられていた Gnus が再開するときに最初に呼ばれるフックです。

`gnus-startup-hook`
: Gnus が起動された後に、一番最後に実行されるフックです。

`gnus-started-hook`
: Gnus の起動に成功した後に、一番最後に実行されるフックです。

`gnus-setup-news-hook`
: `.newsrc` ファイルを読み込んだ後で、グループバッファーを作成する前に実行されるフックです。

`gnus-check-bogus-newsgroups`
: もし`nil` でないと、Gnus は起動時にすべての偽グループを調べて削除します。「偽グループ」(bogus group) はあなたの`.newsrc` ファイルには存在するけれど、ニュースサーバーには実際には存在しない、というグループのことです。偽グループを調べるのにはかなり時間がかかるので、時間と資源を節約するために、この機能は使わないほうがいいでしょう。そして、代わりにグループバッファーで時々偽グループを調べるのが良いでしょう(see Section 3.13 [Group Maintenance], p. 34)。

`gnus-inhibit-startup-message`
: もし`nil` でないと、起動時のメッセージは表示されません。そのようにすれば、仕事の代わりにニュースを読んでいるのを上司に気付かれにくくなるでしょう。この変数は`~/.gnus.el` がロードされる前に使われるので、`.emacs` に設定するべきである点を注意してください。

`gnus-no-groups-message`
: グループが一つも存在しないときに Gnus が表示するメッセージです。

3 グループバッファー

グループバッファー(*group buffer*) は有効なグループを全部(あるいは一部を) 一覧表示します。これは Gnus を起動したときに最初に表示されるバッファーで、Gnus が生きている限り決して消されることはありません。

3.1 グループバッファーの形式

グループモードのツール・バーをカスタマイズすることができます。*M-x customize-apropos* RET *gnus-group-tool-bar* を試してみてください。

ツール・バーのアイコンは、今ではカーソルの位置に応じて正しく有効に、または無効にされるので、グループバッファー内での移動は遅くなります。これは変数gnus-group-update-tool-bar で禁止することができます。そのディフォルト値は Emacs のバージョンに依存しています。

3.1.1 グループ行の仕様

グループバッファーのディフォルトの形式はきれいでつまんないけど、これは君の好きなように、サイコーにダサくすることもできます。

これがグループ行の例です。

```
        25: news.announce.newusers
    *    0: alt.fan.andrea-dworkin
```

とっても簡単でしょ?

‘news.announce.newusers’ には 25 の未読記事があるのがわかります。‘alt.fan.andrea-dworkin’ には未読記事はないけれども、印を付けた記事がいくつかあります(行頭のちっちゃなアスタリスクが見える?)。

この形式はgnus-group-line-format 変数をいじることで、どんな風にでも変えられます。この変数はformat の仕様風に動作します。つまり(あのクソ) C 言語を使う人たちのための printf の仕様とほぼ同じです。See Section 10.4 [Formatting Variables], p. 272.

上記の行を生成するのは‘%M%S%5y:%B%(%g%)\n’ という値です。

コロンは、この行の中に必ず無くてはいけません。カーソルは何かの操作をした後は常にコロンのところに移動するからです。See Section 10.4.6 [Positioning Point], p. 276. 他には何も必要ではありません---グループ名さえもです。表示されている文字はすべてただの画面の飾りであり、Gnus がそれを調べることはありません。Gnus は必要とするすべての実情報を、テキスト属性を使って憶えています。

(もし君が、すごくヘンな、素晴らしい、表計算風のレイアウトを作ったとしたら、みんな、君は会計の仕事が忙しくって、ニュースを読んで時間を無駄使いしたりなんかしてない、って信じてくれるよ。)

以下が使用できるフォーマット文字のリストです。

‘M’ そのグループに印の付いた記事しか無いときは、アスタリスク文字。

‘S’ そのグループが購読されているかどうか。

‘L’ 購読度のレベル。

‘N’ 未読記事の数。

‘I’ 保留記事の数。

‘T’ 印付き記事の数。

‘R’ 既読記事の数。

‘U’ まだ読まれたことが無い記事の数。

‘t’ 推定全記事数(これは実際は*max-number* - *min-number* + 1)。

Gnus がこの推定を使うのは、NNTP プロトコルは能率の良い*max-number* と*min-number* へのアクセスを提供するものの、本当の未読記事の数を得るには必ずしも能率的ではないからです。ヒステリックなレーズン(訳注:「歴史的な理由」のモジり) により、メールバックエンドにおいても、限定された同じインターフェースを使って、本当の未読記事の数を能率的に得ることはできるかもしれません。この制限を Gnus から取り払うことはバックエンドのインターフェースを変更することを意味し、それは楽な仕事ではありません。

`nnml` バックエンド(see Section 7.4.13.3 [Mail Spool], p. 188) には、この欠陥を巧みに補う「グループ圧縮」(group compaction) という機能があります。それは、記事の番号を 1 から順に振り直してすきまを取り除けば正しい全記事数を得ることができる、という着想によります。将来は他のバックエンドもこれをサポートするかもしれません。全記事数をまあまあ最新の状態にしておくためには、時々グループを(またはサーバーのディレクトリーを) 圧縮する必要があるでしょう。See Section 3.18 [Misc Group Stuff], p. 43, See Section 7.1.2 [Server Commands], p. 147.

‘y’ 未読でも、印付きでも、保留でもない記事の数。

‘i’ 印付き記事と保留記事の数。

‘g’ グループ名のフルネーム。

‘G’ グループ名。

‘C’ グループのためのコメント(see Section 3.10 [Group Parameters], p. 23)、またはグループパラメーターにコメントの要素が無い場合はグループ名。

‘D’ ニュースグループの説明。これらが現れる前に、グループの説明を読む必要があります。それには`gnus-read-active-file` を設定するか、グループバッファーで*M-d* コマンドを使ってください。

‘o’ 司会者付きの場合‘m’。

‘O’ 司会者付きの場合‘(m)’。

‘s’ 選択方法。

‘B’ そのグループの概略バッファーが開いているかどうか。

‘n’ どこからの選択か。(訳注: バックエンドのシンボル名)

‘z’ 外部選択方法が使われている場合、‘<%s:%n>’ と同じ文字列。

‘P’ トピック(see Section 3.16 [Group Topics], p. 35) のレベルに応じた字下げ。

‘c’ 短い(省略した) グループ名。`gnus-group-uncollapsed-levels` 変数は、どのレベルまでグループ名を全部残すかを示します。ディフォルトは 1 です---この意味は、‘gnu.emacs.gnus’ のようなグループ名を‘g.e.gnus’ に短縮するということです。

‘m’ そのグループに最近新着メールが届いている場合は‘%’ (gnus-new-mail-mark)。

‘p’ ‘#’ (gnus-process-mark) で、そのグループにプロセス印が付いていることを示します。

‘d’ 最後にいつこのグループを読んだかを示す文字列(see Section 3.18.3 [Group Timestamp], p. 45)。

‘F’ キャッシュとエージェントの両方によって取得された記事がディスクに占める容量。値はカラム幅を最小にするために、自動的にバイト(B)、キロバイト(K)、メガバイト(M)、またはギガバイト(G) に縮尺されます。固定幅カラム用には %7F の形式で足ります。

‘u’ 利用者定義指定。フォーマット文字列中で、この次の文字はアルファベット文字でなければいけません。Gnus はgnus-user-format-function-‘X’ 関数を呼び出します。ここで‘X’ は‘%u’ に続いている文字です。この関数は引数に一つのダミーパラメーターを渡されます。この関数は、他の各指定文字の情報と同様に、バッファーに挿入される文字列を返さなければなりません。

すべての「〜の数」の指定は、もしその情報が利用できない場合にはアスタリスク(‘*’)で埋められます---例えば、起動されていない外部グループや、不正な基本グループの場合です。

3.1.2 グループモード行の仕様

モード行はgnus-group-mode-line-format (see Section 10.4.2 [Mode Line Formatting], p. 273) を設定することで変更できます。こいつは指定文字をあんまりたくさん知っていません。

‘S’ 基本ニュースサーバー。

‘M’ 基本選択方法。

3.1.3 グループのハイライト

グループバッファーのハイライトはgnus-group-highlight 変数によって制御されます。これは(*form* . *face*) のようなものを要素に持つ連想リストです。*form* が評価された結果が、nil 以外の何かになると、その行に対して*face* が使用されます。

以下がこの変数の値の例です。これは背景が暗い設定ではきれいに見えるかもしれません。

```
(cond (window-system
       (setq custom-background-mode 'light)
       (defface my-group-face-1
         '((t (:foreground "Red" :bold t))) "First group face")
       (defface my-group-face-2
         '((t (:foreground "DarkSeaGreen4" :bold t))) "Second group face")
       (defface my-group-face-3
         '((t (:foreground "Green4" :bold t))) "Third group face")
       (defface my-group-face-4
         '((t (:foreground "SteelBlue" :bold t))) "Fourth group face")
       (defface my-group-face-5
         '((t (:foreground "Blue" :bold t))) "Fifth group face")))
```

```
(setq gnus-group-highlight
      '(((> unread 200) . my-group-face-1)
        ((and (< level 3) (zerop unread)) . my-group-face-2)
        ((< level 3) . my-group-face-3)
        ((zerop unread) . my-group-face-4)
        (t . my-group-face-5)))
```

Section 10.7 [Faces and Fonts], p. 282, も参照してください。

この form が評価されるときに動的に束縛されている変数には以下のものがあります。

`group` グループ名。

`unread` そのグループの未読記事の数。

`method` 選択方法。

`mailp` そのグループがメールのグループかどうか。

`level` そのグループのレベル。

`score` そのグループのスコア。

`ticked` そのグループ中の印の付いた記事の数。

`total` そのグループ中の全記事数。もっと正確に言うと、*max-number* マイナス*min_number* プラス 1。

`topic` トピックマイナーモードを使用している時、この変数は挿入されている現在のトピックに束縛されます。

この form が評価(`eval`) されるときは、ポイントは問題のグループの行頭にあります。従って、通常の Gnus の関数のほとんどを使ってそのグループの情報を取ってくることができます。

`gnus-group-update-hook` はグループ行が変更されたときに呼び出されます。これは`gnus-visual` が`nil` のときは呼び出されません。

3.2 グループ操作

すべての移動コマンドは数値接頭引数を理解するので、期待する通りの動作をします。たぶんね。

n 次の未読記事のあるグループに移動します(`gnus-group-next-unread-group`)。

p
DEL 一つ前の未読記事のあるグループに移動します(`gnus-group-prev-unread-group`)。

N 次のグループに移動します(`gnus-group-next-group`)。

P 一つ前のグループに移動します(`gnus-group-prev-group`)。

M-n 一つ前の同じレベル(もしくはそれより小さいレベル) の未読グループに移動します(`gnus-group-prev-unread-group-same-level`)。

M-p 次の同じレベル(もしくはそれより小さいレベル) の未読グループに移動します(`gnus-group-next-unread-group-same-level`)。

次の三つの命令はグループにジャンプするためのものです:

j グループにジャンプします(それが見えるようになっていなかったら見えるようにします) (gnus-group-jump-to-group)。kill されているグループも、生きているグループと同様にジャンプできます。

, 最も小さいレベルの未読グループにジャンプします(gnus-group-best-unread-group)。

. 最初の未読記事のあるグループにジャンプします(gnus-group-first-unread-group)。

gnus-group-goto-unread をnil にすると、すべての移動コマンドは、次の未読グループではなく次のグループに移動するようになります。そのコマンドが次の未読グループに移動すると言い張っていてもです。ディフォルトはt です。

概略バッファーを出たときにgnus-summary-next-group-on-exit がt だったら、グループバッファーで次の未読のグループに移動します。それ以外の場合は出たグループに留まります。ディフォルトはt です。

> 訳注: gnus-group-goto-unread が次の未読グループか単に次のグループのどちらに移動するかを指定するのに対してgnus-summary-next-group-on-exit は移動するかしないかを指定するために使います。ただし後者は*q* (または*Z Q*, gnus-summary-exit) で概略バッファーを出たときだけに効果を及ぼします。

3.3 グループの選択

SPC 現在のグループを選択し、概略バッファーに切り替えて最初の未読記事を表示します(gnus-group-read-group)。もしそのグループに未読記事が無い、もしくはこの命令に数値以外の接頭引数を与えると、Gnus はサーバーからこのグループのすべての古い記事を取得しようとします。*n* の数値接頭引数を与えると、Gnus の取得する記事数は*n* になります。*n* が正の数であれば Gnus は新しい方から*n* 個の記事を取得し、*n* が負の数であれば Gnus は古い方からabs(n) 個の記事を取得します。

したがって、SPC では普通にグループに入り、*C-u* SPC では古い記事が現れます。*C-u 4 2* SPC では 42 個の最新の記事を取得し、*C-u - 4 2* SPC では 42 個の最も古い記事を取得します。

グループにいる(概略バッファーにいる) ときは、*M-g* で新しい記事を取得できるし、*C-u M-g* では古い記事を表示することができます。

RET 現在のグループを選択し、概略バッファーに切り替えます(gnus-group-select-group)。gnus-group-read-group と同じ引数を取ります---唯一の違いは、グループに入ったときに最初の未読記事を表示しない、ということです。

M-RET これは上記のコマンドと同じ動作をしますが、「ゴタゴタ」は最低限にしようとします(gnus-group-quick-select-group)。スコア・kill の処理は行なわれず、ハイライトも記事消去もしません。これは、あなたが本当に急いでいて、どっかのやたらでっかいグループに入らなければいけないときに役に立つかもしれません。また、接頭引数に 0 を与えれば(すなわち*0 M*-RET)、Gnus は概略バッファーを作ろうとさえしません。これは概略バッファーを作る前にスレッド表示を切り替えたいとき役に立ちます(see Section 4.27.3 [Summary Generation Commands], p. 118)。

M-SPC これはRET コマンドと同じ動作をするさらにもう一つのコマンドですが、このコマンドは記事消去と保留記事を隠す処理を行ないません(gnus-group-visible-select-group)。

C-*M*-RET 最後にこのコマンドは、現在のグループを一度限り、その内容に一切の処理をすることのないように選択します(gnus-group-select-group-ephemerally)。スレッド表示さえも行なわれません。この方法で選択した後にこのグループに対して行なったことはすべて、その後に影響を与えることはありません。

gnus-large-newsgroup 変数は、何を大きなグループと考えるべきかを Gnus に与えます。nil だったら、どのグループも大きいと考えません。ディフォルト値は 200 です。グループに(未読と可視の) 記事がこの数以上あれば、Gnus はそのグループに入る前に利用者に確認を求めます。利用者はサーバーからいくつの記事を取得するかを指定できます。もし利用者が負の数($-n$) を指定すれば、古い方からn 個の記事を取得します。正の数であれば、新しく到着した方からn 個の記事を取得します。

gnus-large-ephemeral-newsgroup はgnus-large-newsgroup と同じですが、一時ニュースグループのためにだけ使われます。

いくつかのニュースサーバーのとあるグループでは、期限切れ消去されない少数の非常に古い記事と最新のものとの間に大きな隙間があるかもしれません。そういう場合、そのサーバーはLIST ACTIVE group コマンドに対して例えば(1 . 30000000) のようなデータを返すでしょう。たとえ実際には 1〜10 と 29999900〜30000000 の記事しか無くても、Gnus は最初からそのことを知っているわけではないので 30000000 通の記事を受け取るための準備をします。しかしそれでは何百メガバイトのメモリーを消費してしまうし、場合によっては Emacs を立ち往生させてしまうかもしれません。もしそのようなサーバーを使うのであれば、変数gnus-newsgroup-maximum-articles を正の数に設定してください。値は、あらゆるグループでその数の最新の記事以外を Gnus が無視することを意味します。例えば 10000 という数は Gnus に 29990001〜30000000 の記事だけを取得させるようにします(最新の記事番号が 30000000 だった場合です)。この変数に数値を設定すると、非常に古い記事を読むことができなくなってしまうかもしれないことに注意してください。変数gnus-newsgroup-maximum-articles のディフォルト値はnil で、その場合 Gnus は古い記事を無視しません。

もしgnus-auto-select-first が非-nil だったら、SPC コマンドでグループに入ったときに自動的に記事を選択します。どの記事が選択されるかは、変数gnus-auto-select-subject で制御されます。この変数に設定できる有効な値は:

unread 最初の未読記事の表題の行にポイントを移動させます。

first 最初の記事の表題の行にポイントを移動させます。

unseen まだ読まれたことが無い最初の記事の表題の行にポイントを移動させます。

unseen-or-unread
まだ読まれたことが無い最初の記事があれば、その記事の表題の行にポイントを移動させ、無かったら最初の未読記事の表題の行にポイントを移動させます。

best スコアが最も高い未読記事の表題の行にポイントを移動させます。

この変数は関数であることもできます。その場合、その関数は表題の行にポイントを移動させるために呼ばれます。

もしあるグループで自動記事選択をやめたいのであれば(例えばでっかい記事のあるバイナリーグループでは、とか)、グループが選択されたときに呼び出されるgnus-select-group-hookの中で変数gnus-auto-select-first をnil に設定することができます。

3.4 購読制御コマンド

以下のコマンド群は、グループバッファーで(それぞれのグループを) 購読するかどうかの管理をできるようにします。もしたくさんのグループを購読したいのであればSection 7.1 [Server Buffer], p. 146, に行って、そこでRET またはSPC を使ってサーバーを選ぶ方がおそらくもっと便利かもしれません。その後Section 3.14 [Browse Foreign Server], p. 34, で列挙しているコマンドが、手元で使えるようになります。

S t
u 現在位置のグループを購読する/しないを切り替えます(gnus-group-toggle-subscription-at-point)。

S s
U グループを指定して、それを購読する/しないを切り替えます(gnus-group-toggle-subscription)。

S k
C-k 現在のグループを kill します(gnus-group-kill-group)。

S y
C-y 最後に kill したグループを yank します(gnus-group-yank-group)。

C-x C-t 二つのグループの順序を置き換えます(gnus-group-transpose-groups)。これは本当は購読コマンドではありませんが、kill と yank を何度か続ける代わりにこのコマンドが使えます。

S w
C-w リージョン内のすべてのグループを kill します(gnus-group-kill-region)。

S z すべてのゾンビグループを kill します(gnus-group-kill-all-zombies)。

S C-k あるレベルのグループをすべて kill します(gnus-group-kill-level)。kill した後、これらのグループを yank で戻すことはできないので、このコマンドはいくらか注意して使ってください。このコマンドが本当に便利になるのは、.newsrc に捨ててしまいたい未購読のグループがたくさんあるときだけです。レベル 7 で*S C-k* を行なうと、.newsrc ファイル中にメッセージ番号がない未購読グループをすべて kill します。

Section 3.6 [Group Levels], p. 19, も参照してください。

3.5 グループデータ

c そのグループ内のすべての無印の記事を既読にします(gnus-group-catchup-current)。グループバッファーから既読にした場合はgnus-group-catchup-group-hook が呼び出されます。

C そのグループの全記事を、印付きの記事も含めて既読にします(gnus-group-catchup-current-all)。

M-c 現在のグループのすべてのデータをクリアします---印と既読記事のリストを消し去ります(gnus-group-clear-data)。

M-x gnus-group-clear-data-on-native-groups
もしNNTP サーバーを別のものに切り替えたとすると、すべての印と既読情報はもう役には立ちません。このコマンドを使って基本グループのすべてのデータをクリアすることができます。注意して使ってね。

3.6 グループレベル

すべてのグループは「購読度」(*subscribedness*) のレベルを持ちます。例えば、あるグループがレベル 2 だとすれば、それはレベル 5 のグループよりも「より購読している」ということです。Gnus に対して、あるレベルかそれより小さいレベルのグループのみ一覧表示するように頼むこともできるし(see Section 3.11 [Listing Groups], p. 31)、あるレベルかそれより小さいレベルのグループの新着記事のみを確認することもできます(see Section 3.18.1 [Scanning New Messages], p. 44)。

忘れないで: グループのレベルが大きいほど、重要度は低くなるということ。

S l 現在のグループのレベルを設定します。数値の接頭引数が与えられると、そこから*n* 個のグループのレベルが設定されます。レベルを入力するためのプロンプトが出ます。

Gnus はレベル 1 からgnus-level-subscribed (この値を含む) (ディフォルトは 5) までのグループを購読、gnus-level-subscribed (この値を含まない) からgnus-level-unsubscribed (この値を含む) (ディフォルトは 7) までのグループを非購読、gnus-level-zombie をゾンビ(歩く屍) (ディフォルトは 8)、gnus-level-killed を kill されている(完全に死んでいる) (ディフォルトは 9) と判断します。Gnus は購読と非購読のグループはまったく同様に扱いますが、ゾンビと kill グループは、どの記事を読んだか、存在するかなどの情報を一切記憶しません。この死んでいるグループと生きているグループの区別は、別にそれがきれいだからとか賢いからというわけではなく、純粋に効率的な理由のためです。

メール用のグループは(もしあれば) 非常に小さいレベル(例えば 1 か 2) にしておくことをお勧めします。

次の Gnus のディフォルトの動作の説明は、ことによると、これらのレベルのすべてを理解する助けになるかもしれません。ディフォルトでは、Gnus は購読している空でないグループを表示しますが、*L* を叩くことによって空のグループや非購読のグループも表示させることができます。つまり、非購読のグループは隠されている、と言っても良いでしょう。

ゾンビと kill グループは、ディフォルトでは隠されている点で非購読のグループに似ています。しかし、Gnus がニュースサーバーに対してゾンビと kill グループに関する情報(記事数、未読記事数) の問い合わせをしない点で、購読および非購読のグループとは違っています。ふつう、あなたは興味の無いグループを*C-k* で kill しますよね。もし、ほとんどのグループが kill されていると、Gnus は速くなります。

なぜ Gnus はゾンビと kill グループを区別するのでしょう? ええと、サーバーに新しいグループができると、Gnus はディフォルトでそれをゾンビにします。これは、あなたがふつうは新しいグループに煩わされないことを意味しますが、あなたは*A z* で新しいグループのリストを得ることができます。あなたは好みのものを購読し、要らないものは kill すれば良いのです。(*A k* で kill されたグループのリストを表示します。)

もしレベル変数で遊びたいのであれば、多少注意をしてまわる必要があります。いったんそれを設定したら、二度とそれに触らないでください。さらに言えば、自分で何をやっているかを正確に理解していない限り、一切触らないでください。

身近に関係する二つの変数は`gnus-level-default-subscribed` (ディフォルトは 3) と`gnus-level-default-unsubscribed` (ディフォルトは 6) です。これらは新しいグループが(非) 購読されたときのレベルです。もちろん、これら二つの変数の値は、意味のある正しい範囲でなくてはなりません。

`gnus-keep-same-level` が`nil` 以外であれば、移動コマンドのいくつかは同一(あるいはそれより小さい) レベルのグループのみの移動になります。特に、あるグループの最後の記事から次のグループに移るとき、次の同一(あるいはそれより小さい) レベルのグループに移動します。これは残りのグループを読むより先に、より重要なグループを読んでおきたいときには便利かもしれません。

もしこの値が`best` だったら、最も重要な(最もレベルの値が小さい) グループに移動します。

ディフォルトでは`gnus-group-default-list-level` と同じかそれより小さいレベルのグループが、グループバッファーに一覧表示されます。この変数は関数であることもできます。その場合、関数が呼ばれてその結果が値として使われます。

`gnus-group-list-inactive-groups` が`nil` 以外であれば、未読のグループにアクティブでないグループも一緒に表示します。この変数はディフォルトでは`t` です。もしこれが`nil` であれば、アクティブでないグループは表示されません。

`gnus-group-use-permanent-levels` が`nil` 以外であれば、いったん*g* や*l* コマンドの接頭引数にレベルを与えると、その後のすべてのコマンドにおいてそのレベルが「作用する」レベルになります。

Gnus は通常、`gnus-activate-level` かそれより小さいレベルのグループのみを起動します(つまりサーバーに問い合わせをする)。購読していないグループを起動したくなければ、この変数を例えば 5 に設定するとよいかもしれません。ディフォルトは 6 です。

3.7 グループのスコア

普通は重要なグループは高レベルにしておくでしょうけれども、この方法では少々制限がきついですよね。ひょっとしたら、グループをどれくらい頻繁に読むかによってグループバッファーを並べ替えたいなあ、なんて思いませんか? 理にかなってるでしょ?

「グループスコア」(*group score*) はそのためのものです。Gnus に以下で説明されている機構で、それぞれのグループに対してスコアを指定することができます。そしてグループバッファーをこのスコアを基に並べ替えることができます。あるいは、スコア順で並べ替えてその後レベルで並べ替えることもできます。(レベルとスコアをひとまとめにして、グループの「ランク」(*rank*) と呼びます。レベルが 4 でスコアが 1 のグループは、レベルが 5 でスコアが 300 のグループよりも高いランクとなります。(レベルの方が重要度が高く、スコアの方は重要度が低くなります。))

頻繁に読むグループに、めったに読まないグループよりも高いスコアを与えたいときは、`gnus-summary-exit-hook` フックに`gnus-summary-bubble-group` 関数を追加することができます。これでバブル並べ替えの実行結果が(並べ替えの後で) 得られるでしょう。概略モードを終了するたびにこの活動をさせたいのであれば、同じフックに`gnus-group-sort-groups-by-rank` または`gnus-group-sort-groups-by-score` を追加できますが、いくらか遅くなるでしょう。

3.8 グループへの印

もしいくつかのグループに対して何らかの命令を実行したい場合で、それらがグループバッファーに連続してある場合には、通常通り命令に対して数値接頭引数を与えるだけです。そうすればほとんどのグループ命令は、これらのグループに対してあなたの命令に従います。

しかしそれらのグループが順番に並んでいない場合においても、いくつかのグループに対して命令を実行することができます。単に始めにプロセス印でグループに印を付けておき、そして命令を実行するだけです。

`#`
`M m` 現在のグループのプロセス印をトグルします(gnus-group-mark-group)。gnus-process-mark-toggle がnil である場合は、現在のグループにプロセス印を付けます。

`M-#`
`M u` 現在のグループから、もしプロセス印があれば、削除します(gnus-group-unmark-group)。

`M U` すべてのグループからプロセス印を削除します(gnus-group-unmark-all-groups)。

`M w` 領域内のすべてのグループにプロセス印を付けます(gnus-group-mark-region)。

`M b` バッファー内のすべてのグループに印を付けます(gnus-group-mark-buffer)。

`M r` ある正規表現に合致するすべてのグループに印を付けます(gnus-group-mark-regexp)。

Section 10.1 [Process/Prefix], p. 271, も参照してください。

プロセス印が付けられているすべてのグループに対して何かの命令を実行したいときは、`M-&` (gnus-group-universal-argument) 命令を使うことができます。プロンプトから実行したい命令を入力します。

3.9 外部グループ

どうやってサーバーを購読するか(see Section 2.1 [Finding the News], p. 3) を考えるとき、最初に思い付くのはgnus-secondary-select-methods およびgnus-select-method に Emacs Lisp で定義を書くことでしょう。別のやり方は外部のサーバーとグループを使うことです。ここで「外部(Foreign)」は、選択方法(select methods) によるものではないことを意味します。すべての外部サーバーに関する設定と何を購読するかは~/.newsrc.eld ファイルだけに格納されます。

以下では、一般的な外部グループの作成、変更を行なうグループモードの命令をいくつか、および特別な目的のグループを簡単に作成する命令を紹介します。これらの命令はすべて、新規に作成したグループをポイント位置に挿入します—gnus-subscribe-newsgroup-method は参照されません。

これらグループを編集するコマンド群で行なった変更は~/.newsrc.eld (gnus-startup-file) に格納されます。代わりの手段として、変数gnus-parameters も用意されています(see Section 3.10 [Group Parameters], p. 23)。

`G m` 新しいグループを作成します(gnus-group-make-group)。Gnus はプロンプトを表示して、名前と方法と、場合によっては*address* の入力を求めてきます。

より簡単にNNTP グループを購読する方法については、Section 3.14 [Browse Foreign Server], p. 34, を参照してください。

G M 一時ニュースグループを作成します(gnus-group-read-ephemeral-group)。Gnus はプロンプトを表示して、名前、方法および*address* の入力を求めます。

G r 現在のグループの名前を、何か別のものに変更します(gnus-group-rename-group)。これはある種のグループ---主にメールグループに対してのみ有効です。このコマンドはバックエンドによっては非常に遅いことも有り得ます。

G c グループパラメーターをカスタマイズする(gnus-group-customize)。

G e 現在のグループの選択方法を修正するためのバッファーに移動します(gnus-group-edit-group-method)。

G p グループパラメーターを修正するためのバッファーに移動します(gnus-group-edit-group-parameters)。

G E グループ情報を修正するためのバッファーに移動します(gnus-group-edit-group)。

G d ディレクトリーグループを作成します(see Section 7.6.1 [Directory Groups], p. 202)。ディレクトリー名をプロンプトで入力します(gnus-group-make-directory-group)。

G h

Gnus ヘルプグループを作成します(gnus-group-make-help-group)。

G D 任意のディレクトリーをnneething バックエンドニュースグループであるかのように読み込みます(gnus-group-enter-directory)。See Section 7.6.2 [Anything Groups], p. 203.

G f 何らかのファイルをもとにグループを作成します(gnus-group-make-doc-group)。このコマンドに接頭引数を与えた場合、ファイル名とファイルタイプをプロンプトで入力します。現在サポートされているファイルタイプはmbox, babyl, digest, news, rnews, mmdf, forward, rfc934, rfc822-forward, mime-parts, standard-digest, slack-digest, clari-briefs, nsmail, outlook, oe-dbx およびmailman です。接頭引数なしでこのコマンドを実行すると、Gnus はファイルタイプを推測します。See Section 7.6.3 [Document Groups], p. 204.

G u gnus-useful-groups にあるグループの一つを作ります(gnus-group-make-useful-group)。

G w ウェブ検索結果をもとに一時的なグループを作成します(gnus-group-make-web-group)。このコマンドに接頭引数を与えると、一時的ではなく固定したグループを作成します。プロンプトで検索エンジンの種類(search engine type) と検索文字列を入力します。有効な検索エンジンの種類にはgoogle とdejanews があります。See Section 7.5.1 [Web Searches], p. 198.

もし、google 検索エンジンを用いる場合には、'shaving group:alt.sysadmin.recovery' のような合致する文字列を用いることによって、検索対象を特定のグループに限定することが可能です。

G R RSS feed に基づくグループを作ります(gnus-group-make-rss-group)。URL の入力を促されます。See Section 7.5.2 [RSS], p. 199.

G DEL　この関数は現在のグループを削除します(gnus-group-delete-group)。接頭引数が与えられると、この関数はそのグループ内の全記事を本当に削除し、グループ自身をこの世から強制的に抹殺してしまいます。接頭引数は、あなたが何をやろうとしているか、本当に自信があるときにのみ使ってください。まあ、このコマンドは(nntp グループのような) 読み出し専用グループには使えませんけれど。

G V　新しい、新鮮な、空のnnvirtual グループを作成します(gnus-group-make-empty-virtual)。See Section 7.7 [Virtual Groups], p. 209.

G v　現在のグループをnnvirtual グループに追加します(gnus-group-add-to-virtual)。これはプロセス印/接頭引数の習慣に従います。

さまざまな選択方法に関するさらなる情報はChapter 7 [Select Methods], p. 146, を参照してください。

もしgnus-activate-foreign-newsgroups が正の数であれば、Gnus は起動時に、この数かそれよりも小さいレベルの外部グループをすべてチェックします。これは特に違ったNNTPサーバーからたくさんのグループを購読している場合には、しばらく時間がかかるかもしれません。Section 3.6 [Group Levels], p. 19, も参照してください。gnus-activate-level も外部ニュースグループの活性化に影響を及ぼします。

以下のコマンドは一時的なグループを作ります。それらは Group バッファーからだけではなく、どの Gnus バッファーからも呼ぶことができます。

gnus-read-ephemeral-emacs-bug-group
: 一時的なグループで Emacs のバグリポートを購読します。Gnus はバグの番号(複数可) を尋ねます。ディフォルトは現在位置の番号です。URL の雛形はgnus-bug-group-download-format-alist で指定します。

gnus-read-ephemeral-debian-bug-group
: 一時的なグループで Debian のバグリポートを購読します。gnus-read-ephemeral-emacs-bug-group に似ています。

これらのコマンドのいくつかは、記事のボタンとしても便利です。See Section 4.18.6 [Article Buttons], p. 98.

例:

```
(require 'gnus-art)
(add-to-list
 'gnus-button-alist
 '("#\\([0-9]+\\)\\>" 1
   (string-match "\\<emacs\\>" (or gnus-newsgroup-name ""))
   gnus-read-ephemeral-emacs-bug-group 1))
```

3.10 グループパラメーター

グループパラメーターは、ある特定のグループに固有な情報を保持します。

グループパラメーターの修正には*G p* か*G c* 命令を使ってください(*G p* は Lisp ベースの、*G c* は Custom ふうのインターフェースを提供します)。トピックパラメーターについて読んでみることも面白いでしょう(see Section 3.16.5 [Topic Parameters], p. 40)。加えて、gnus-parameters 変数を介してグループパラメーターを設定することもできます。下記参照してください。

以下はグループパラメーターリストの例です:

```
((to-address . "ding@gnus.org")
 (auto-expire . t))
```

それぞれの要素は『点対』(dotted pair)---つまり点(dot) の前に鍵、点の後ろに値があるもの、で構成されます。すべてのパラメーターはこの形式を取りますが、例外 としてローカル変数の指定は点対ではなく通常のリスト(訳注: 後述の(*variable form*) の項を参照)になります。

いくつかのパラメーターは対応するカスタマイズ可能な変数を持っています。それらは正規表現と値の連想リストです。

以下は利用可能なグループパラメーターです:

`to-address`
: フォローアップとニュースへの投稿をするときに使用されるアドレス。

```
(to-address . "some@where.com")
```

これは主に、閉じたメーリングリストを表わすメールグループにおいて便利なものです---すなわちメーリングリストに投稿する人はすべてそれを購読しているはず、というメーリングリストのことです。このパラメーターを使用すると、メールはそのメーリングリストにしか投稿されないことが保証されるので、参加者はあなたのフォローアップ記事を二通受け取ることはありません。

`to-address` を指定すると、そのグループが外部グループであるかどうかに関わらず有効になります。例えば‘`fa.4ad-l`’ というグループがサーバー上にあったとしましょう。これは本当のニュースグループですが、サーバーはメールニュースゲートウェイを通して記事を受け付けます。つまりこのグループに対して直接投稿することは不可能で、代わりにそのメーリングリストにメールを送信しなければなりません。

`gnus-parameter-to-address-alist` も参照してください。

`to-list`
: そのグループで`a` を押したときに使用されるアドレス。

```
(to-list . "some@where.com")
```

これはフォローアップをしたときは完全に無視されます---例外はそれがニュースグループを表わしているときは、*f* を押したときにメールグループのルールが適用されるということです。

もし`a` コマンドをメールグループで実行したときに、`to-list` グループパラメーターも`to-address` もグループパラメーターも無ければ、`to-list` グループパラメーターは、`gnus-add-to-list` が`t` に設定されていればメッセージ送信時に自動的に付加されます。

もしこのグループパラメーターが設定されていると、概略バッファーに入ったときに`gnus-mailing-list-mode` が有効になります。

`subscribed`
: もしこのパラメーターが`t` に設定されていると、Gnus はあなたがこのグループを to-address と to-list パラメーターのアドレスで購読しているメーリングリストであると解釈します。この情報を Gnus に与えることは、あなたがそれらのメーリングリストに投稿するときに正しい Mail-Followup-To ヘッダーを生成するための(ほんの) 第一歩です。二歩目は`.gnus.el` に以下を入れることです。

```
(setq message-subscribed-address-functions
```

'(gnus-find-subscribed-addresses))

利用できる MFT 対応機能を完全に扱うには、ここ(see Section “メーリングリスト” in *The Message Manual*) を見てください。

`visible` グループパラメーターのリスト中に`(visible . t)` という要素があれば、そのグループはグループバッファーにおいて、未読記事があるかどうかに関わらず、常に表示されます。

このパラメーターを`gnus-parameters` を介して設定することはできませんが、代わりに`gnus-permanently-visible-groups` を使えば良いでしょう。

`broken-reply-to`
`(broken-reply-to . t)` という要素があれば、そのグループでは`Reply-To` は無視され、`reply-to` が`gnus-boring-article-headers` の部分であれば、ヘッダーが隠されるという意味です。これはある listserv によるメーリングリストを購読していて、それが`Reply-To` 欄を listserv 自身に返すように付けられている場合に有効でしょう。これはおかしな振る舞いです。だからこれが要るんです!

`to-group` `(to-group . "some.group.name")` という要素は、そのグループへの投稿はすべて`some.group.name` に送られる、という意味です。

`newsgroup`
グループパラメーターリストに`(newsgroup . t)` があれば、Gnus はすべての応答をニュース記事に対する応答であるかのように扱います。これは実際にはニュースグループのミラーであるメールグループに対して有効です。

`gcc-self` グループパラメーターリストにもし`(gcc-self . t)` があると、新しく作成するメッセージは現在のグループに`gcc` されます。もし`(gcc-self . none)` があれば、`Gcc:` 欄は生成されません。もし`(gcc-self . "group")` があると、この文字列はそのまま`Gcc:` ヘッダーとして挿入されます。それはグループ名でなければなりません。

`gcc-self` の値は文字列と`t` のリストにすることも可能です。例えば`(gcc-self "group1" "group2" t)` は、新しく作成するメッセージを`"group1"`、`"group2"`、および現在のグループに`gcc` するということです。

`gcc-self` パラメーターは後に説明するどんなディフォルトの`Gcc` の規則よりも(再送するメッセージのための例外を除いて) 優先されます(see Section 6.5 [Archived Messages], p. 137)。

警告: `nntp` (またはその種の) グループのパラメーターリストに`(gcc-self . t)` を加えることに効力はありません。`nntp` サーバーは記事を受け入れません。

`auto-expire`
グループパラメーターに`(auto-expire . t)` のような要素があれば、すべての既読記事は期限切れ消去されるように印を付けられます。他の方法は、See Section 7.4.9 [Expiring Mail], p. 181.

`gnus-auto-expirable-newsgroups` も参照してください。

`total-expire`
グループパラメーターに`(total-expire . t)` のような要素があれば、既読記事は、期限切れ消去の印が付いていなくてもすべて期限切れ消去処理を施されます。注意して使用してください。未読記事、印付き記事、保留記事は期限切れ消去されません。

gnus-total-expirable-newsgroups も参照してください。

expiry-wait
: グループパラメーターに(expiry-wait . 10) のような要素があれば、この値は記事を期限切れ消去するときにnnmail-expiry-wait やnnmail-expiry-wait-function の設定(see Section 7.4.9 [Expiring Mail], p. 181) よりも優先されます。この値は期限切れ消去の日数(整数である必要はない) かもしくはneverかimmediate のシンボルを指定できます。

expiry-target
: 期限切れ消去されるメッセージの果てる場所。このパラメーターはnnmail-expiry-target よりも優先されます。

score-file
: (score-file . "file") のような要素は、file を現在のグループに適用されるスコアファイルにします。すべての適用されるスコア・エントリーはこのファイルに入ります。

adapt-file
: (adapt-file . "file") のような要素は、file を現在のグループの適応ファイルにします。すべての適応スコア・エントリーはこのファイルに入ります。

admin-address
: メーリングリストから脱会するときは、脱会通知メールをそのメーリングリスト自身に送信してはいけません。代わりに管理用アドレスにメッセージを送信します。このパラメーターにはどこか都合の良いな管理用アドレスを書いておくことができます。

display
: (display . MODE) のような要素は、グループに入るときにどの記事を表示するかを指定します。有効な値は、

  all
  : 未読、既読記事の両方をすべて表示します。

  an integer
  : そのグループの最後の*integer* 個の記事を表示します。これは*C-u integer* でそのグループに入るのと同じです。

  default
  : ディフォルトの記事を表示します。これは通常は未読記事と可視記事です。

  配列
  : 述語を満足するように記事を表示します。
    いくつか例を挙げます:

    [unread] 未読の記事だけを表示します。

    [not expire]
    : 期限切れ消去可能な記事以外のすべてを表示します。

    [and (not reply) (not expire)]
    : 期限切れ消去可能とすでに返信した記事以外のすべてを表示します。

    利用できる演算子はnot, and およびor です。述語はtick, unsend, undownload, unread, dormant, expire, reply, killed, bookmark, score, save, cache, forward およびunseen を含みます。

display パラメーターは、概略バッファーを指定した一部の組だけに制限するように働きます。制限を外すのは/ w コマンドでできます(see Section 4.8 [Limiting], p. 67)。

comment (comment . "This is a comment") のような要素は、そのグループに対する任意のコメントです。グループ行に表示することができます(see Section 3.1.1 [Group Line Specification], p. 12)。

charset (charset . iso-8859-1) のような要素は、iso-8859-1 をディフォルトの文字セットにします。すなわち、文字セットを指定しないすべての記事に、その文字セットが使われます。

gnus-group-charset-alist も見てください。

ignored-charsets
(ignored-charsets x-unknown iso-8859-1) のような要素は、iso-8859-1 とx-unknown を無視します。すなわち、記事のデコードにディフォルトの文字セットが使われます。

gnus-group-ignored-charsets-alist も見てください。

posting-style
このグループの追加の投稿様式をここに保存することができます(see Section 6.6 [Posting Styles], p. 140)。書式はgnus-posting-styles 連想リストと同じですが、ここにはグループ名に合致する正規表現はありません(当然です)。このグループの様式の要素はgnus-posting-styles で見つかったものよりも優先されます。

例えば、このグループのみ、かっこいい名前と署名にしたいなら、gnus-posting-styles をいじらずに、このようなものをグループパラメーターに入れることができます:

```
(posting-style
  (name "Funky Name")
  ("X-Message-SMTP-Method" "smtp smtp.example.org 587")
  ("X-My-Header" "Funky Value")
  (signature "Funky Signature"))
```

グループバッファーを整理するためにトピック(see Section 3.16 [Group Topics], p. 35) を使っている場合は、トピックパラメーターでも投稿様式を設定することができます。トピックパラメーターにある投稿様式は、そのトピックのすべてのグループに適用されます。もっと正確に言うと、あるグループのための投稿様式の設定は、そのグループおよびそれが属するすべてのトピックのパラメーターにあるすべての投稿様式の設定を、階層的に合併することによって生成されます。

post-method
もしこれが設定されていると、メッセージを送信するための選択方法としてgnus-post-method の代わりに使われます。

mail-source
これが設定されていて、かつmail-sources の設定がgroup グループ・メールソース(see Section 7.4.4 [Mail Sources], p. 166) を含んでいるならば、その値がこのグループのメールソースになります。

`banner` `(banner . regex)` のような項目は、記事のすべての場所で正規表現*regex* に合致するものを削除します。*regex* の代わりにシンボル`signature` (最後の署名を削除) や連想リスト`gnus-article-banner-alist` の各要素を使うこともできます。

`sieve` このパラメーターは、入ってきたメールがこのグループに置くに値するかどうかを調べる Sieve (ふるい) テストを持ちます。このグループパラメーターを元に‘`fileinto "group.name";`’というテスト条件を本体に持つ、Sieve の‘`IF`’制御構造体が作られます。

例えば、もし‘`INBOX.list.sieve`’グループが`(sieve address "sender" "sieve-admin@extundo.com")` というグループパラメーターを持っていたならば、グループパラメーターを Sieve スクリプトに変換する(see Section 3.18.5 [Sieve Commands], p. 45) ときに、以下の Sieve コードが作られます:

```
if address "sender" "sieve-admin@extundo.com" {
        fileinto "INBOX.list.sieve";
}
```

さらに次のような要領で正規表現の拡張も使うことができます:

```
(sieve header :regex "list-id" "<c++std-\\1.accu.org>")
```

複数の電子メールアドレスのためのテストを生成するには、`(sieve address "sender" ("name@one.org" "else@two.org"))` のようなグループパラメーターを使ってください。Sieve スクリプト(see Section 3.18.5 [Sieve Commands], p. 45) を生成すると、以下のような Sieve コードが作られます:

```
if address "sender" ["name@one.org", "else@two.org"] {
        fileinto "INBOX.list.sieve";
}
```

Sieve パラメーターに関連する重要なコマンドと変数については、Section 3.18.5 [Sieve Commands], p. 45, を参照してください。

Sieve 言語は RFC 3028 で述べられています(see *Emacs Sieve*)。

`match-list`
もしこのパラメーターが`t` に設定され、かつ`nnmail-split-method` が`gnus-group-split` に設定されていたら、Gnus は`list` 分割の省略形と対比して`to-address`、`to-list`、`extra-aliases` および`split-regexp` が一致するかどうかを調べます。RFC2919 の`List-ID` に準拠するため、分割の正規表現はメールアドレス中の`@` またはドット`.` に合致するように変更されます。

メーリングリストのヘッダーに合致する正規表現については`nnmail-split-abbrev-alist` を参照してください。

グループパラメーターに基づいて自動的に行なわれる分割についてはSection 7.4.7 [Group Mail Splitting], p. 178, を参照してください。

`(agent parameters)`
エージェントを使うようにしてあると、個々のグループでエージェントの振る舞いを制御するどのパラメーターも設定することができます。エージェントパラメーターについてはSection 7.9.2.1 [Category Syntax], p. 218, を参照してください。たいていの利用者は、設定に要する苦労を最小限にするために、エージェントカテゴリーかグループトピックのどちらかでエージェントパラメーターを設定することを選ぶでしょう。

(*variable form*)
: グループに入るときに、そのグループローカルの変数を設定するグループパラメーターを使用することができます。'news.answers' においてスレッド表示を行ないたくないときは、そのグループにグループパラメーターに(gnus-show-threads nil) と書けます。gnus-show-threads は、その概略バッファーの中のローカル変数になり、form のnil はそこでeval (評価) されます。

この機能は*variable* が変数として存在する場合に限って、それを概略バッファーでローカルに設定することに注意してください。さもなければ form の評価だけが行なわれるでしょう。したがって、もし form を評価した結果を変数に設定する必要があるのならば、defvar などを使って、前もってその変数を定義しておかなければなりません。

でも、いくつかの変数は記事バッファーか(返信、フォロー、あるいは新規に作られたメッセージの) メッセージバッファーで評価されます。代わりに、問題の変数をgnus-newsgroup-variables に加えることが助けになるかもしれません。したがって、グループパラメーターを介してmessage-from-style を設定したいならば、~/.gnus.el ファイルのどこか他のところに、次の述語が必要になるかもしれません:

```
(add-to-list 'gnus-newsgroup-variables 'message-from-style)
```

この機能の用途の一つは、記事の表題欄からメーリングリストの標識タグをはぎ取ることです。もしニュースグループ

```
nntp+news.gnus.org:gmane.text.docbook.apps
```

が、すべての記事の表題に'DOC-BOOK-APPS:' というタグを持っているならば、そのグループのグループパラメーターに(gnus-list-identifiers "DOCBOOK-APPS:") を入れることによって、そのグループの概略バッファーに表示される記事の表題からタグをはぎ取ることができます。

これはもし必要であれば、グループ毎のフック関数としても使用できます。もしあるグループに入ったときにビープ音を鳴らしたければ、そのグループのパラメーターに(dummy-variable (ding)) みたいなものを書いておくこともできます。もしdummy-variable という変数が存在していれば(上記参照)、それに(無意味な) (ding) の評価結果が設定されます。

あるいは、*variable* はそのグループに対してローカルになるので、この様式は一時的にフックを変更するために使うことができます。例えば、以下のものがグループパラメーターに追加されると、

```
(gnus-summary-prepared-hook
  (lambda nil (keymap-local-set "d" (local-key-binding "n"))))
```

そのグループに入ったときに*d* キーは記事に期限切れ消去の印を付けないようになります。

グループパラメーターはgnus-parameters 変数を介在して設定することもできます。でもいくつかのパラメーター、例えばvisible は効力を発揮しません(その場合、代替としてgnus-permanently-visible-groups を使うことができます)。例です:

```
(setq gnus-parameters
      '(("mail\\..*"
         (gnus-show-threads nil)
```

```
 (gnus-use-scoring nil)
 (gnus-summary-line-format
  "%U%R%z%I%(%[%d:%ub%-23,23f%]%) %s\n")
 (gcc-self . t)
 (display . all))

("^nnimap:\\(foo.bar\\)$"
 (to-group . "\\1"))

("mail\\.me"
 (gnus-use-scoring  t))

("list\\..*"
 (total-expire . t)
 (broken-reply-to . t))))
```

グループ名が合致するすべての項が使われますが、最後の設定が「勝ち」ます。そういうわけですから、もしそのグループ名に合致する二つの項があって、例えばどちらも`display`を設定していると、最後の設定が最初の設定を覆します。

グループ名が合致する最初の項が使われます。

文字列のパラメーターは`to-group` の例が示すように正規表現による置き換えを受けることがあります。

グループ名と`gnus-parameters` で指定されたこれらの正規表現の一つを比較するときに大文字と小文字を区別するかどうかは、ディフォルトではその比較を行なう時点での`case-fold-search` の値に依存します。一般的に`case-fold-search` の値は`t` で、それは例えば`("INBOX\\.FOO" (total-expire . t))` という要素が、‘`INBOX.FOO`’ グループと‘`INBOX.foo`’ グループの両方に適用されることを意味します。これらの正規表現が常に大文字と小文字を区別するようにしたい場合は、`gnus-parameters-case-fold-search` 変数の値を`nil` に設定してください。あるいは、それらが常に大文字と小文字を区別しないようにしたいなら、それを`t` に設定してください。

`gnus-parameters` を介することによって、グループによって異なる並べ替えを定義することができます。これは、NNTP グループでは最新のニュースが先頭になるように日付で、RSS グループでは表題で、それぞれ並べ替えを行なう例です。この例の最初のグループは、news.gmane.io から取得する Debian のデイリーニュースです。RSS グループは RSS フィードで配信されている Debian のウィークリーニュース`https://packages.debian.org/unstable/newpkg_main.en.rdf` に対応します。See Section 7.5.2 [RSS], p. 199.

```
(setq
 gnus-parameters
 '(("nntp.*gmane\\.debian\\.user\\.news"
    (gnus-show-threads nil)
    (gnus-article-sort-functions '((not gnus-article-sort-by-date)))
    (gnus-use-adaptive-scoring nil)
    (gnus-use-scoring nil))
   ("nnrss.*debian"
    (gnus-show-threads nil)
    (gnus-article-sort-functions 'gnus-article-sort-by-subject)
```

```
(gnus-use-adaptive-scoring nil)
(gnus-use-scoring t)
(gnus-score-find-score-files-function 'gnus-score-find-single)
(gnus-summary-line-format "%U%R%z%d %I%(%[ %s %]%)\n"))))
```

3.11 グループの一覧表示

これらのコマンドは、利用できるグループをいろいろに切り分けて表示します。

l
A s 未読記事を持つすべてのグループを表示します(gnus-group-list-groups)。数値接頭引数を使うと、このコマンドは引数の数かそれよりも小さいレベルのグループのみを表示します。ディフォルトでは、これはレベル 5 (つまりgnus-group-default-list-level) かそれより小さいレベル(すなわち購読しているグループのみ) を表示します。

L
A u 未読記事のあるなしに関わらず、すべてのグループを表示します(gnus-group-list-all-groups)。数値接頭引数を使用すると、このコマンドは引数の数かそれよりも小さいレベルのグループのみを表示します。ディフォルトでは 7 かそれよりも小さいレベルのグループ(すなわち購読、非購読のグループのみ) が表示されます。

A l 未読記事があるグループのうち、指定したレベルのものだけを表示します(gnus-group-list-level)。接頭引数を与えると、未読記事の無いグループも含めて表示します。

A k kill されたグループをすべて表示します(gnus-group-list-killed)。接頭引数を与えると、購読または非購読のどちらにもなっていないすべての利用可能なグループを表示します。これはサーバーからアクティブファイルを読むことになるでしょう。

A z すべてのゾンビグループを表示します(gnus-group-list-zombies)。

A m 正規表現に合致する名前を持つグループで、未読記事のある購読グループをすべて表示します(gnus-group-list-matching)。

A M 正規表現に合致するグループを表示します(gnus-group-list-all-matching)。

A A 今接続しているサーバーのアクティブファイルにあるグループを、本当に全部表示します(gnus-group-list-active)。これはしばらく時間がかかることも有り得ます。たぶん*A M* を実行して、合致させたい部分を‘.’としてすべての合致するリストを表示させた方が良いでしょう。また、このコマンドは(まだ) 存在しないグループも表示するかもしれません---これは kill されたグループであるかのように表示されます。出力は多少割り引いて受け取ってね。

A a 正規表現に合致する名前を持つグループをすべて表示します(gnus-group-apropos)。

A d 正規表現に合致する名前か説明文を持つグループをすべて表示します(gnus-group-description-apropos)。

A c キャッシュ記事を持つグループをすべて表示します(gnus-group-list-cached)。

A ? 保留記事を持つグループをすべて表示します(gnus-group-list-dormant)。

A ! 可視記事(ticked articles) があるグループをすべて表示します(gnus-group-list-ticked)。

A / さらに現在の選択された範囲に限定したグループを表示します(gnus-group-list-limit)。最初に*A ?* で保留記事があるグループに制限してあるなら、*A / c* でさらにキャッシュされた記事があるグループ、つまり保留記事があって、かつキャッシュされた記事もあるグループに制限することができます。

A f 現在の選択されたグループを書き出します(gnus-group-list-flush)。

A p 現在の選択されたグループを加えたグループを表示します(gnus-group-list-plus)。

gnus-permanently-visible-groups 正規表現に合致するグループは、未読記事があるかないかに関わらず常に表示されます。あるいはグループパラメーターにおいてvisible 要素を追加することでも同様の効果を得ることができます。

印付きの記事のみを持つグループは通常グループバッファーに表示されます。もしgnus-list-groups-with-ticked-articles がnil であれば、そのグループは完全に空のグループであるかのように扱われます。ディフォルトはt です。

3.12 グループの並べ替え

C-c C-s (gnus-group-sort-groups) 命令は、グループバッファーをgnus-group-sort-function 変数で与えられる関数に従って並べ替えます。利用可能な並べ替え関数(sorting function) には以下のものがあります:

gnus-group-sort-by-alphabet
グループ名でアルファベット順に並べ替えます。これがディフォルトです。

gnus-group-sort-by-real-name
グループを本当の(前に何も付いていない) グループ名でアルファベット順に並べ変えます。

gnus-group-sort-by-level
グループレベルで並べ替えます。

gnus-group-sort-by-score
グループのスコアで並べ替えます。See Section 3.7 [Group Score], p. 20.

gnus-group-sort-by-rank
グループのスコアで並べ替え、次にグループレベルで並べ替えます。レベルとスコアは、ひとまとめにして「ランク」と呼ばれます。See Section 3.7 [Group Score], p. 20.

gnus-group-sort-by-unread
未読記事の数で並べ替えます。

gnus-group-sort-by-method
選択方法のアルファベット順で並べ替えます。

gnus-group-sort-by-server
サーバー名のアルファベット順で並べ替えます。

gnus-group-sort-function は並べ替え関数のリストであっても構いません。この場合、もっとも重要な並べ替えの鍵を持つ関数は最後でなくてはなりません。

ある種の並べ替え用には、直接並べ替える命令もいくつかあります。

G S a グループバッファーをグループ名のアルファベット順で並べ替えます(gnus-group-sort-groups-by-alphabet)。

G S u グループバッファーを未読記事の数で並べ替えます(gnus-group-sort-groups-by-unread)。

G S l グループバッファーをグループレベルで並べ替えます(gnus-group-sort-groups-by-level)。

G S v グループバッファーをグループのスコアで並べ替えます(gnus-group-sort-groups-by-score)。See Section 3.7 [Group Score], p. 20.

G S r グループバッファーをグループのランクで並べ替えます(gnus-group-sort-groups-by-rank)。See Section 3.7 [Group Score], p. 20.

G S m グループバッファーをバックエンドの名前でアルファベット順に並べ替えます(gnus-group-sort-groups-by-method)。

G S n グループバッファーを本当の(前に何も付いていない) グループ名でアルファベット順に並べ替えます(gnus-group-sort-groups-by-real-name)。

以下のすべての命令はプロセス/接頭引数の習慣に従います(see Section 10.1 [Process/Prefix], p. 271)。

シンボル接頭引数(see Section 10.3 [Symbolic Prefixes], p. 272) が与えられたときは、これらすべての命令は逆順で並び換えます。

また、グループの一部を並べ替えることもできます。

G P a グループをグループ名のアルファベット順で並べ替えます(gnus-group-sort-selected-groups-by-alphabet)。

G P u グループを未読記事の数で並べ替えます(gnus-group-sort-selected-groups-by-unread)。

G P l グループをグループレベルで並べ替えます(gnus-group-sort-selected-groups-by-level)。

G P v グループをグループのスコアで並べ替えます(gnus-group-sort-selected-groups-by-score)。See Section 3.7 [Group Score], p. 20.

G P r グループをグループのランクで並べ替えます(gnus-group-sort-selected-groups-by-rank)。See Section 3.7 [Group Score], p. 20.

G P m グループをバックエンドの名前でアルファベット順に並べ替えます(gnus-group-sort-selected-groups-by-method)。

G P n グループを本当の(前に何も付いていない) グループ名でアルファベット順に並べ替えます(gnus-group-sort-selected-groups-by-real-name)。

G P s グループをgnus-group-sort-function に従って並べ替えます。

最後に、*C-k* と*C-y* を使って、手動でグループをあちこちに移動できることもお忘れなく。

3.13 グループの管理

b 不正なグループを見つけて、削除します(gnus-group-check-bogus-groups)。

F 新しいグループを見つけて、それらを処理します(gnus-group-find-new-groups)。一回の*C-u* の後で押されると、サーバーに新しいグループを尋ねるためにask-server の方法を使います。二回の*C-u* の後で押されると、サーバーに新しいグループを尋ねるために最も完全であると思われる方法を用い、新しいグループをゾンビとして購読します。

C-c C-x 現在のグループの期限切れ消去可能な記事に対して(もしあれば) すべて期限切れ消去の処理を行ないます(gnus-group-expire-articles)。これは、そのグループにしばらく存在していた期限切れ消去可能なすべての記事を消去するということです。(see Section 7.4.9 [Expiring Mail], p. 181)。

C-c C-M-x すべてのグループのすべての期限切れ消去可能な記事に対して、期限切れ消去の処理を行ないます。(gnus-group-expire-all-groups)。

3.14 外部サーバーの閲覧

B 選択方法とサーバー名を聞かれます。Gnus はこのサーバーに接続し、そこにあるグループを閲覧しようとします(gnus-group-browse-foreign-server)。

利用可能なグループのリストを持った新しいバッファーが現れます。このバッファーはgnus-browse-mode を使用します。このバッファーは通常のグループバッファーにちょっと(というか、とっても) 似ています。

以下が閲覧モード(browse mode) で使用できるキー操作のリストです:

n 次のグループに移動します(gnus-group-next-group)。

p 一つ前のグループに移動します(gnus-group-prev-group)。

SPC 現在のグループに入り、最初の記事を表示します(gnus-browse-read-group)。

RET 現在のグループに入ります(gnus-browse-select-group)。

u 現在のグループを購読する/しないを切り替えます(gnus-browse-toggle-subscription)。新しいグループをグループバッファーに編入する方法を、変数gnus-browse-subscribe-newsgroup-method を使って制御することができます。利用できるオプションについては: See Section 2.4.2 [Subscription Methods], p. 6.

l
q 閲覧モード(browse mode) を終了します(gnus-browse-exit)。

d 現在のグループを購読にします(gnus-browse-describe-group)。

? 閲覧モード(browse mode) を簡単に説明します(まあ、大して説明することもないんですけれどね) (gnus-browse-describe-briefly)。

3.15 Gnus の終了

そう、Gnus は最後(サイコー) です(訳注: く、苦しい。原文は“Yes, Gnus is ex(c)iting.”)。

z Gnus を中断します(gnus-group-suspend)。これは Gnus を実際には終了させず、グループバッファー以外のすべてのバッファーを消すだけです。僕はこれのうれしさがよくわかんないんだけど、誰か分かる人います?

q Gnus を終了します(gnus-group-exit)。

Q .newsrc ファイルを保存せずに Gnus を終了します(gnus-group-quit)。ドリブルファイルは保存されますけれど(see Section 2.7 [Auto Save], p. 9)。

Gnus を中断するときはgnus-suspend-gnus-hook が呼び出されます。Gnus を終了するときはgnus-exit-gnus-hook が呼び出され、さらに Gnus を終了するときの最後としてgnus-after-exiting-gnus-hook が呼び出されます。

3.16 トピック

もしあなたがたーくさんのグループを読んでいるのであれば、グループをトピック毎に階層分けできると便利でしょう。Emacs のグループをこっちへ、セックスのグループをあっちへ、で、残りを(え? グループが二つくらいしかないの?) 邪魔にならないようにその他のセクションに入れましょう。あるいは Emacs セックスのグループを Emacs グループ、セックスグループのどちらかの副トピックとすることさえもできます---あるいは両方に! すんごいでしょう!

これが例です:

```
Gnus
  Emacs -- こいつはすげーぜ!
     3: comp.emacs
     2: alt.religion.emacs
    えっちな Emacs
     452: alt.sex.emacs
       0: comp.talk.emacs.recovery
    その他
     8: comp.binaries.fractals
    13: comp.sources.unix
```

この素晴らしい 機能を使うには、gnus-topic マイナーモードを(何と!) 単にスイッチオンするだけ---グループバッファーで、*t* を押してください(これはトグルコマンドです)。

さあやってみよう。とにかく試してみて。君が戻ってくるまで、僕はここで待ってるからさ。ララ、タララン...、いい曲だね、これ...ラ、ラ、ラ...え? 戻ってきた? よし、じゃ次は*l* を押してみて。ほら。これですべてのグループが‘misc’ の下に表示されました。興奮してクラクラしてこない? アツくって、いまいましいくらいでしょ?

これをずっと有効にしたければ、グループモードのフックにこのマイナーモードを追加してください。以下の行を~/.gnus.el ファイルに入れて、ね。

```
(add-hook 'gnus-group-mode-hook 'gnus-topic-mode)
```

3.16.1 トピック命令

トピックマイナーモードが有効であるときは、*T* サブマップが新しく利用できるようになります。さらに標準キーの中でも、定義がちょっと変わるものが少しあります。

だいたいにおいて、次のようなトピックの操作が可能です。まず第一に、あなたはトピックを作ることを望むでしょう。第二に、あなたはグループをトピックに入れて、それらをあなたの好みの順序になるまで、あちこちに移動することを望むでしょう。第三に行なう操作は、それらの一切合切を表示したり隠すことでしょう。他のグループの概要を見やすくするために、あなたは副トピックやグループによっては、トピックが隠れるようにする必要があるかもしれませんね。

ここには、あなたの好むやり方でトピックを設定するために必要になりそうな、基本的なキーのリストがあります。

T n 新しいトピック名の入力を促し、それを作成します(gnus-topic-create-topic)。

T TAB
TAB 現在のトピックの「字下げ」を行ない、その前のトピックの副トピックにします(gnus-topic-indent)。接頭引数を与えると、反対にそのトピックの字下げを回復(un-indent) します。

M-TAB 現在のトピックの「字下げ回復」(un-indent) を行ない、それが現在の親の親の副トピックになるようにします(gnus-topic-unindent)。

以下の二つのキーは、グループとトピックをあちこち移動するために使われます。それらは、よく知られているカット＆ペーストのように動作します。*C-k* はカット、*C-y* はペーストです。もちろん、Emacs ではカット＆ペーストではなくて kill & yank という用語を使いますが。

C-k グループあるいはトピックを kill します(gnus-topic-kill-group)。トピック内にあったグループもすべて、トピックと一緒に削除されます。

C-y 直前の kill されたグループあるいはトピックを yank します(gnus-topic-yank-group)。すべてのトピックは、すべてのグループの前に yank されることに気を付けてください。

ですから、あるトピックをトピックのリストの先頭に移動するには、単にそこで*C-k* を叩きます。これはカット＆ペーストのカットに相当します。そうしたらカーソルをバッファーの先頭(“Gnus”トピックの真下) に移動して、*C-y* を叩いてください。これはカット＆ペーストのペーストに相当します。なあんだ、簡単じゃん。

C-k と*C-y* はトピックと同様にグループにも使えます。すなわち、あなたはグループと同じようにトピックの移動もできるのです。

あなたの望みのままにトピックを使えるようにした後で、あなたはトピックを隠したり再び見えるようにしようと思うでしょう。そのために以下のキーを用意しています。

RET
SPC グループを選択するか、あるいはトピックを折りたたみます(gnus-topic-select-group)。グループの上でこのコマンドを実行すると、通常通りそのグループに入ります。トピック行の上で行なうと、そのトピックは(すでに表示されているときは) 折りたたまれるか、(すでに折りたたまれているときは) 展開されます。つまりトピックに対してはこれはトグルコマンドです。さらに、数値の接頭引数を与えると、そのレベル(とそれよりも小さいレベル) のグループが表示されます。

さてお次は、他のコマンドのリストです。順序には特に意味はありません。

T m 現在のグループを、どこか他のトピックに移動させます(gnus-topic-move-group)。このコマンドはプロセス印/接頭引数の習慣に従います(see Section 10.1 [Process/Prefix], p. 271)。

T j トピックにジャンプします(gnus-topic-jump-to-topic)。

T c 現在のグループを、どこか他のトピックにコピーします(gnus-topic-copy-group)。このコマンドはプロセス印/接頭引数の習慣に従います(see Section 10.1 [Process/Prefix], p. 271)。

T h 現在のトピックを隠します。接頭引数が与えらると、そのトピックを永久に隠します。

T s 現在のトピックを表示します。接頭引数が与えられると、そのトピックを永久に表示します。

T D グループを現在のトピックから削除します(gnus-topic-remove-group)。この命令は主にいくつかのトピックに同じグループがあって、それをトピックの一つから取り除きたいときに役立ちます。あなたはグループをすべてのトピックから取り除くかもしれませんが、その場合は、Gnus はあなたが次回に Gnus を起動したときにそれをルートトピックに付け加えます。実際のところ、すべての新しいグループ(もちろん、それはどのトピックにも属していません) はルートトピックに現われます。

この命令はプロセス印/接頭引数の習慣に従います(see Section 10.1 [Process/Prefix], p. 271)。

T M 正規表現に合致するすべてのグループを、あるトピックに移動させます(gnus-topic-move-matching)。

T C 正規表現に合致するすべてのグループを、あるトピックにコピーします(gnus-topic-copy-matching)。

T H 空のトピックの表示・非表示をトグルします(gnus-topic-toggle-display-empty-topics)。

T # 現在のトピックにあるグループすべてのプロセス印をトグルします(gnus-topic-mark-topic)。接頭引数を与えなければ、このコマンドは副トピックに対して再帰的に働きます。
gnus-process-mark-toggle がnil である場合は、現在のトピックにプロセス印を付けます。

T M-# 現在のトピックにあるすべてのグループからプロセス印を消します(gnus-topic-unmark-topic)。接頭引数を与えなければ、このコマンドは副トピックに対して再帰的に働きます。

C-c C-x (もしあれば) 現在のグループかトピックかのすべての期限切れ消去可能記事を期限切れ消去します(gnus-topic-expire-articles)。(see Section 7.4.9 [Expiring Mail], p. 181)。

T r トピックの名前を変更します(gnus-topic-rename)。

T DEL 空のトピックを削除します(gnus-topic-delete)。

A T Gnus が知っているトピック化されたグループをすべて表示します(gnus-topic-list-active)。

T M-n 次のトピックに移動します(gnus-topic-goto-next-topic)。

T M-p 前のトピックに移動します(gnus-topic-goto-previous-topic)。

G p トピックパラメーターを修正します(gnus-topic-edit-parameters)。See Section 3.16.5 [Topic Parameters], p. 40.

3.16.2 トピック変数

前の章では、どのトピックを表示するかを Gnus に言う方法を説明しました。この章では、それぞれのトピックの何を表示するかを Gnus に言う方法を説明します。

トピック行それ自体は、gnus-topic-line-format 変数の値に従って作成されます(see Section 10.4 [Formatting Variables], p. 272)。有効な要素は、

'i' 字下げ。

'n' トピック名。

'v' 見えるかどうか。

'l' レベル。

'g' トピック中のグループの数。

'G' トピックとすべての副トピックにあるグループの数。

'a' トピック中の未読記事の数。

'A' トピックとすべての副トピックの未読記事の数。

各副トピック(と副トピック内のグループ) は、トピックレベル数のgnus-topic-indent-level 倍の空白分の字下げが行なわれます。ディフォルトは 2 です。

gnus-topic-mode-hook はトピックマイナーモードバッファーで呼び出されます。

gnus-topic-display-empty-topics はトピックの中に未読記事が無い場合でもそのトピックを表示するようにします。ディフォルトはt です。

変数gnus-topic-display-predicate に、トピックを表示するかどうかを指定する関数を設定することができます。関数は 1 つのパラメーター(トピックの名前) で呼び出され、トピックを表示する場合はnil ではない値を返さなければなりません。

例えば、勤務時間外に仕事があることを思い出したくない場合は、仕事に関連するすべてのグループを'"Work"' というトピックにまとめておいて、次のような設定を行なうことができます:

```
(setq gnus-topic-display-predicate
      (lambda (name)
        (or (not (equal name "Work"))
            (< 090000
               (string-to-number (format-time-string "%H%M%S"))
               170000))))
```

3.16.3 トピックの並べ替え

以下に示す命令で、各トピック毎に別々にグループを並べ替えることができます:

T S a 現在のトピックをグループ名のアルファベット順に並べ替えます(gnus-topic-sort-groups-by-alphabet)。

T S u 現在のトピックを未読記事の数で並べ替えます(gnus-topic-sort-groups-by-unread)。

T S l 現在のトピックをグループのレベルで並べ替えます(gnus-topic-sort-groups-by-level)。

T S v 現在のトピックをグループのスコアで並べ替えます(gnus-topic-sort-groups-by-score)。See Section 3.7 [Group Score], p. 20.

T S r 現在のトピックをグループのランクで並べ替えます(gnus-topic-sort-groups-by-rank)。See Section 3.7 [Group Score], p. 20.

T S m 現在のトピックをバックエンドの名前でアルファベット順に並べ替えます(gnus-topic-sort-groups-by-method)。

T S e 現在のトピックをサーバーの名前でアルファベット順に並べ替えます(gnus-topic-sort-groups-by-server)。

T S s 現在のトピックを、変数gnus-group-sort-function で与えられる関数に従って並べ替えます(gnus-topic-sort-groups)。

接頭引数が与えられると、これらすべてのコマンドは逆順の並べ替えを行ないます。グループの並べ替えについてのさらなる情報はSection 3.12 [Sorting Groups], p. 32, を参照してください。

3.16.4 トピックの位相構造

それでは、グループバッファーの例を見ていきましょう。

```
Gnus
  Emacs -- こいつはすげーぜ!
     3: comp.emacs
     2: alt.religion.emacs
    えっちな Emacs
     452: alt.sex.emacs
       0: comp.talk.emacs.recovery
  その他
     8: comp.binaries.fractals
    13: comp.sources.unix
```

つまり、ここでは一つのトップレベルのトピック(‘Gnus’) があり、その下に二つのトピックがあり、そのうちの一方の副トピック中に一つ副トピックがあります(トップレベルトピックは常に一つしかありません)。この構造は、以下のように表現できます:

```
(("Gnus" visible)
 (("Emacs -- こいつはすげーぜ!" visible)
  (("えっちな Emacs" visible)))
 (("その他" visible)))
```

これは実に、上記の表示を行なうための、変数gnus-topic-topology の値そのものなのです。この変数は.newsrc.eld ファイルに保存され、手でいじくり回してはいけません---本当にやりたいときは別ですが。この変数は.newsrc.eld ファイルから読み込まれるので、その他のスタートアップファイルの設定にはまったく影響を与えません。

この構造は、どのトピックがどのトピックの副トピックであるかと、どのトピックが表示されているかを示しています。現在は二つの設定値—visible とinvisible を使うことができます。

3.16.5 トピックパラメーター

トピック内のすべてのグループはグループパラメーターを、その親(と先祖) のトピックパラメーターから継承します。グループパラメーターとして正しいものはすべて、トピックパラメーターとしても正しいものです(see Section 3.10 [Group Parameters], p. 23)。エージェントを使うようにしてあると、すべてのエージェントパラメーター(Section 7.9.2.1 [Category Syntax], p. 218, の Agent Parameters を参照(訳注: 必要なら Index を使って)) は有効なトピックパラメーターでもあります。

さらに、以下のパラメーターはトピックパラメーターとしてのみ有効です:

subscribe
: トピックで新しいグループを購読し始める場合(see Section 2.4.2 [Subscription Methods], p. 6) に、どのグループがどのトピックに行くかをsubscribe トピックパラメーターで指定します。値はそのトピックに行くグループに合致する正規表現である必要があります。

subscribe-level
: トピックで新しいグループを購読し始める場合(subscribe パラメーターを参照)、そのグループの購読度のレベルはgnus-level-default-subscribed の代わりにsubscribe-level トピックパラメーターの値になります。

グループパラメーターは(もちろん) トピックパラメーターよりも優先され、副トピックのトピックパラメーターは親トピックのトピックパラメーターよりも優先されます。分かるよね。ごく普通の継承ルールです(ルール(*Rules*) はここでは名詞であって、動詞の「線を引く」ではありません。このルールには反対したくなるかもしれないけど、それはご自由に)。

```
Gnus
  Emacs
     3: comp.emacs
     2: alt.religion.emacs
   452: alt.sex.emacs
    息抜き
     452: alt.sex.emacs
       0: comp.talk.emacs.recovery
  その他
     8: comp.binaries.fractals
    13: comp.sources.unix
   452: alt.sex.emacs
```

‘Emacs’ トピックはトピックパラメーター(score-file . "emacs.SCORE") を持っています。‘息抜き’ トピックはトピックパラメーター(score-file . "relief.SCORE") を持ち、‘その他’ トピックはトピックパラメーター(score-file . "emacs.SCORE") を持

っています。さらに、‘alt.religion.emacs’ はグループパラメーター(score-file . "religion.SCORE") を持っています。

さて、ここで‘息抜き’ トピックの‘alt.sex.emacs’ グループに入ったとき、relief.SCORE が基本スコアファイルとなります。もし‘Emacs’ トピックの同じグループに入ると、emacs.SCORE が基本スコアファイルになるでしょう。‘alt.religion.emacs’ グループに入れば、religion.SCORE が基本スコアファイルになるでしょう。

これってとっても簡単で自明のことのように見えるでしょ? まあ、その通りです。ですが問題がある場合もあります。特にtotal-expiry パラメーターに関してです。例えばあるメールグループを二つのトピックの中に、一方はtotal-expiry ありで、もう一方はそれなしで持っているとしましょう。ここで*M-x gnus-expire-all-expirable-groups* を実行すると、何が起こるでしょうか? Gnus は、あなたがどちらのトピックから記事を期限切れ消去したいのかを知る方法がないため、最悪の事態が発生するかもしれません。実際、私はこのとき何が起こるのかは「未定義*undefined*」である、とここに宣言します。この手のことをやりたい場合には十分注意しなければなりません。

3.17 英字以外の名前のグループへのアクセス

世界には、それぞれの母国語で表した名前のグループを提供するニュースサーバーがあります。例えば、あるニュースサーバーには名前が中国語で綴られたニュースグループがあって、そこで人々は中国語で話をしています。もちろん Gnus でそのようなニュースグループを購読することは可能です。Gnus は現在nntp バックエンドだけでなくnnml バックエンドとnnrss バックエンドで非-ASCII グループ名をサポートしています。

すべてのそのようなグループ名は、サーバー側で、ある文字セットでエンコードされています(NNTP サーバーでは管理者が文字セットを決めますが、他のバックエンドのグループではあなたがそれを決めます)。Gnus はあなたのためにグループバッファーと記事バッファーでデコードされたものを表示しなければなりませんし、サーバーと通信するときはエンコードされたものを使う必要があります。しかし Gnus は、各々の非-ASCII グループ名にどんな文字セットが使われているかを知りません。以下の二つの変数は、まさに各グループでどんな文字セットを使うべきかを Gnus に伝えるためのものです:

gnus-group-name-charset-method-alist

選択方法と文字セットの連想リストです。ディフォルト値はnil です。その選択方法で指定されるサーバーにあるグループの名前は、すべてその対応する文字セットを使うものと仮定されます。例です:

```
(setq gnus-group-name-charset-method-alist
      '(((nntp "news.com.cn") . cn-gb-2312)))
```

この変数でグループに指定された文字セットは、同じグループにgnus-group-name-charset-group-alist 変数で指定されたものより優先されます(下記参照)。

選択方法は非常に長くなる場合がありますね。このように:

```
(nntp "gmane"
      (nntp-address "news.gmane.io")
      (nntp-end-of-line "\n")
      (nntp-open-connection-function
       nntp-open-via-rlogin-and-telnet)
      (nntp-via-rlogin-command "ssh")
```

```
(nntp-via-rlogin-command-switches
 ("-C" "-t" "-e" "none"))
(nntp-via-address ...))
```

そのような場合、この変数の中では(nntp "gmane") に切り詰めることができます。つまり、バックエンド名とサーバー名だけを含んでいれば十分です。

gnus-group-name-charset-group-alist
: グループ名の正規表現と、それに合致するグループのための文字セットの連想リストです。UTF-8 がサポートされている場合は((".*" utf-8)) がディフォルト値で、それ以外の場合のディフォルトはnil です。例です:

```
(setq gnus-group-name-charset-group-alist
      '(("\\.com\\.cn:" . cn-gb-2312)
        (".*" . utf-8)))
```

gnus-group-name-charset-method-alist で合致した場合、この変数は無
視されることに注意してください。

これら二つの変数はnntp 以外のバックエンドにある非-ASCII 名のグループをエンコードおよびデコードする文字セットを決定するためにも使われます。つまり、それを決めるのはあなたです。何もしなければそれらのバックエンドのグループ名で使われる文字セットは、gnus-group-name-charset-group-alist の最後の素子のせいですべてutf-8 になるでしょう。

さて、非-ASCII グループ名のための重要な変数がもう一つあります:

nnmail-pathname-coding-system
: この変数の値は coding system もしくはnil でなければなりません。ディフォルトはnil です。

nnml バックエンド、nnrss バックエンド、エージェント、およびキャッシュは、それらのファイルとディレクトリーで非-ASCII グループ名を使います。この変数は、それらのファイル名とディレクトリー名をエンコードおよびデコードするときに用いられる coding system を指定するfile-name-coding-system の値を上書きします。

file-name-coding-system の値がnil だった場合、またはnil であるnnmail-pathname-coding-system の値にそれが束縛される場合、default-file-name-coding-system の値を使用します。

通常default-file-name-coding-system の値は言語環境に従って初期化されるので、その値が非-ASCII グループ名をエンコードおよびデコードするために適切であれば、何もする必要は無いでしょう。

この変数(またはdefault-file-name-coding-system) の値は、必ずしもgnus-group-name-charset-method-alist およびgnus-group-name-charset-group-alist によって決定される値と同じである必要はありません。

例えば中国語で綴られたニュースグループを購読したいのに、default-file-name-coding-system またはこの変数がディフォルトでiso-latin-1-unix に初期化されてしまうのならば、それはnnmail-pathname-coding-system をカスタマイズしなければならない最も典型的な場合です。utf-8-unix coding system

はそのための良い候補です。あるいは`default-file-name-coding-system` またはこの変数が適切な値に初期化されるように、あなたのシステムの言語環境を変更しても良いでしょう。

記事を非-ASCII グループから他のグループにコピーまたは移動する場合、グループ名をエンコードおよびデコードするための文字セットが両方のグループで同じでなければならないことに注意してください。さもないと記事バッファーに Newsgroups ヘッダーが正しく表示されないでしょう。

3.18 その他のグループ関連

v *v* キーはユーザー用に予約されています。そのまま何かのコマンドに割り当てても構いませんが、接頭キーとして使う方が良いでしょう。例です:

```
(keymap-set gnus-group-mode-map "v j d"
  (lambda ()
    (interactive)
    (gnus-group-jump-to-group "nndraft:drafts")))
```

Emacs でユーザー用に予約されているキーと一般的なキーバインドについては、See Section “Keymaps” in *The Emacs Editor*.

^ サーバーバッファーモードに入ります(`gnus-group-enter-server-mode`)。See Section 7.1 [Server Buffer], p. 146.

a メッセージ(ディフォルトはニュース) の作成を開始します(`gnus-group-post-news`)。接頭引数が与えられると、現在位置のグループに投稿します。もし接頭引数が 1 だったら、どのグループに投稿するかを尋ねます。この関数の名前から連想されることとは裏腹に、接頭引数でメールグループが指定された場合は、ニュースの代わりにメールの様式が用意されます。See Chapter 6 [Composing Messages], p. 135.

m メールをどこかに送ります(`gnus-group-mail`)。接頭引数が与えられると、現在位置のグループの投稿様式(posting style) を使います。もし接頭引数が 1 だったら、どのグループの投稿様式を使うかを尋ねます。See Chapter 6 [Composing Messages], p. 135.

i ニュースの作成を開始します(`gnus-group-news`)。接頭引数が与えられると、現在位置のグループに投稿します。もし接頭引数が 1 だったら、どのグループに投稿するかを尋ねます。See Chapter 6 [Composing Messages], p. 135.

この関数は、たとえメールグループで使われたとしても、実際にはニュースの様式を用意します。これは、メッセージを実際にはネットワーク経由で送らずに、メールグループに「投稿」するのに便利です; それらは当のグループに単に直接保存されます。対応するバックエンドが投稿のためのメソッド(request-post method) を持っていなければなりませんが。

G z

現在位置のグループを圧縮します(`gnus-group-compact-group`)。今のところ`nnml` (see Section 7.4.13.3 [Mail Spool], p. 188) だけに実装されています。これは記事番号のすきまを取り除くので、正しい全記事数を得ることができるようになります。

以下はグループバッファーのための変数です:

`gnus-group-mode-hook`
: グループバッファーが作成された時に呼び出されます。

`gnus-group-prepare-hook`
: グループバッファーが生成されたあとに呼び出されます。これはバッファーを何か変な、自然ではない方法で修正したいときに使われるかもしれません。

`gnus-group-prepared-hook`
: グループバッファーが生成された後の一番最後に呼び出されます。例えばポイントをどこかに移動させたいときなどに使えます。

`gnus-permanently-visible-groups`
: この正規表現に合致するグループは、それが空であるかどうかに関わらず、常にグループバッファーに表示されます。

3.18.1 新着メッセージを探す

g
: サーバーの新着記事をチェックします。数値の接頭引数を使用すると、この命令は引数*arg* かそれより小さいレベルのグループのみをチェックします(`gnus-group-get-new-news`)。数値以外の接頭引数を与えると、この命令はそのバックエンドからアクティブファイルを強制的に全部読み直します。

M-g
: 現在のグループに新着記事があるかどうかをチェックします(`gnus-group-get-new-news-this-group`)。`gnus-goto-next-group-when-activating` はこの命令が次のグループ位置へ移動するかどうかを決めます。ディフォルトは`t` です。

C-c M-g
: 無条件にすべてのグループを起動します(`gnus-activate-all-groups`)。

R
: Gnus を再起動します(`gnus-group-restart`)。これは`.newsrc` ファイルをセーブし、すべてのサーバーの接続を閉じ、すべての Gnus ランタイム変数をクリアした後、Gnus をもう一度最初から開始します。

`gnus-get-new-news-hook` は新着ニュースをチェックする直前に実行されます。

`gnus-after-getting-new-news-hook` 新着ニュースをチェックした後に実行されます。

3.18.2 グループ情報

H d
C-c C-d
: 現在のグループの説明を表示します(`gnus-group-describe-group`)。接頭引数を与えると、説明文をサーバーから強制的に再読み込みします。

M-d
: すべてのグループの説明を表示します(`gnus-group-describe-all-groups`)。接頭引数を与えると、説明文ファイルをサーバーから強制的に再読み込みします。

H v
V
: 現在の Gnus のバージョン番号を表示します(`gnus-version`)。

?
: とても短いヘルプメッセージを与えます(`gnus-group-describe-briefly`)。

C-c C-i
: Gnus の info ノードに移動します(`gnus-info-find-node`)。

3.18.3 グループの日付

Gnus に、あなたが最後にいつグループを読んだかを記録させると便利かもしれません。この活動を始めさせるには、gnus-group-set-timestamp をgnus-select-group-hook に追加してください。

```
(add-hook 'gnus-select-group-hook 'gnus-group-set-timestamp)
```

これを行なった後、あなたがグループに入るたびにそれが記録されます。

この情報はさまざまな方法で表示できます---もっとも簡単なのは、グループ行フォーマットで'%d' 指定を使う方法です:

```
(setq gnus-group-line-format
      "%M\%S\%p\%P\%5y: %(%-40,40g%) %d\n")
```

この結果として、各行は以下のように表示されます:

```
*        0: mail.ding                                19961002T012943
         0: custom                                   19961002T012713
```

見て分かるとおり、日付はコンパクトな ISO 8601 形式で表示されます。これではちょっとあんまりなので、以下のような感じにすると日付だけを表示できます。

```
(setq gnus-group-line-format
      "%M\%S\%p\%P\%5y: %(%-40,40g%) %6,6~(cut 2)d\n")
```

もっと凝った日付の形式をお望みなら、利用者定義によるフォーマットの仕様を使うことができます。以下のようなものはうまくいくでしょう:

```
(setq gnus-group-line-format
      "%M\%S\%p\%P\%5y: %(%-40,40g%) %ud\n")
(defun gnus-user-format-function-d (headers)
  (let ((time (gnus-group-timestamp gnus-tmp-group)))
    (if time
        (format-time-string "%b %d  %H:%M" time)
      "")))
```

動的に束縛されている変数(gnus-tmp-group のようなもの) に何があるかを知るにはソースコードを眺める必要があります。変数名が複数の Gnus のバージョンを通して一定していることも保証されていません。

3.18.4 ファイル命令

r 初期化ファイルの再読み込みを行ないます(gnus-init-file、ディフォルトは~/.gnus.el) (gnus-group-read-init-file)。

s .newsrc.eld ファイル(と、もしそうしたければ.newsrc ファイル) をセーブします(gnus-group-save-newsrc)。

3.18.5 Sieve コマンド

Sieve はサーバー側で使われるメールフィルター言語です。Gnus では、各グループに適用される sieve の規則を指定する、sieve グループパラメーター(see Section 3.10 [Group Parameters], p. 23) を使うことができます。guns はそれらすべてのグループパラメーターを、サーバーで使うことも可能な正しい Sieve スクリプトに翻訳する、二つのコマンドを提供します。

作成された Sieve スクリプトは`gnus-sieve-file` (ディフォルトは`~/.sieve`) に置かれます。Gnus が作るコードは二つの区切り記号`gnus-sieve-region-start` と`gnus-sieve-region-end` の間に置かれるので、これらの区切り記号の外に追加の Sieve コードを書いても、次回 Sieve スクリプトを再作成するときに消されてしまうことはありません。

変数`gnus-sieve-crosspost` は Sieve スクリプトがどのように作られるかを制御します。もし非-`nil` (ディフォルト) だったら記事は規則に合致するすべてのグループに置かれます。そうでない場合、記事は最初の規則に合致するグループだけに置かれます。例えばグループパラメーター‘`(sieve address "sender" "owner-ding@hpc.uh.edu")`’ は、`gnus-sieve-crosspost` が`nil` だったら以下の Sieve コードの断片を作ります。(`gnus-sieve-crosspost` が非-`nil` だった場合は、行が含む`stop` の呼び出しが削除されること以外は同じです。)

```
if address "sender" "owner-ding@hpc.uh.edu" {
        fileinto "INBOX.ding";
        stop;
}
```

See *Emacs Sieve*.

D g `sieve` グループパラメーターから Sieve スクリプトを再作成して、`gnus-sieve-file` に書き込みます。以前の内容は保存されません。

D u `sieve` グループパラメーターを元に`gnus-sieve-file` の Gnus が管理している部分を再作成してファイルにセーブし、`sieveshell` プログラムを使ってサーバーにアップロードします。

4 概略バッファー

概略バッファー(summary buffer) ではそれぞれの記事が一行で表示されます。その中を動き回り、記事を読み、投稿し、返答をすることができます。

概略バッファーに移る一番普通の方法は、グループバッファーでグループを選択することです(see Section 3.3 [Selecting a Group], p. 16)。

好きなだけたくさんの概略バッファーを開いておくことができます。

概略モードのツール・バーをカスタマイズすることができます。*M-x customize-apropos* RET *gnus-summary-tool-bar* の結果を参照してください。

v キーはユーザー用に予約されています。そのまま何かのコマンドに割り当てても構いませんが、接頭キーとして使う方が良いでしょう。例です:

```
;; 副スレッドのスコアを下げる。
(keymap-set gnus-summary-mode-map "v -" "LrS")
```

4.1 概略バッファーの様式

Gnus は変数gnus-extract-address-components の値をFrom ヘッダーの名前とアドレスの部分を抽出するための関数として使います。すでに定義されている関数が二つ存在します: ディフォルトはgnus-extract-address-components で、とても簡単に割り切った解決法ですが非常に速く動作します。mail-extract-address-components は良く動作しますが遅いです。ディフォルトの関数は 5% の割合で間違った答を返します。もしこれに我慢ならないのであれば、代わりに他の関数を使ってください:

```
(setq gnus-extract-address-components
      'mail-extract-address-components)
```

gnus-summary-same-subject は今読んでいる記事が、その前の記事と同じ表題(subject) であることを示す文字列です。この文字列は、それを要求する書法仕様で使われます。ディフォルトでは"" です。

4.1.1 概略バッファーの行

変数gnus-summary-line-format の値を変えることによって、概略バッファーの行の様式(format) を変更することができます。いくつかの拡張(see Section 10.4 [Formatting Variables], p. 272) とともに、普通のformat 文字列と同じように動作します。

行には常にコロンかポイント位置のマーカーが存在していなければなりません。操作した後に、カーソルはいつもコロンかポイント位置のマーカーの場所に移動します。(もちろん、この動作を変えることができないとしたら Gnus にはあるまじきことです。関数gnus-goto-colon を、あなたが好きなカーソルの動きになるように、新たに書けば良いのです。) See Section 10.4.6 [Positioning Point], p. 276.

ディフォルトの文字列は‘%U%R%z%I%(%[%4L: %-23,23f%]%) %s\n’ です。

以下の様式指示文字と拡張様式指示を使うことができます:

‘N’ 記事数。

‘S’ 表題の文字列。gnus-list-identifiers の設定によってメーリングリストの標識が削除されます。See Section 4.18.3 [Article Hiding], p. 92.

‘s’ スレッド(thread) の元記事であるときか直前の記事が違う表題のときはその表題で、それ以外は`gnus-summary-same-subject`。(`gnus-summary-same-subject` はディフォルトで`""`。)

‘F’ 完全な`From` 欄。

‘n’ 名前(`From` 欄より)。

‘f’ 名前、`To` 欄の内容、または`Newsgroups` 欄の内容のどれかです(see Section 4.1.2 [To From Newsgroups], p. 50)。

‘a’ 名前(`From` 欄より)。これと`n` との違いは、これは変数`gnus-extract-address-components` で指定されている関数を使って名前を取得することです。この方が遅いですが、おそらくより完全に近いでしょう。

‘A’ 名前(`From` 欄より)。これは`a` と同じように動作します。

‘L’ 記事の行数。

‘Z’ 記事の検索スコア値(RSV)。nnselect グループにない場合は`nil`。

‘G’ 記事の元のグループ名。nnselect グループにない場合は`nil`。

‘g’ 記事の元のグループ名の短縮形。nnselect グループにない場合は`nil`。

‘c’ 記事の文字数。この名前指定子は(nnfolder のような) いくつかの選択方法をサポートしません。

‘k’ 整形された記事の文字数; 例えば‘`1.2k`’ や‘`0.4M`’。

‘I’ スレッドのレベルによる字下げ(see Section 4.9.1 [Customizing Threading], p. 70)。

‘B’ 複雑な trn 様式のスレッド木(tree)。どのような応答が行なわれたかの記録を表示します。スレッドはこのように描かれるでしょう:

```
>
+->
| +->
| | \->
| |   \->
| \->
+->
\->
```

以下のオプションで見栄えをカスタマイズすることができます。ディフォルトのASCII 文字を線描画用の図案で置き換えることによって、スレッド表示を実に巧妙に見せることができることに気付いてください。

`gnus-sum-thread-tree-root`
: スレッドの根(root) に使われます。`nil` だったら、代わりに表題を使います。ディフォルトは‘`> `’ です。

`gnus-sum-thread-tree-false-root`
: スレッドのにせの根に使われます(see Section 4.9.1.1 [Loose Threads], p. 70)。`nil` だったら、代わりに表題を使います。ディフォルトは‘`> `’ です。

`gnus-sum-thread-tree-single-indent`
単一のメッセージのスレッドに使われます。`nil` だったら、代わりに表題を使います。ディフォルトは‘’ です。

`gnus-sum-thread-tree-vertical`
縦線の描画に使われます。ディフォルトは‘| ’ です。

`gnus-sum-thread-tree-indent`
行下げ(indenting) に使われます。ディフォルトは‘ ’ です。

`gnus-sum-thread-tree-leaf-with-other`
兄弟がいる葉っぱに使われます。ディフォルトは‘+-> ’ です。

`gnus-sum-thread-tree-single-leaf`
兄弟がいない葉っぱに使われます。ディフォルトは‘\-> ’ です。

‘T’ 記事が元記事であれば何も表示せず、そうでない場合はたくさんの空白です(それより後のものをすべて画面の外に追い出してしまいます)。

‘[’ 開き括弧。普通は‘[’ ですが、養子記事には‘<’ にすることができます(see Section 4.9.1 [Customizing Threading], p. 70)。これは次の設定を使ってカスタマイズすることができます:

`gnus-sum-opening-bracket`
通常の(養子ではない) 記事の開き括弧。ディフォルトは‘[’ です。

`gnus-sum-opening-bracket-adopted`
養子記事の開き括弧。ディフォルトは‘<’ です。

‘]’ 閉じ括弧。普通は‘]’ ですが、養子記事には‘>’ にすることができます。これは次の設定を使ってカスタマイズすることができます:

`gnus-sum-closing-bracket`
通常の(養子ではない) 記事の閉じ括弧。ディフォルトは‘]’ です。

`gnus-sum-closing-bracket-adopted`
養子記事の閉じ括弧。ディフォルトは‘>’ です。

‘>’ それぞれのスレッドのレベルに対して一つの空白。

‘<’ (20 - スレッドレベル) 個の空白。

‘U’ 未読。See Section 4.7.2 [Read Articles], p. 63.

‘R’ この紛らわしい名前指定子は「第二の印」(the secondary mark) です。この印は記事がすでに返答済みのものか、キャッシュされたものか、あるいは保存されたものかを表します。See Section 4.7.3 [Other Marks], p. 63.

‘i’ 数値としてのスコア(see Chapter 8 [Scoring], p. 233)。

‘z’ これは、zcore でディフォルトのレベルよりも上であれば‘+’ で、ディフォルトのレベルよりも下であれば‘-’ です。`gnus-summary-default-score` との差が`gnus-summary-zcore-fuzz` よりも小さいと、この仕様は使われません。

‘V’ スレッド全体のスコア。

‘x’ `Xref`.

‘D’ Date.

‘d’ DD-MM 様式によるDate。

‘o’ *YYYYMMDD*T*HHMMSS* 様式によるDate。

‘M’ Message-ID.

‘r’ References.

‘t’ 現在の副スレッドの記事の数。この仕様を使うと概略バッファーの生成が幾分遅くなります。

‘e’ 記事に子記事があると、‘=’ (gnus-not-empty-thread-mark) が表示されます。

‘P’ 行数。

‘O’ ダウンロードの印。

‘*’ カーソルを(最初のコロンの後ろの代わりに) 置く場所。

‘&user-date;’
経過時間の様式。いろいろな様式がgnus-user-date-format-alist で定義されています。

‘u’ 利用者定義指定子。フォーマット文字列の中の次の文字は英字でなければなりません。これにより Gnus は関数gnus-user-format-function-*x* を呼び出しますが、ここで*x* は‘%u’ の次の文字です。関数には現在の記事のヘッダーが引数として渡されます。関数は文字列を返さなければなりません。それは他の概略指定と同様に概略に挿入されます。

‘%(’ と‘%)’ の間にあるテキストは、そこにマウスがあるときにgnus-mouse-face でハイライトされます。そういう領域は一つだけです。

‘%U’ (状態), ‘%R’ (返答済み), ‘%z’ (zcore) の扱いには気を付ける必要があります。効率のために、Gnus はこれらの文字がどの桁に現れるかを計算し、『ハード・コード』します。これは、可変長の仕様の後では、これらは意味を持たないということです。まぁ、さすがに逮捕はされないでしょうが、概略バッファーは変になります。それでも十分悲しいでしょうけど。

賢い選択はこれらの指定をできるだけ左に持ってくることです。(でも、そういうことはすべてに当てはまるのではないでしょうか。閑話休題。)

この制限は将来の版では無くなるかもしれません。

4.1.2 To From Newsgroups

いくつかのグループ(特にアーカイブグループ) ではFrom ヘッダーはあまり興味を引きません。そこのすべての記事はあなたによって書かれたものですから。代わりに、ToやNewsgroups ヘッダーの情報を表示するためには、三つのことを決める必要があります: どの情報を集めるか, どこに表示するか, いつ表示するか。

1. 追加のヘッダーの情報はgnus-extra-headers により制御されます。これはヘッダーのシンボルのリストです。例えば:

```
(setq gnus-extra-headers
      '(To Newsgroups X-Newsreader))
```

これによって Gnus はこれらの三つのヘッダーを取得しようとし、後の容易な取得のためにヘッダー構造に保存します。

2. これらの追加のヘッダーの値はgnus-extra-function 関数を通じて取得することができます。これはX-Newsreader ヘッダーを使う書式行の仕様です:

```
"%~(form (gnus-extra-header 'X-Newsreader))@"
```

3. gnus-ignored-from-addresses 変数はいつ‘%f’ 概略行仕様がTo, Newsreader やFrom ヘッダーを返せば良いかを決めます。この変数は正規表現、または判断するための関数です。もしこれがFrom ヘッダーの内容に合致すると、To やNewsreader ヘッダーの値が代わりに使われます。

   それらのFrom フィールドが入れ替わっている記事と、普通の記事を区別するために、概略行のTo またはNewsgroups ヘッダーに、ある文字列が前置されます。その文字列はディフォルトで、To には‘-> ’ が、Newsgroups には‘=> ’ が使われますが、gnus-summary-to-prefix とgnus-summary-newsgroup-prefix によって、それらの文字列をカスタマイズすることができます。

関連する変数はnnmail-extra-headers で、overview (NOV) ファイルを作る際にいつ追加のヘッダーを含めるかを制御します。古い overview ファイルがある場合は、この変数を変更した後にサーバーバッファーに^ で入って適切なメールサーバー(例えば nnml) で*g* を押し、再生成する必要があります。

さらにgnus-summary-line-format 変数の%n 仕様を%f 仕様に変更することによってデータを表示するように、Gnus に指示する必要があります。

要約すると、普通は以下のようなものを~/.gnus.el に置くことになります:

```
(setq gnus-extra-headers
      '(To Newsgroups))
(setq nnmail-extra-headers gnus-extra-headers)
(setq gnus-summary-line-format
      "%U%R%z%I%(%[%4L: %-23,23f%]%) %s\n")
(setq gnus-ignored-from-addresses
      "Your Name Here")
```

(上記の値は Gnus のディフォルト値です。あなたの役に立つように変えてください。)

ニュース管理人、またはニュース管理人を説得してサポートの追加をしてもらおうと思っている利用者のみなさんへのご注意:

NOV ファイルの生成を制御できるメールグループでは、上記のことはたいていの場合役立ちます。しかし、管理人を説得して(特に INN の普通の実装において) 以下のものをoverview.fmt ファイルの最後に追加してもうらうことができれば、メールグループでの追加ヘッダーのようにそれを使うことができます。

```
Newsgroups:full
```

それができない場合は、‘XOVER’ を使ってヘッダーの取り込みを無効にしなければなりません:

```
(setq nntp-nov-is-evil t
      gnus-nov-is-evil t)
```

ただし、これはNNTP グループに入ることを、非常に、非常に遅くしてしまうので、お勧めはしません。

‘XOVER’ を使わないことが望ましい 1 つの特別なシナリオは、例えばnnvirtual グループにおいて、それを構成するグループ群による制限をサポートする場合です。グループパラメーターは継承されないので、その構成グループ群のためのnov-is-evil が変数として適切に設定されている個別の選択方法が必要です。

4.1.3 概略バッファーのモード行

概略のモード行の様式も変更することができます(see Section 10.4.2 [Mode Line Formatting], p. 273)。`gnus-summary-mode-line-format` を何でも好きなものに設定してください。ディフォルトは‘`Gnus: %%b [%A] %Z`’ です。

以下はあなたが遊ぶことのできる要素たちです:

‘`G`’ グループ名。

‘`p`’ 接頭語を取り除いた名前。

‘`A`’ 現在の記事番号。

‘`z`’ 現在の記事スコア。

‘`V`’ Gnus バージョン。

‘`U`’ そのグループでの未読記事の数。

‘`e`’ 概略バッファーに表示されていない未読記事の数。

‘`Z`’ 未読と未選択の記事の数とともに表される文字列で、未読かつ未選択の記事がある場合の‘`<%U(+%e) more>`’、および未読記事のみの場合の‘`<%U more>`’ のどちらかです。

‘`g`’ 短縮グループ名。例えば、‘`rec.arts.anime`’ は‘`r.a.anime`’ に短縮されます。

‘`S`’ 現在の記事の表題。

‘`u`’ 利用者定義の仕様(see Section 10.4.4 [User-Defined Specs], p. 274)。

‘`s`’ 現在のスコアファイルの名前(see Chapter 8 [Scoring], p. 233)。

‘`d`’ 保留記事の数(see Section 4.7.1 [Unread Articles], p. 62)。

‘`t`’ 可視印付き記事の数(see Section 4.7.1 [Unread Articles], p. 62)。

‘`r`’ その概略バッファーで記事を読んだ結果、既読の印が付いた記事の数。

‘`E`’ スコアファイルによって抹消された記事の数。

4.1.4 概略のハイライト

`gnus-visual-mark-article-hook`
: このフックは記事を選択した後に実行されます。これは何らかの方法で記事をハイライトするように意図されています。`gnus-visual` が`nil` だったら実行されません。

`gnus-summary-update-hook`
: このフックは概略行が変化したときに呼ばれます。`gnus-visual` が`nil` だったら実行されません。

`gnus-summary-selected-face`
: これは概略バッファーでの現在の記事をハイライトするために使われるフェース(もしくは、ある人たちが「フォント」と呼ぶようなもの) です。

`gnus-summary-highlight`
: 概略行はこの変数にしたがってハイライトされます。この変数は要素が(*form* . *face*) の形式のリストです。例えば、印付きの記事を斜体、高いスコアの記事を太字にしたければ、この変数を次のように設定することができます。

```
(((eq mark gnus-ticked-mark) . italic)
 ((> score default) . bold))
```

ご想像のとおり、*form* が`nil` でない値を返すと、*face* がその行に適用されます。

4.2 概略間の移動

すべての直接移動命令は数値接頭引数を受け付け、かなり期待どおりに動作するでしょう。

これらの命令はどれも記事を選択しません。

G M-n

M-n
: 概略行の次の未読記事に移ります(`gnus-summary-next-unread-subject`)。

G M-p

M-p
: 概略行の前の未読記事に移ります(`gnus-summary-prev-unread-subject`)。

G g
: 記事番号を尋ね、その記事を表示せずに、その概略行に行きます(`gnus-summary-goto-subject`)。

Gnus が次のグループ移動することを確認するためにキー入力を求めた場合、*C-n* キーと*C-p* キーを使うことによって、実際にグループバッファーに戻らなくても、次に読むグループを探すことができます。

概略の移動に関連した変数:

`gnus-auto-select-next`
: 移動命令の一つ(*n* のような) を発したときに現在の記事の後に未読記事が無いと、Gnus は次のグループに移動することをうながします。この変数が`t` で次のグループが空っぽだったら、Gnus は概略モードを抜けてグループバッファーに戻ります。この変数が`t` でも`nil` でもなければ、Gnus はさらに次の未読記事があるグループを選択します。特別な場合として、この変数が`quietly` だったら、Gnus は確認をせずに次のグループを選択します。この変数が`almost-quietly` だった場合は、グループの一番最後の記事を読んでいたときに限って同じことが起こります。最後に、もしこの変数が`slightly-quietly` だったら、*Z n* 命令は確認をせずに次のグループに移ります。Section 3.6 [Group Levels], p. 19, も参照してください。

`gnus-auto-select-same`
: `nil` でないと、すべての移動命令は現在の記事と同じ表題の記事に移動しようとします。(「同じ」はここでは「大体同じ」という意味かもしれません。詳細は`gnus-summary-gather-subject-limit` を見てください(see Section 4.9.1 [Customizing Threading], p. 70)。) 同じ表題の記事が無いときは、最初の未読記事に移動します。

この変数は、スレッド表示を行なっているときはあまり役に立たないでしょう。

`gnus-paging-select-next`
: ページングするときに(*SPC* や*DEL* などのコマンドで) 次または前の記事を選択するかどうかを制御します。`nil` 以外の場合は、記事の最後に到達したときに次の記事を選択します(または、逆方向にページングする場合は前の記事を選択します)。

もし`nil` だったら記事の末尾や先頭で何もしません。

`gnus-summary-check-current`
: これが`nil` ではない場合、すべての『未読』移動命令は、現在の記事が未読だったら次(もしくは前) の記事に移動しません。代わりに、それらは現在の記事を選びます。

`gnus-auto-center-summary`
: `nil` でないと、Gnus は概略バッファーでのポイントを常に真中に保ちます。これをすると、とてもこぎれいになりますが、遅いネットワークに接続していたり、この Emacs らしくない流儀が好きになれないのであれば、この変数を`nil` にすることによって、普通の Emacs のスクロールにすることができます。これは概略バッファーの水平方向でポイントが真ん中になるようにする操作(horizontal re-centering) も禁止してしまうので、非常に長いスレッドを読むときは不便かもしれません。

この変数は数値でも構いません。その場合は、ウィンドウの先頭からその数の行だけ下がった位置に常にポイントがあるように制御されます。

`gnus-summary-stop-at-end-of-message`
: もし`nil` でなければ、`SPC` を叩いても次の記事に行かずに、その記事の最後にとどまります。

4.3 記事の選択

4.3.1 選択命令

以下の移動コマンドはどれも数値接頭引数を受け付けません。それらはすべて、記事を選択して表示します。

新しい記事を取り込んだり、グループを再表示したいときはSection 4.28 [Exiting the Summary Buffer], p. 119, を参照してください。

`SPC`
: 現在の記事、またはそれが既読だった場合は次の未読記事を選択します(`gnus-summary-next-page`)。

すでに記事ウィンドウを開いているときに再び`SPC` を押すと、その記事はスクロールされます。これによって、ニュースグループ全体を`SPC` だけで便利に通読することができます。See Section 4.4 [Paging the Article], p. 56.

G n
n
: 次の未読記事に移動します(`gnus-summary-next-unread-article`)。

G p
p
: 前の未読記事に移動します(`gnus-summary-prev-unread-article`)。

G N
N
: 次の記事に移動します(`gnus-summary-next-article`)。

G P
P 前の記事に移動します(gnus-summary-prev-article)。

G u
] 次のまだ読まれたことが無い記事に移動します(gnus-summary-next-unseen-article)。

G U
[前のまだ読まれたことが無い記事に移動しますgnus-summary-prev-unseen-article)。

G C-n 同じ表題の次の記事に移動します(gnus-summary-next-same-subject)。

G C-p 同じ表題の前の記事に移動します(gnus-summary-prev-same-subject)。

G f
. 最初の未読記事に移動します(gnus-summary-first-unread-article)。

G b
, 最高スコアの未読記事に移動します(gnus-summary-best-unread-article)。接頭引数が与えられると、ディフォルトのスコアより大きいスコアを持つ最初の未読記事に移動します。

G l
l 直前に読んだ記事に移動します(gnus-summary-goto-last-article)。

G o 概略の履歴(history) から最後の記事を一つ取り出して選択します(gnus-summary-pop-article)。この命令が上の命令と違うのは、*l* が最後の二つの記事の間を移動するだけなのに対して、これは好きなだけ前の記事を履歴から選び出すことができる点です。これに多少関係することについて、Section 4.15 [Article Backlog], p. 81, を参照してください(これらの命令をたくさん使うのであれば)。

G j
j 記事番号かMessage-ID を尋ね、それからその記事に行きます(gnus-summary-goto-article)。

4.3.2 選ぶための変数

記事の移動と選択に関連するいくつかの変数:

gnus-auto-extend-newsgroup
この変数がnil でないと、すべての移動命令は、記事が概略バッファーに表示されていない場合でも、前(もしくは次) の記事に移動しようとします。その際 Gnus はサーバーから記事を取得して、記事バッファーに表示します。

gnus-select-article-hook
このフックは記事が選択されたときに常に呼ばれます。ディフォルトはnil です。購読するそれぞれの記事をエージェントに保存させたい場合は、このフックにgnus-agent-fetch-selected-article を追加すれば良いでしょう。

gnus-mark-article-hook
このフックは記事が選択されたときに常に呼ばれます。これは記事に既読の印を付けるために使われることを意図しています。ディフォルト値

はgnus-summary-mark-read-and-unread-as-read で、ほとんどすべての読んだ記事の印をgnus-read-mark に変更します。この関数に影響されない記事は、可視、保留、期限切れ消去可能記事だけです。未読記事に既読の印を付けたいだけであれば、代わりにgnus-summary-mark-unread-as-read を使うことができます。gnus-low-score-mark やgnus-del-mark (など) の印はそのまま残します。

4.4 記事のスクロール

SPC
: SPC を押すと、現在の記事を一ページ先にスクロールします。記事の最後に行き着いた場合は次の記事を選択します(gnus-summary-next-page)。

  gnus-article-skip-boring が非-nil で、かつ記事の残りに引用と署名しか無い場合、それはスキップされ、代わりに次の記事が表示されます。gnus-article-boring-faces で、つまらないと思うものをカスタマイズすることができます。どんなにうんざりするものでも、*C-M-v* を使うことによって、手動で記事のページを見ることはできます。

DEL
: 現在の記事を一ページ前にスクロールします(gnus-summary-prev-page)。

RET
: 現在の記事を一行先にスクロールします(gnus-summary-scroll-up)。

M-RET
: 現在の記事を一行後ろへスクロールします(gnus-summary-scroll-down)。

A g
g
: 現在の記事を(再) 取得します(gnus-summary-show-article)。もし接頭引数が一つ与えられると、サーバーから来たままの完全な『生の』記事を表示します。もし接頭引数が 2 回与えられると(つまり*C-u C-u g*')、現在の記事を取得しますが、記事をトリートメントする関数は実行しません。

  接頭引数を与えると、手動で文字セットの操作を行なうことができます。*C-u 0 g cn-gb-2312* RET により、メッセージはあたかもcn-gb-2312 文字セットでエンコードされたかのようにデコードされます。以下のような設定を用意しておくと、*C-u 1 g* で同じ効果を得ることができます。

```
(setq gnus-summary-show-article-charset-alist
      '((1 . cn-gb-2312)
        (2 . big5)))
```

A <
<
: 記事の最初までスクロールします。(gnus-summary-beginning-of-article)。

A >
>
: 記事の最後までスクロールします(gnus-summary-end-of-article)。

A s
s
: 記事バッファーでインクリメンタルサーチ(isearch) を行ないます(gnus-summary-isearch-article)。

h
: 記事バッファーを選択します(gnus-summary-select-article-buffer)。

4.5 返答、フォローアップ、投稿

4.5.1 概略でのメールの命令

メールメッセージを作成するための命令:

S r

r 現在の記事を書いた人に返答のメールを送ります(gnus-summary-reply)。

S R

R 現在の記事を書いた人に、元記事を含んだ返答のメールを出します(gnus-summary-reply-with-original)。この命令はプロセス/接頭引数の習慣を使います。

S w 現在の記事を書いた人に対して、広い返答(wide reply) をします(gnus-summary-wide-reply)。「広い返答」とはヘッダーのTo, From, (もしくはReply-To) とCc) のすべての人に返答をすることです。Mail-Followup-To があれば、代わりにそれが使われます。

S W 現在の記事に元記事を含んだ広い返答のメールを送ります(gnus-summary-wide-reply-with-original)。この命令はプロセス/接頭引数の習慣を使います。ただし、受取人の決定には最初の記事のヘッダーだけを使います。

S L メーリングリストで配信されたメッセージに返事をするとき、元のメッセージの引用付きでそのメーリングリストに返信します(gnus-summary-reply-to-list-with-original)。

S v 現在の記事を書いた人に対して、非常に広い返答(very wide reply) をします(gnus-summary-very-wide-reply)。「非常に広い返答」とは、プロセス/接頭引数で指定されたすべての記事のヘッダーのTo, From, (もしくはReply-To) とCc) のすべての人に返答をすることです。この命令はプロセス/接頭引数の習慣を使います。もし接頭引数を与えると、現在の記事の本文を引用します。

S V 現在の記事に元記事を含んだ非常に広い返答のメールを送ります(gnus-summary-very-wide-reply-with-original)。この命令はプロセス/接頭引数の習慣を使います。

S B r 現在の記事を書いた人に対して返答をしますがReply-To フィールドは無視します(gnus-summary-reply-broken-reply-to)。メーリングリストがそのリストを指すReply-To を過って設定するためにこれが必要なのであれば、おそらくあなたは代わりにbroken-reply-to グループパラメーターを設定する必要があります。そうすれば、ものごとは正しく働くようになるでしょう。See Section 3.10 [Group Parameters], p. 23.

S B R 現在の記事を書いた人に対して元記事を含んだ返答をしますがReply-To フィールドは無視します(gnus-summary-reply-broken-reply-to-with-original)。

S o m

C-c C-f 誰か他の人に現在の記事を転送します(gnus-summary-mail-forward)。接頭引数が与えられない場合、メッセージはmessage-forward-as-mime およびmessage-forward-show-mml の値に従ったやり方で転送されます。接頭引数が 1 だったら、デコードしたメッセージをそのまま転送します。2 だったらrfc822 形式のMIME パートを転送します(訳注: この場合、元のメッセージはデコードされません)。3 ではデコードしたメッセージを rfc822 形式のMIME パートとして転送します(訳注: 送信する際に再びエンコードされます)。接頭引

数 4 ではメッセージをそのまま転送します。接頭引数がこれら以外の場合にはmessage-forward-as-mime の値を一時的に反転して、接頭引数が与えられなかった場合と同じ動作を行ないます。ディフォルトでは、転送するメッセージをそのメールに埋め込み(インライン) します。

元のメッセージのどのヘッダーが転送メッセージに含まれるかは、message-mode に固有のオプションによって決まります(see Section “転送” in *Message manual*)。加えて、このコマンドには*M-i* a を使ってシンボル接頭引数‘a’ を付けることができるので、元のヘッダーほとんどを含めることが可能です。

S m
m メールを作成します(gnus-summary-mail-other-window)。ディフォルトでは現在のグループの投稿様式(posting style) を使います。接頭引数が与えられると、それは使いません。もし接頭引数が 1 だったら、どのグループの投稿様式を使うかを尋ねます。

S i ニュースを作成します(gnus-summary-news-other-window)。ディフォルトでは現在のグループに投稿します。接頭引数が与えられると、現在のグループ名は使われません。もし接頭引数が 1 だったら、どのグループに投稿するかを尋ねます。

この関数は、たとえメールグループで使われたとしても、実際にはニュースの様式を用意します。これは、メッセージを実際にはネットワーク経由で送らずに、メールグループに「投稿」するのに便利です; それらは当のグループに単に直接セーブされます。対応するバックエンドが投稿のためのメソッド(request-post method) を持っていなければなりませんが。

S D b メールを送ったのに、何らかの理由(アドレスの間違い、転送の不調) で戻ってきたときに、この命令を使って戻ってきたメールをもう一回送ることができます(gnus-summary-resend-bonced-mail)。メールバッファーにそのメールが現れて、そこでもう一度メールを送る前にヘッダーを編集することができます。この命令に接頭引数を与えると、戻ってきたメールが何か他のメールへの返答であった場合に、Gnus はそのメールを取得して、それのヘッダーの精密調査ができるように画面に表示しようとします。ま、これはとてもよく失敗しますけど。

S D r 上の命令と混同しないでください。gnus-summary-resend-message は現在のメッセージを送る宛先のアドレスの入力を促して、その場所にメールを送ります。メッセージのヘッダーは変更されません---しかしResent-To, Resent-From などの、たくさんのヘッダーが付け加えます。これは、(おそらく) あなた自身をTo 欄に書いた本人にもメールを送ってしまうということです。これは混乱を招くでしょう。ですから当然、あなたが本当に邪悪な人でなければ、これは使わないでしょう。

この命令は主に、あなたがいくつかのメールアカウントを持っていて、自分自身の違ったアカウントにメールを転送したいときに用いられます。(もしあなたがroot であり、postmaster でもあり、root 宛てにpostmaster へのメールを受け取った場合は、それをpostmaster にも再送する必要があるかもしれません。秩序がなければなりません! (Ordnung muss sein!))

この命令はプロセス/接頭引数の習慣に従います(see Section 10.1 [Process/Prefix], p. 271)。

S D e 一つ前のコマンドに似ていますが、再送する前にあたかも新しいメッセージのように編集することができます。

S O m 現在の一連の記事(see Section 4.17 [Decoding Articles], p. 86) の要約を作り、メールでその結果を送ります(gnus-uu-digest-mail-forward)。この命令はプロセス/接頭引数の習慣に従います(see Section 10.1 [Process/Prefix], p. 271)。

S M-c 現在の記事の書き手に、過剰なクロスポストへの苦情のメールを送ります(gnus-summary-mail-crosspost-complaint)。

この命令は、現在 Usenet に横行しているクロスポストの世界的流行に対して反撃を行なう手段として提供されています。これは変数gnus-crosspost-complaintを序文にして返答を作成します。この命令はプロセス/接頭引数の習慣(see Section 10.1 [Process/Prefix], p. 271) に従い、それぞれのメールを送る前に送信するかどうかの確認をします。

またSee Section “ヘッダー命令” in *The Message Manual*, にさらなる情報があります。

4.5.2 概略の投稿命令

ニュースの記事を投稿するための命令:

S p
a 投稿するための記事を作成します(gnus-summary-post-news)。ディフォルトでは現在のグループに投稿します。接頭引数が与えられると、現在のグループ名は使われません。もし接頭引数が 1 だったら、代わりに別のどのグループに投稿するかを尋ねます。

S f
f 現在の記事のフォローアップを投稿します(gnus-summary-followup)。

S F
F 元記事を取り込んで、現在の記事にフォローアップをします(gnus-summary-followup-with-original)。この命令はプロセス/接頭引数の習慣を用います。

S n メールのメッセージを受け取っていたとしても、現在の記事のフォローアップをニュースに投稿します(gnus-summary-followup-to-mail)。この命令はプロセス/接頭引数の習慣を用います。

S N メールのメッセージを受け取っていたとしても、元記事を引用して、現在の記事のフォローアップをニュースに投稿します(gnus-summary-followup-to-mail-with-original). この命令はプロセス/接頭引数の習慣を用います。

S o p 現在の記事をニュースグループに転送します(gnus-summary-post-forward)。接頭引数が与えられない場合、メッセージはmessage-forward-as-mime およびmessage-forward-show-mml の値に従ったやり方で転送されます。接頭引数が 1 だったら、デコードされたメッセージが直接埋め込まれた転送用のバッファーが作られます。2 だったら rfc822 形式のMIME パートが挿入されます。この場合、元のメッセージはデコードされません。3 ではデコードされた rfc822 形式のMIME パートが挿入されます(実際に送信する際に再びエンコードされます)。接頭引数 4 では、1 の場合と同じ動作になります。接頭引数がこれら以外の場合には、message-forward-as-mime の値を一時的に反転して、接頭引

数が与えられなかった場合と同じ動作を行ないます。ディフォルトでは、デコードされたメッセージが rfc822 形式のMIME パートとして生成されます。

S O p 現在の一連の記事を要約して、その結果をニュースグループに送ります(gnus-uu-digest-post-forward)。この命令はプロセス/接頭引数の習慣を用います。

S u ファイルを uuencode して分割し、それらを連続して投稿します(gnus-uu-post-news)。(see Section 4.17.5.3 [Uuencoding and Posting], p. 89)。

またSee Section “ヘッダー命令” in *The Message Manual*, にさらなる情報があります。

4.5.3 概略メッセージ命令

S y 現在の記事を、すでに存在するメッセージ作成バッファーに貼り付けます(gnus-summaryyank-message)。この命令は貼り付けたいメッセージバッファーの入力を促し、プロセス/接頭引数の習慣を理解します(see Section 10.1 [Process/Prefix], p. 271)。

S A 現在の記事を、すでに存在するメッセージ作成バッファーに添付します(gnus-summary-attach-message)。もしそのようなバッファーが無いときは新しいものを作成します。このコマンドは、どのメッセージバッファーに添付したいかを尋ねます。また、プロセス/接頭引数の習慣を理解します(see Section 10.1 [Process/Prefix], p. 271)。

4.5.4 記事を取り消す

何かを書いた後で、本当に、本当に、ほんとうにそれを投稿していなければなぁと思ったことはありませんか。

えーと、メールは取り消すことはできないのですが、ニュースの投稿は取り消すことができます。

取り消したい記事を見つけてください(取り消すことができるのは自分の記事だけです。変なことは試さないでください)。そして*C* か*S c* を押してください(gnus-summary-cancel-article)。あなたの記事が取り消されます---世界中の機械があなたの記事を取り消します。この命令はプロセス/接頭引数の習慣を用います(see Section 10.1 [Process/Prefix], p. 271)。

しかし注意して欲しいのは、すべてのサイトが取り消しを扱っているわけではないことです。ですから、たいていのサイトが問題の記事を取り消しても、あちこちであなたの記事は生き残るかもしれません。

Gnus は取り消すときに『現在』の選択方法を使います。標準の投稿方法を使いたいのであれば、文字接頭引数‘a’ を使ってください(see Section 10.3 [Symbolic Prefixes], p. 272)。

Gnus はCancel-Lock ヘッダー(see Section “ニュースを取り消す” in *The Message Manual*) を使って、あなただけがあなたのメッセージをキャンセルできるようにします。

もし何か間違いをしたのに気付いて、訂正をしたいのであれば、「代替」(*superseding*) 記事を投稿して元記事を置き換えることができます。

元記事のところへ移動して、*S s* を押してください(gnus-summary-supersede-article)。それを普通に送信する前に、記事を好きなように編集することができます。

代替に関しても、取り消しと同じことが当てはまります。こちらの方がもっとよく当てはまるかもしれません: いくつかのサイトは代替を受け付けません。これらのサイトでは、ほとんど同じ記事を二回投稿したようになってしまいます。

もしさっき記事を投稿したばかりですぐに変更したくなった場合、記事が最初にあなたのサイトに現れる前に取り消し/代替をするための巧妙な手段があります。まず、投稿バッファー(*sent ...* のようになっています) に戻ってください。そこにはあなたがちょうど投稿した記事があり、すべてのヘッダーがそのままあります。Message-ID ヘッダーをCancel もしくはSupersedes に変更してください。そして、いつもやっているように単に*C-c C-c* を押して記事を送信してください。前の記事は取り消されるか置き換えられるでしょう。

ちょっと覚えておいてください: 'supersede' (代替) という語の中に'c' は無いということを。

4.6 遅延記事

ときとして、あなたはメッセージの送信を先延ばしにしたいと思うことはありませんか。例えば、あなたが大切なだれかの誕生日を思い出すために、ちょうどその日に届くメッセージを用意したいと思ったとしましょう。gnus-delay パッケージはこれにうってつけです。設定は簡単です:

```
(gnus-delay-initialize)
```

普段はメッセージを送信するのに Message モードで*C-c C-c* コマンドを使いますよね。先延ばしにするには、代わりに*C-c C-j* (gnus-delay-article) を使ってください。そうすると、どのくらい遅らせるかを尋ねてきます。可能な返事は次の通りです:

- 期間。整数と一つの文字で指定します。例えば42d は 42 日遅らせることを意味します。使うことができる文字はm (分)、h (時)、d (日), w (週), M (月) およびY (年) です。
- 日付。YYYY-MM-DD のような形式で指定します。メッセージの送信はその日の特定の時刻(ディフォルトは 8 時) まで遅らせられます。gnus-delay-default-hour も参照してください。
- 時刻。am/pm を含まない 24 時間制の、hh:mm の形式で与えます。送信されるのは今日のその時刻ですが、すでにその時刻を過ぎてしまっていた場合は翌日のその時刻になります。ですから、朝の 10 時に11:15 を指定した場合は 1 時間 15 分後に送信されることになります。しかし9:20 を指定した場合は翌日の時刻を意味します。

gnus-delay-article の動作は、以下の数個の変数に影響されます:

gnus-delay-default-hour
: 特定の日付を指定した場合に、メッセージがその日の何時に送信されるかを与えます。可能な値は 0 から 23 までの整数です。

gnus-delay-default-delay
: ディフォルトの遅延を与える文字列です。前述のどんな形式でも可能です。

gnus-delay-group
: 遅延記事は、ドラフトサーバーのこのグループに期限が来るまで保管されます。たぶんあなたはこれを変更する必要は無いでしょう。ディフォルトの値は"delayed" です。

gnus-delay-header
: それぞれの記事が送信される日時はヘッダーに記録されます。この変数はヘッダー名の文字列です。たぶんあなたはこれを変更する必要は無いでしょう。ディフォルトの値は"X-Gnus-Delayed" です。

送信の先延ばしはこんなふうに行なわれます: gnus-delay-article コマンドで、あなたはどのくらい遅らせるかを指定します。Gnus はメッセージを送信する日時を計算し

てX-Gnus-Delayed ヘッダーに記録し、そのメッセージをnndraft:delayed グループに納めます。

そして、あなたが新着ニュースを取得しようとするときはいつも、Gnus は送信する期限に達した記事をそのグループで探して、それらを送信します。これには関数gnus-delay-send-queue が使われます。ディフォルトではこの関数はgnus-get-new-news-hook に追加されますが、もちろんあなたはこれを変更することができます。おそらくあなたは、ドラフトの送信にデーモンを使いたいと思うのではないでしょうか? それには、デーモンに関数gnus-delay-send-queue を実行せよ、と言うだけで良いのです。

gnus-delay-initialize
: ディフォルトではこの関数はgnus-delay-send-queue のgnus-get-new-news-hook への追加を行ないます。ですが、これは第二オプション引数no-check を受け付けます。もしそれが非-nil だったらgnus-get-new-news-hook は変更されません。第一オプション引数は無視されます。

  例えば(gnus-delay-initialize nil t) は何もしないことを意味します。あなたは遅延記事の送信にデーモンを使いたいのでしょうね。でも、それを設定することを忘れないでくださいね。:-)

4.7 記事に印を付ける

記事に付けられる印はいくつかあります。

記事の「購読度」(うひょーっ、何てすらばやしい造語だ!) を決定する印があります。英字でない文字が一般に「未読」を意味するのに対して、英字の印は一般に「既読」を意味します。

加えて、購読度に影響しない印もあります。

4.7.1 未読記事

以下の印は何らかの方法で記事に未読の(ような) 印を付けます。

'!'
: 可視記事(ticked) として印を付けます(gnus-ticked-mark)。

  「可視記事」とは常に可視状態である記事のことです。おもしろいと思う記事があった場合や、読むのを先に延ばしたいときや、後で返答をしたいときに、普通は可視印を付けます。しかし、記事は期限切れ消去されることもあります(ニュースサーバー上の記事を消去するのはニュースサーバーのソフトウェアで、Gnus 自体は可視記事を期限切れ消去しません) ので、永遠に記事を保存しておきたい場合は、その記事を永続にする必要があります(see Section 4.13 [Persistent Articles], p. 80)。

'?'
: 保留として印を付けます(gnus-dormant-mark)。

  「保留記事」はフォローアップがあったときにだけ概略バッファーに現れます。フォローアップが無いときも表示させたいときは、/ D 命令を使ってください(see Section 4.8 [Limiting], p. 67)。それ以外は(見えるかどうかは別にして)、可視記事(ticked) と似たようなものです。

'SPC'
: 未読として印を付けます(gnus-unread-mark)。

  「未読記事」は今までまったく読まれていない記事のことです。

4.7.2 既読記事

以下のすべての印は記事に既読の印を付けます。

'r' 利用者が手動で*d* 命令もしくはそれに類する手段を使って、既読の印を付けた記事です(gnus-del-mark)。

'R' 実際に読まれた記事(gnus-read-mark)。

'O' 前回のセッションで既読の印を付けて、今は「古く」なってしまった記事。

'K' 削除された印(gnus-killed-mark)。

'X' 削除ファイルによって削除の印が付いた記事(gnus-kill-file-mark)。

'Y' 低すぎるスコアのために既読の印が付いた記事(gnus-low-score-mark)。

'C' キャッチアップによって既読の印が付いた記事(gnus-catchup-mark)。

'G' 取り消された記事(gnus-canceled-mark)。

'Q' まばらに参照された記事(gnus-sparse-mark)。See Section 4.9.1 [Customizing Threading], p. 70.

'M' 重複抑制により既読の印の付いた記事(gnus-duplicate-mark)。See Section 4.30 [Duplicate Suppression], p. 121.

これらのすべての印は、本当にただ記事が既読として印が付いていることを意味するだけです。適応スコアリングをしたときには違ったように解釈されますけれど。

もう一つ、特別な印があります:

'E' 期限切れ消去可能として印の付いた記事(gnus-expirable-mark)。

記事を「期限切れ消去可能」として印を付ける(もしくは、自動的にそのように印を付ける) ことは、普通のグループではあまり意味がありません---利用者はニュース記事の期限による削除を制御していません。しかし、例えばメールグループでは、「期限切れ消去可能」として印の付いた記事は、いつでも Gnus によって削除されることがあります。

4.7.3 他の印

記事が読まれたかどうかには関係しない印がいくつかあります。

- 現在の記事にしおりを挟むことができます。あなたは猫のおしっこの習慣に関する長い論文を読んでいて、それを読み終わる前に晩ご飯を食べに家に帰らなければならなかったとしましょう。そんなとき、記事にしおりを挟むことができます。次にその記事に出くわすと、Gnus はそのしおりのところへ移動するでしょう。See Section 4.7.4 [Setting Marks], p. 64.
- 返信したかフォローアップした(つまり、答えた) すべての記事には、二桁目に'A' の印が付きます(gnus-replied-mark)。
- 転送したすべての記事には、二桁目に'F' の印が付きます(gnus-forwarded-mark)。
- 記事キャッシュに貯められている記事は、二桁目に'*' の印が付きます(gnus-replied-mark)。See Section 4.12 [Article Caching], p. 79.
- (何らかの方法によって; 必ずしも宗教的というわけではなく)『救済された』(原文は saved== 保存された) 記事は、二桁目に'S' の印が付きます(gnus-saved-mark)。

- まだ Gnus で読まれたことがない記事は、二桁目に‘.’ の印が付きます(gnus-unseen-mark)。
- Gnus エージェント(see Section 7.9.1 [Agent Basics], p. 216) を使っているとき、記事は unplugged (ネットワークから切り離されている状態) で見るためにダウンロードされるかもしれません。‘%O’ の仕様を使っていると、それらの記事にはその仕様に‘+’ の印が付きます。(変数gnus-downloaded-mark でどの文字を使うかを制御します。)
- Gnus エージェント(see Section 7.9.1 [Agent Basics], p. 216) を使っているとき、いくつかの記事はダウンロードされていないかもしれません。Unplugged (ネットワークから切り離されている状態) ではそのような記事を見ることができません。‘%O’ の仕様を使っていると、それらの記事にはその仕様に‘-’ の印が付きます。(変数gnus-undownloaded-mark でどの文字を使うかを制御します。)
- Gnus エージェント(see Section 7.9.1 [Agent Basics], p. 216) はいくつかの記事を自動的にダウンロードしますが、自動的にダウンロードされない記事にもダウンロードのための明示的な印を付けることは可能です。そのような明示的に印が付けられた記事には、最初の桁に‘%’ の印が付きます。(変数gnus-downloadable-mark でどの文字を使うかを制御します。)
- ‘%e’ の仕様が使われると、スレッドがあるかどうかの印がgnus-not-empty-thread-mark またはgnus-empty-thread-mark によって、三桁目に付きます。
- 最後に「プロセス印」があります(gnus-process-mark)。いろいろな種類の命令が、プロセス印があるとそれに対して実行されます。例えば*X u* (gnus-uu-decode-uu) は、プロセス印の付いたすべての記事を uudecode して表示します。プロセス印の付いた記事は二桁目に‘#’ があります。

たいていのこれら『購読度と関係無い』印は、ディフォルトでは二桁目に現れることに気付いたでしょう。では、キャッシュされていて、保存されていて、返答した記事にプロセス印を付けた場合は、どうなるのでしょう?

たいしたことはありません。優先順位は次のようになっています: プロセス→ キャッシュ→ 返答済み→ 保存。ですから、ある記事がキャッシュに入っていて返答されていた場合、キャッシュ印が見えるだけで、返答済み印は見えません。

4.7.4 印を付ける

すべての印を付けるための命令は、数値接頭引数を受け付けます。

M c
M-u 現在の記事から、すべての購読度に関する印を消去します(gnus-summary-clear-mark-forward)。要するに、記事に未読の印を付けます。

M t
! 現在の記事に可視記事の印を付けます(gnus-summary-tick-article-forward)。See Section 4.12 [Article Caching], p. 79.

M ?
? 現在の記事に保留記事の印を付けます(gnus-summary-mark-as-read-forward)。See Section 4.12 [Article Caching], p. 79.

M d
d 現在の記事に既読の印を付けます(gnus-summary-mark-as-read-forward)。

D 現在の記事に既読の印を付け、前の行にポイントを移動します(gnus-summary-mark-as-read-backward)。

M k
k 現在の記事と同じ表題を持つすべての記事を既読として印を付け、次の未読記事を選択します(gnus-summary-kill-same-subject-and-select)。

M K
C-k 現在の記事と同じ表題を持つすべての記事を既読として印を付けます(gnus-summary-kill-same-subject)。

M C すべての未読記事に既読の印を付けます(gnus-summary-catchup)。

M C-c グループのすべての記事に---可視記事や保留記事でさえも、既読の印を付けます(gnus-summary-catchup-all)。

M H 現在のグループの、現在位置とそれ以前の記事を既読として印を付けます(gnus-summary-catchup-to-here)。

M h 現在のグループの、現在位置とそれ以降の記事を既読として印を付けます(gnus-summary-catchup-from-here)。

C-w ポイントとマークの間の記事に既読の印を付けます(gnus-summary-mark-region-as-read)。

M V k ディフォルトのスコア(もしくは数値接頭引数) よりも低いスコアの記事を削除します。

M e
E 現在の記事を期限切れ消去可能として印を付けます(gnus-summary-mark-as-expirable)。

M b 現在の記事にしおりを設定します(gnus-summary-set-bookmark)。

M B 現在の記事のしおりを削除します(gnus-summary-remobe-bookmark)。

M V c ディフォルトのスコア(もしくは数値接頭引数) よりも大きいスコアを持つ記事のすべての印を消去します(gnus-summary-clar-above)。

M V u ディフォルトのスコア(もしくは数値接頭引数) よりも大きいスコアを持つすべての記事に可視印を付けます(gnus-summary-tick-above)。

M V m 印の入力を促し、ディフォルトのスコア(もしくは数値接頭引数) よりも大きなスコアを持つすべての記事にその印を付けます(gnus-summary-mark-above)。

変数gnus-summary-goto-unread は印が付けられた後にどのような動作がなされるかを決定します。もしnil でないと、ポイントは次/前の未読記事に移動します。もしnil であると、ポイントは一行上か下に行くだけです。特別な場合として、この変数がnever であると、すべての印を付ける命令と(SPC のような) 他の命令は次の記事が未読であろうが無かろうが次の記事に移動します。ディフォルトはt です。

4.7.5 一般的な印を付けるコマンド

記事に可視の印を付けるコマンド(*!*) が、(印を付けた後で) 次の記事に移動することを望む人がいます。次の未読記事に移動してもらいたい人もいます。さらに、現在の記事に留まっていてもらいたい人もいるでしょう。そして、前の(未読の) 記事に行って欲しい人がいるとはまだ聞いたことはありませんが、そうしたいと思う人も間違いなくいると思います。

この五つの動作を五つの違った印付け命令と掛け算すると、どの命令が何をすべきかの非常に複雑な変数の組を持つことになります。

この窮地を脱するために、Gnus はこれらすべての違ったことをする命令を提供します。これらは概略バッファーの*M M* マップにあります。すべてを見るためには*M M C-h* を入力してください---このマニュアルで一覧を出すには多過ぎます。

これらの命令を直接使うことはできますが、ほとんどの利用者は概略モードのキーマップを交換する方を好むでしょう。例えば、*!* 命令に次の未読記事の代わりに次の記事に移動して欲しいとすると、このようなことができます:

```
(add-hook 'gnus-summary-mode-hook 'my-alter-summary-map)
(defun my-alter-summary-map ()
  (keymap-local-set "!" 'gnus-summary-put-mark-as-ticked-next))
```

もしくは、

```
(defun my-alter-summary-map ()
  (keymap-local-set "!" "MM!n"))
```

4.7.6 プロセス印を付ける

プロセス印は概略バッファーに# として表示され、他のコマンドで処理させる記事に印を付けるために使われます。例えば、四つの記事に印を付けてから* コマンドを使うと、Gnus はそれら四つの記事をキャッシュに入れます。詳しくはSection 10.1 [Process/Prefix], p. 271, をどうぞ。

M P p
現在の記事のプロセス印をトグルします(gnus-summary-mark-as-processable)。
gnus-process-mark-toggle がnil である場合は、現在の記事にプロセス印を付けます。

M P u
M-# もし現在の記事にプロセス印があれば取り除きます(gnus-summary-unmark-as-processable)。

M P U すべての記事からプロセス印を取り除きます(gnus-summary-unmark-all-processable)。

M P i プロセス印の付いている記事とそうでない記事を逆にします(gnus-uu-mark-by-regexp)。

M P R 正規表現に合致するSubject ヘッダーを持つ記事に印を付けます(gnus-uu-mark-by-regexp)。

M P G 正規表現に合致するSubject ヘッダーを持つ記事から印を削除します(gnus-uu-unmark-by-regexp)。

M P r 領域にある記事に印を付けます(gnus-uu-mark-region)。

M P g 領域にある記事から印を削除します(gnus-uu-unmark-region)。

M P t 現在のスレッド(または副スレッド) のすべての記事に印を付けます(gnus-uu-mark-thread)。

M P T 現在のスレッド(または副スレッド) のすべての記事から印を取り除きます(gnus-uu-unamrk-thread)。

M P v 接頭引数よりも大きなスコアを持つすべての記事に印を付けます(gnus-uu-mark-over)。

M P s 現在の一連の記事に印を付けます(gnus-uu-mark-series)。

M P S すでにいくつか印の付いた記事を持つ一連の記事群すべてに印を付けます(gnus-uu-mark-sparse)。

M P a 一連の記事が出てくる順番にそれに属するすべての記事に印を付けます(gnus-uu-mark-all)。

M P b バッファーのすべての記事を現れている順番に印を付けます(gnus-uu-mark-buffer)。

M P k 現在のプロセス印をスタックに積んで、すべての記事を無印にします(gnus-summary-kill-process-mark)。

M P y スタックから前回のプロセス印を取り出して、それを復元します(gnus-summary-yank-process-mark)。

M P w 現在のプロセス印をスタックに積みます(gnus-summary-save-process-mark)。

そして、記事の本文の内容に基づいてプロセス印を付けるやり方については、Section 4.27.2 [Searching for Articles], p. 118, の*&* 命令を参照してください。

4.8 制限をする

現在グループにある記事の一部だけを表示するように概略バッファーを制限できれば便利なことがあります。多くの制限命令が持つ効果は、概略バッファーから少し(もしくは多く)の記事を削除することです。

制限命令はサーバーからすでに取得された記事の一部分に作用します。これらの命令はサーバーに追加の記事を要求しません。

/ /

/ s 概略バッファーをいくつかの表題と合致するものだけに制限します(gnus-summary-limit-to-subject)。接頭引数が与えられると、合致する記事を除外します。

/ a 概略バッファーを何人かの著者に合致するものだけに制限します(gnus-summary-limit-to-author)。接頭引数が与えられると、合致する記事を除外します。

/ R 概略バッファーをいくつかの受信者に合致する記事だけに制限します(gnus-summary-limit-to-recipient)。接頭引数が与えられると、合致する記事を除外します。

/ A 概略バッファーを、その From、To または Cc ヘッダーの内容が与えられたアドレスと合致する記事に制限します(gnus-summary-limit-to-address)。接頭引数が与えられると、合致する記事を除外します。

/ S 概略バッファーを表示されているスレッドに属さない記事だけに制限します(gnus-summary-limit-to-singletons)。接頭引数が与えられると、表示されているスレッドに属する記事だけに制限します。

/ x 「追加」のヘッダーの一つに合致する記事に概略バッファーを制限します(see Section 4.1.2 [To From Newsgroups], p. 50) (gnus-summary-limit-to-extra)。接頭引数が与えられると、合致する記事を除外します。

/ u
x 概略バッファーを既読の印が付いていない記事に制限します(gnus-summary-limit-to-unread)。接頭引数が与えられると、バッファーを完全に未読記事のみに制限します。これは、可視と保留の記事は含まれないということです。

/ m 印を尋ねて、その印が付いている記事に制限します(gnus-summary-limit-to-marks)。

/ t 数値を尋ねて、概略バッファーをその日数より古い(もしくは同じ) 記事に制限します(gnus-summary-limit-to-age)。接頭引数が与えられると、その数値の日よりも新しい記事に制限します。

/ n 概略バッファーを、接頭引数'n' で指定された次の'n' 個の記事に制限します。接頭引数が与えられないと、代わりにプロセス印が付いている記事に制限します。(gnus-summary-limit-to-articles)。

/ w 前の制限をスタックから取り出して、復元します(gnus-summary-pop-limit)。接頭引数を与えられると、すべての制限をスタックから取り出します。

/ . 概略バッファーをまだ読まれたことが無い記事に制限します(gnus-summary-limit-to-unseen)。

/ v 概略バッファーを、与えたスコアと同じか、それより大きなスコアを持つ記事に制限します(gnus-summary-limit-to-score)。

/ p 概略バッファーをdisplay グループパラメーターの述語を満足させるように制限します(gnus-summary-limit-to-display-predicate)。この述語に関する詳細はSection 3.10 [Group Parameters], p. 23, を参照してください。

/ r 概略バッファーを返信した記事だけに制限します(gnus-summary-limit-to-replied)。接頭引数が与えられると、返信した記事以外の記事に制限します。

/ E
M S すべての消去された記事を制限に含めます(gnus-summary-limit-include-expunged)。

/ D すべての保留記事を制限に含めます(gnus-summary-limit-include-dormant)。

*/ **** すべてのキャッシュに入っている記事を制限に含めます(gnus-summary-limit-include-cached)。

/ d すべての保留記事を制限から除外します(gnus-summary-limit-exclude-dormant)。

/ M すべての印付き記事を除外します(gnus-summary-limit-exclude-marks)。

/ T 現在のスレッドのすべての記事を制限に含めます(gnus-summary-limit-include-thread)。

/ c 子記事の無いすべての保留記事を制限から除外します(gnus-summary-limit-exclude-childless-dormant)。

`/ C` すべての除外された未読の記事に既読の印を付けます(`gnus-summary-limit-mark-excluded-as-read`)。接頭引数が与えられると、可視と保留のみの印の記事も既読として印を付けます。

`/ b` 概略バッファーを、ある正規表現に本文が合致する記事だけに制限します(`gnus-summary-limit-to-bodies`)。接頭引数が与えられると、制限を逆にします(訳注: 合致しない記事だけに制限します)。合致するものを探すためにそれぞれの記事を取り込まなければならないので、このコマンドはとても遅いです。

`/ h` この前のコマンドに似ていますが、代わりにこれは、ある正規表現にヘッダーが合致する記事だけに制限します(`gnus-summary-limit-to-headers`)。

以下は制限命令ではありませんが、同様に接頭キー`/` を使います。

`/ N` すべての新しい記事を概略バッファーに挿入します。*back-end*`-get-new-mail` が非`-nil` だったら、新しいメールの到来を調べるということです。

`/ o` すべての古い記事を概略バッファーに挿入します。数値の接頭引数が与えられると、その個数の記事を取り込みます。

4.9 スレッド

Gnus はディフォルトで記事をスレッド表示します。「スレッドにする」とは、ある記事への応答を応答した記事の直後に置く---階層的流儀で、ということです。

スレッドは記事の`References` 欄を調べることによって行なわれます。理想的な世界では、これだけで木を完成させるのに十分なのですが、不運なことに`References` 欄はしばしば壊れているか、時には単に無いことがあります。怪しげなニュースの伝搬は問題を悪化させるので、満足な結果を得るためには他の検出法を採用しなければなりません。過剰な対策法は存在していて、その恐るべき詳細はSection 4.9.1 [Customizing Threading], p. 70, に詳しく書いてあります。

まず、概念の概観です:

根本*(root)* スレッドで一番頂点にある記事です；スレッドの最初の記事です。

スレッド*(thread)*
木のような記事の構成です。

副スレッド*(sub-thread)*
木のような構造の(より) 小さな部分です。

無束縛スレッド*(loose threads)*
記事の期限切れ消去や、根本がすでに前回のセッションで読まれたことにより概略バッファーに表示されない、等の理由により、スレッドはしばしば根本を失います。そのようなときには、普通は多くの副スレッドがあって、本当は一つのスレッドに属しているのですが、根本にはつながっていない、ということになります。こういうスレッドが無束縛スレッドと呼ばれています。

スレッド集め*(thread gathering)*
副スレッドを大きなスレッドに集めようとする試みです。

まばらスレッド*(sparse threads)*
そこではたぶんいくつかの記事が失われてしまったのだろうと『推測された』スレッドのことで、概略バッファーでは空行で表示されます。

4.9.1 スレッドをカスタマイズする

4.9.1.1 無束縛スレッド

`gnus-summary-make-false-root`
: もし`nil` でないと、Gnus はすべてのつながっていない部分木を一つの大きな木にして、頂上にみせかけの根本を作ります。(ちょっと待ってください。頂上に根元(root) ですって? ええ、そうなのです。) つながっていない部分木は本当の根本が期限切れ消去されたか、前回のセッションで根本を読んだり削除したときにできます。

  本当のスレッドが無いときは、Gnus は何かでっち上げをする必要があります。この変数は Gnus が使うべきごまかしの方法を示しています。値としてとることができる四つの候補があります。

  `養子(adopt)`
  : Gnus は孤児になった記事群の最初のものを親にします。この親はすべての他の記事を養子にします。それらの養子記事は、標準の角括弧(‘`[]`’) の代わりに、先の尖った括弧(‘`<>`’) で印が付けられます。これがディフォルトの手段です。

  `みせかけ(dummy)`
  : Gnus は親のふりをするみせかけの概略行を作ります。このみせかけの行はどの本当の記事にも対応しないので、それを選択することは、みせかけの記事の後の最初の本当の記事を選択をするだけになります。みせかけの根本の様式を指定するために、`gnus-summary-dummy-line-format` が使われます。これはたった一つだけのフォーマットの仕様を受け付けます: それは‘`S`’ で、記事の表題です(see Section 10.4 [Formatting Variables], p. 272)。たとえ集めるものが無くても、すべてのスレッドにみせかけの根本を持たせたい場合は、`gnus-summary-make-false-root-always` を`t` に設定してください。

  `空(empty)`
  : Gnus は実際にはどの記事も親にはせず、最初の孤児を除いてすべての孤児の表題欄を単に空のままにします。(実際は`gnus-summary-same-subject` を表題として使います(see Section 4.1 [Summary Buffer Format], p. 47)。)

  `none`
  : まったくどの記事も親にしません。スレッドを集めて、単に順繰りに表示するだけです。

  `nil`
  : 無束縛スレッドを集めません。

`gnus-summary-gather-subject-limit`
: 無束縛スレッドは記事の表題を比較することによって集められます。もしこの変数が`nil` であると、Gnus は無束縛スレッドを一つの大きな超スレッドに集める前に、無束縛スレッドの表題が完全に一致することを要求します。これは、長い表題の行を切り落としてしまう間抜けなニュースリーダーが存在する現状では、あまりに厳しい要求かもしれません。そう思うのなら、この変数を例えば 20 に設定して、表題の最初の 20 文字だけが一致することを要求するようにしてください。この変数を本当に低い数値に設定すると、目についたもののす

べてを Gnus が一つのスレッドに集めるのを見ることになるでしょう。それはあまり有用ではありません。

この変数を特別な値`fuzzy` に設定すると、Gnus は表題の文字列を大雑把に比較するアルゴリズムを使います(see Section 10.17 [Fuzzy Matching], p. 291)。

`gnus-simplify-subject-fuzzy-regexp`
正規表現または正規表現のリストのどちらかです。表題の大雑把な比較を行なうときに、それらに合致する文字列を表題から取り除きます。

`gnus-simplify-ignored-prefixes`
もし`gnus-summary-gather-subject-limit` を 10 くらいに低く設定したならば、この変数を何か意味のあるものに設定することを考えるでしょう:

```
(setq gnus-simplify-ignored-prefixes
      (concat
       "\\`\\[?\\("
       (regexp-opt '("looking"
                     "wanted" "followup" "summary" "summary of"
                     "help" "query" "problem" "question"
                     "answer" "reference" "announce"
                     "How can I" "How to" "Comparison of"
                     ;; ...
                     ))
       "\\)\\s *\\("
       (regexp-opt '("for" "for reference" "with" "about"))
       "\\)?\\]?:?[ \t]*"))
```

この正規表現に合致するすべての語は、二つの表題を比較する前に取り除かれます。

`gnus-simplify-subject-functions`
`nil` でないと、この変数は`gnus-summary-gather-subject-limit` よりも優先されます。この変数は`Subject` の文字列に反復して作用させて簡単にするための、関数のリストである必要があります。

このリストに入れて役に立つような関数は次のようなものです:

`gnus-simplify-subject-re`
前の方にある‘`Re:`’ を取り除きます。

`gnus-simplify-subject-fuzzy`
大雑把な比較ができるように簡単にします。

`gnus-simplify-whitespace`
余分な空白(whitespace) を取り除きます。

`gnus-simplify-all-whitespace`
すべての空白(whitespace) を取り除きます。

もちろん、あなた自身の関数を書くこともできます。

`gnus-summary-gather-exclude-subject`
無束縛スレッド集めは表題だけで行なわれるので、特に‘’ や‘`(none)`’ のような良くある表題のときは、多くの間違いを起こす可能性があります。この状況を

少し良くするために、正規表現gnus-summary-gather-exclude-subject を使うことによって、集める過程においてどんな表題を除外するかを指示することができます。ディフォルトは‘^ *$\\|^(none)$’ です。

gnus-summary-thread-gathering-function
: Gnus はSubject 欄を調べることによってスレッドを集めます。これは、結果的にまったく関係の無い記事が同じ『スレッド』に含まれるかもしれないことを意味し、混乱の元です。代替手段は、合致するものを見つけるためにReferences 欄にあるMessage-ID をすべて調べることです。これは集められたスレッドが関係の無い記事をまったく含まないことを保証しますが、いかれたニュースリーダーで投稿した記事は適切に集められないということでもあります。ペストかコレラかの選択権はあなたにあります。

  gnus-gather-threads-by-subject
  : この関数はディフォルトの収集関数で、排他的にSubject を調べます。

  gnus-gather-threads-by-references
  : この関数は排他的にReferences 欄を調べます。

  References によって集めることを試してみたいのであれば、次のようにすることができます:

```
(setq gnus-summary-thread-gathering-function
      'gnus-gather-threads-by-references)
```

4.9.1.2 スレッドを埋める

gnus-fetch-old-headers
: もしnil でないと、Gnus は古いスレッドをもっと古いヘッダー、すなわち既読の印が付いている記事のヘッダー、を取得することで構築しようとします。できるだけ少ない概略行を表示したいけれど、できるだけたくさんの無束縛スレッドをつなげておきたいときは、この変数をsome か数値に設定してください。もし数値に設定したときは、それより多い追加のヘッダーは取得されません。どちらの場合でも、古いヘッダーの取得は、使っているバックエンドがoverview ファイルを使っている場合だけ動作します。それらのバックエンドは、普通はnntp, nnspool, nnml およびnnmaildir です。スレッドの根本がサーバーによって期限切れ消去されてしまったら、Gnus はどうしようもないことも覚えておいてください。

  この変数はinvisible に設定することもできます。これは視覚的な効果は何もありませんが、*A T* 命令をよく使うのであれば役に立つでしょう(see Section 4.23 [Finding the Parent], p. 109)。

  サーバーは、このいずれをも機能させるためにNOV をサポートしなければなりません。

  この機能は性能に深刻な影響を与え得ます。すべてのローカルにキャッシュされたヘッダーを無視するからです。記事を期限切れ消去しないサーバー(例えばnews.gmane.io) の、あるグループのためにそれをt に設定すると、概略の生成がとても遅くなってしまいます。

`gnus-fetch-old-ephemeral-headers`
: `gnus-fetch-old-headers` と同じですが、一時ニュースグループのためにだけ使われます。

`gnus-build-sparse-threads`
: 古いヘッダーを取得すると遅くなることがあります。この変数を`some` に設定することによって、同じような低賃金の効果を得ることができます。そうすると、Gnus はすべての記事の完全な`References` 欄を見て、同じスレッドに属する記事をつなごうとします。これは、記事がそのスレッドから失われているとGnus が推測したスレッド表示に「ずれ」を残すでしょう。(これらのずれは普通の概略行のように見えます。もしずれを選択すると、Gnus はその当の記事を取得しようとします。) この変数が`t` であると、Gnus はスレッドを補完するのに役立つかどうかを考慮せずに、すべての「ずれ」を表示します。最後に、この変数が`more` であると、Gnus はどこにもつながっていない枝葉のまばらな節を切り落としません。この変数はディフォルトでは`nil` です。

`gnus-read-all-available-headers`
: これはあまり役に立たない、いささかはっきりしない変数です。ニュースではないグループにおいて、概略バッファーを作るためにバックエンドが極めて多くのものを取り込まなければならず、しかも親記事を辿ることができない場合に使うことを想定しています。それは主にウェブに基づいたグループでの場合です。

  そんなグループを使わない場合はディフォルトの`nil` のままにしておくのが無難です。使いたい場合はグループ名に合致する正規表現か、すべてのグループが対象になる`t` にしてください。

4.9.1.3 もっとスレッドを

`gnus-show-threads`
: この変数が`nil` であると、スレッドは作られず、ここにある残りのすべての変数はまったく効果が無くなります。スレッド作りを止めるとグループの選択が少し速くなりますが、記事を読むのがもっと遅く、不便になることは確実です。

`gnus-thread-hide-subtree`
: これが`nil` でないと、すべてのスレッドは概略バッファーが生成されたときに隠れます。

  これは述語指示子であることもできます(see Section 10.13 [Predicate Specifiers], p. 286)。利用できる述語は`gnus-article-unread-p` と`gnus-article-unseen-p` です。

  これは例です:

```
(setq gnus-thread-hide-subtree
      '(or gnus-article-unread-p
           gnus-article-unseen-p))
```

  (これはかなりばかげた例です。なぜならすべてのまだ読まれたことが無い記事は未読でもあるからなのですが、趣旨は汲み取ってください。)

gnus-thread-expunge-below
: この数値より少ない総スコア(gnus-thread-score-function で定義された関数を使って算出されます) を持つすべてのスレッドは消去されます。この変数はディフォルトではnil で、これはどのスレッドも消去されないということです。

gnus-thread-hide-killed
: スレッドを削除すると、この変数がnil でない場合、部分木は隠されます。

gnus-thread-ignore-subject
: ときどき誰かがスレッドの途中で表題を変更することがあります。この変数がnil でないと(これがディフォルトですが)、表題の変更は無視されます。もしnil だと、表題の変更をすると別のスレッドになります。

gnus-thread-indent-level
: これは、それぞれの副スレッドがどれくらい字下げ(indent) されるべきかを決める数値です。ディフォルトは 4 です。

gnus-sort-gathered-threads-function
: とりわけメーリングリストでは、ときとして手元にメールが到着する順番は必ずしもメーリングリストに到着した順番と同じでは無いことがあります。その結果、副スレッドをディフォルトのgnus-thread-sort-by-number で並べ換えると、応答の方がそれの元記事より先に現れてしまうことがあります。グループパラメーターや適切なフック(例えばgnus-summary-generate-hook) でこの変数を代わりの値(例えばgnus-thread-sort-by-date) に設定することによって、そのような場合に、より論理的な副スレッドの順番を生成することができます。

4.9.1.4 低レベルにおけるスレッド作成

gnus-parse-headers-hook
: すべてのヘッダーを解析する前に実行されるフックです。

gnus-alter-header-function
: この変数の値がnil ではなくて関数であると、ヘッダー構造(訳注: 記事の主要なヘッダーの内容を効率良く保持するための Lisp オブジェクト) を変更するために呼ばれます。関数は記事ヘッダーのベクトル(訳注: すなわちヘッダー構造) とともに呼ばれ、それが何らかの方法で変更されます。例えばMessage-ID を体系的な方法で(接頭語などを付け加えることによって) 変更してしまうメールからニュースへのゲートウェイがある場合、この変数を設定することによって、そのMessage-ID を元の意味のあるものに戻すことができます。これは一つの例です:

```
(setq gnus-alter-header-function 'my-alter-message-id)

(defun my-alter-message-id (header)
  (let ((id (mail-header-id header)))
    (when (string-match
           "\\(<[^<>@]*\\)\\.?cygnus\\..*@\\([^<>@]*>\\)" id)
      (mail-header-set-id
       (concat (match-string 1 id) "@" (match-string 2 id))
       header))))
```

訳注: 取得した記事のMessage-ID 欄から、‘@’ の前に付加された‘cygnus.’ で始まる文字列を取り除きます。

もう 1 つの例: Proton Mail ブリッジはメッセージヘッダーのReferences に偽のメッセージ ID を追加するのですが、そのためにスレッドが混乱する可能性があります。それらの偽の ID を削除するには

```
(setq gnus-alter-header-function 'fix-protonmail-references)

(defun fix-protonmail-references (header)
  (setf (mail-header-references header)
        (mapconcat
         #'(lambda (x) (if (string-search "protonmail.internalid" x) "" x))
         (gnus-split-references (mail-header-references header)) " "))
  header)
```

4.9.2 スレッドの命令

T k
C-M-k 現在のスレッド(または副スレッド) のすべての記事に既読の印を付けます(gnus-summary-kill-thread)。もし接頭引数が正であると、代わりにすべての印を取り除きます。接頭引数が負であると、代わりに記事を可視にします。

T l
C-M-l 現在のスレッド(または副スレッド) のスコアを下げます(gnus-summary-lower-thread)。

T i 現在のスレッド(または副スレッド) のスコアを上げます(gnus-summary-raise-thread)。

T # プロセス印を現在のスレッド(または副スレッド) に付けます(gnus-uu-mark-thread)。

T M-# 現在のスレッド(または副スレッド) からプロセス印を取り除きます(gnus-uu-unmark-thread)。

T T スレッド表示を切り替えます(gnus-summary-toggle-threads)。

T s もしあれば、現在の記事の下に隠れているスレッドを表示します(gnus-summary-show-thread)。

T h 現在のスレッド(または副スレッド) を隠します(gnus-summary-hide-thread)。

T S すべての隠されているスレッドを表示します(gnus-summary-show-all-threads)。

T H すべてのスレッドを隠します(gnus-summary-hide-all-threads)。

T t 現在の記事のスレッドをもう一度作り直します(gnus-summary-rethread-current)。これは概略バッファーがスレッド表示されていないときでも動作します。

T ^ 現在の記事を印付きの(もしくは前の) 記事の子記事にします(gnus-summary-reparent-thread)。

*T M-^* 現在の記事を印付きの記事の親記事にします(gnus-summary-reparent-children)。

以下の命令はスレッド移動命令です。これらはすべて数値接頭引数を受け付けます。

T n
C-M-f
M-down 次のスレッドに移動します(gnus-summary-next-thread)。

T p
C-M-b
M-up 前のスレッドに移動します(gnus-summary-prev-thread)。

T d スレッドを下ります(gnus-summary-down-thread)。

T u スレッドを登ります(gnus-summary-up-thread)。

T o スレッドの頂上に移動します(gnus-summary-top-thread)。

スレッドを作成するときに表題を無視すると、当然ながらいくつかの違った表題があるスレッドが出来上がります。そして*T k* (gnus-summary-kill-thread) のような命令を発するときに、全体のスレッドを削除するのではなく、現在の記事と同じ表題を持つ部分だけを削除したいときがあるかもしれません。もしこの発想が良いと思うのであれば、gnus-thread-operation-ignore-subject をいじってみてください。これがnil でないと(それがディフォルトですが)、スレッドの命令を実行しているときに表題は無視されます。これがnil だったら、同じスレッドにある異なる表題を持つ記事は、そのとき行なう操作の対象に含まれません。この変数がfuzzy であると、大雑把な比較によって等しいと判定される表題を持つ記事だけが対象に含まれます(see Section 10.17 [Fuzzy Matching], p. 291)。

4.10 並べ替え

概略でスレッドの表示を使っているのであれば、gnus-thread-sort-functions を設定することによってスレッドを並べ替えることができます。この変数の値は単独の関数、関数のリスト、または関数と(関数でないもの)の要素を含むリストであることができます。

ディフォルトでは並べ替えは記事番号に基づいて行なわれます。すでに用意されている並べ替え述語関数はgnus-thread-sort-by-number, gnus-thread-sort-by-author, gnus-thread-sort-by-recipient, gnus-thread-sort-by-subject, gnus-thread-sort-by-date, gnus-thread-sort-by-score, gnus-thread-sort-by-most-recent-number, gnus-thread-sort-by-most-recent-date, gnus-thread-sort-by-newsgroups, gnus-thread-sort-by-random およびgnus-thread-sort-by-total-score です。

それぞれの関数は二つのスレッドをとり、最初のスレッドがもう一方より先に並べ替えられるべきであればnil でない値を返します。実際の並べ替えは、普通それぞれのスレッドの根本だけを調べることによって行なわれることに気を付けてください。ただしgnus-thread-sort-by-most-recent-number およびgnus-thread-sort-by-most-recent-date は、この規則に従いません。

二つ以上の関数を使う場合、並べ替えの第一の鍵はリストの最後の関数でなければなりません。並べ替え関数のリストのなるべく先頭に、おそらく常にgnus-thread-sort-by-number を含めておくべきでしょう。これは、他の並べ替えの基準が等しいスレッドが、記事番号の登り順に表示されることを保証します。

スコアの逆順、表題、そして最後に番号、の順に並べ替えたいのであれば、次のようにできます:

```
(setq gnus-thread-sort-functions
      '(gnus-thread-sort-by-number
        gnus-thread-sort-by-subject
        (not gnus-thread-sort-by-total-score)))
```

最大のスコアを持つスレッドが、最初に概略バッファーに表示されます。スレッドが同じスコアの場合は、英字順に並び替えられます。同じスコアと表題を持つスレッドは番号で並べ替えられ、(普通は) 記事が到着した順番になります。

スコア、到着の逆順に並べ替えたいのであれば、次のようにできます:

```
(setq gnus-thread-sort-functions
      '((not gnus-thread-sort-by-number)
        gnus-thread-sort-by-score))
```

ディフォルトでは、スレッドはそのサブスレッドとともにgnus-thread-sort-functionsの値にしたがってソートされますが、gnus-subthread-sort-functions をカスタマイズすることによって、サブスレッドをソートする順序を独自に定義することができます。例えば概略バッファー上で、親スレッドはスコアの高い順から低い順にソート、しかしサブスレッドはスコアとは無関係に古いものから新しいものへの順で時系列にソートされたままにしておく、というようなことができます。

変数gnus-thread-score-function (ディフォルトは+) に設定されている関数は、スレッドの総スコアを計算するために用いられます。役立つ関数はmax, min, もしくは二乗、もしくはあなたの好奇心をくすぐるような何かでしょう。

何か変な理由でスレッド表示を使っていないのなら、変数gnus-article-sort-functionsをいじくる必要があります。これはgnus-thread-sort-functions と非常に似ていますが、記事の比較には少々違った関数を使います。使用可能な並べ替え述語関数はgnus-article-sort-by-number, gnus-article-sort-by-author, gnus-article-sort-by-subject, gnus-article-sort-by-date, gnus-article-sort-by-newsgroups, gnus-article-sort-by-random およびgnus-article-sort-by-score です。

スレッドを使っていない概略の表示を表題で並べ替えたいのであれば、次のようなことをすることができます:

```
(setq gnus-article-sort-functions
      '(gnus-article-sort-by-number
        gnus-article-sort-by-subject))
```

gnus-parameters を介することによって、グループによって異なる並べ替えを定義することができます。See Section 3.10 [Group Parameters], p. 23.

4.11 非同期記事取得

遠くにあるNNTP サーバーからニュースを取得していると、ネットワークの遅延が記事を読むことを嫌な仕事にしてしまうかもしれません。*n* を押してから次の記事が現れるまで、しばらく待たなければなりませんものね。どうして前の記事を読んでいる間に Gnus が先行して記事を取得してくれないのでしょうか? なぜできないんでしょう、本当に。

まず警告しておきましょう。非同期で記事を取得、特に Gnus がそれを行なう場合には、いくつかの落とし穴があります。

例えば、あなたは短い記事 1 を読んでいて、記事 2 はとても長くて、あなたはそれを読むことには興味が無いとしましょう。Gnus はこのことはわからないので、先行して記事 2 を取得します。あなたは記事 3 を読むことにしますが、Gnus は記事 2 を取得している最中なので、接続は封鎖されています。

この状況を避けるために、Gnus はサーバーに二つ(二まで数えてください) の接続を張ります。これはあまり良いことではないと考える人もいるでしょうが、私には実際の代替手段が見つからないのです。余分な接続を立ち上げるためにはいくばくかの時間がかかるので、Gnus の起動は遅くなります。

Gnus はあなたが読むであろう記事よりもたくさんの記事を取得します。これは記事の先行取得を使わないときよりも、あなたのマシンとNNTP サーバー間の接続にもっと負荷をかけることになるでしょう。サーバー自身にももっと負荷がかかるようになります---余分な記事の要求と、余分な接続によって。

はい、本当はこのようなことをすべきで無いことがこれで分かったでしょう... 本当にそうしたいと思わない限りは。

やり方です: `gnus-asynchronous` を`t` に設定してください。それ以外の諸々のことは自動的に行なわれます。

`gnus-use-article-prefetch` を設定することによって、どれくらいの記事を先に取得するべきかを操作することができます。これはディフォルトでは 30 で、グループの記事を読んでいるときに、バックエンドが次の 30 通の記事を先行取得するということです。この変数が`t` であると、バックエンドは取得できるすべての記事を際限なく先行取得しようとします。これが`nil` であると、先行取得は行なわれません。

おそらく先行取得をしたくない記事がいくつかあるでしょう---例えば既読記事です。変数`gnus-async-prefetch-article-p` は記事が先に取得されるかどうかを制御します。この変数に設定される関数は、問題の記事を先行取得するのであれば`nil` でない値を返さなければなりません。ディフォルトの関数は`gnus-async-unread-p` で、これは既読記事には`nil` を返します。この関数は記事のデータ構造を唯一の引数として呼ばれます。

例えば、100 行よりも短い未読記事だけを先に取得したいのであれば、次のようにできます:

```
(defun my-async-short-unread-p (data)
  "Return non-nil for short, unread articles."
  (and (gnus-data-unread-p data)
       (< (mail-header-lines (gnus-data-header data))
          100)))

(setq gnus-async-prefetch-article-p 'my-async-short-unread-p)
```

これらの関数は何度も何度も呼ばれるので、Gnus を遅くしすぎないように、短く簡潔であるのが好ましいです。このようなものをバイトコンパイルするのは、おそらく良い着想でしょう。

記事を先行取得した後に、この`gnus-async-post-fetch-function` が呼ばれます。そのときバッファーは取り込んだ記事の領域に狭められています。有用な値は`gnus-html-prefetch-images` で、これはその記事が参照する画像を先行取得および保存して、その記事を読むときにそれらの取得を待たなくても良いようにします。HTML メッセージが外部参照の画像を持っているときに便利です。

記事は非同期バッファーから遅かれ早かれ削除されなければなりません。gnus-prefetched-article-deletion-strategy はいつ記事を削除するかを指定します。これは以下の要素を含むリストです:

`read` 記事が読まれたときに削除します。

`exit` グループを抜けたときに記事を削除します。

ディフォルトの値は(read exit) です。

4.12 記事のキャッシュ

非常に 遅いNNTP 接続を使っているのならば、記事をキャッシュすることを考えても良いでしょう。それをすると、それぞれの記事はあなたのホームディレクトリーの下にローカルに溜められます。もう感付いたかもしれませんが、これは i ノードを非常に速く食いつぶすだけでなく、巨大な ディスクスペースを食う可能性があります。それはあなたにウォッカの中で泳ぐようなめまいを起こさせるでしょう。

でも注意深く使われれば、それは記事を保存する、より楽な方法になり得ます。

キャッシュを実行させるにはgnus-use-cache をt に設定してください。ディフォルトでは、すべての可視または保留として印の付いている記事はローカルのキャッシュ(gnus-cache-directory) に複写されます。このキャッシュが平らな構造か階層的であるかは、通常通り、変数gnus-use-long-file-name で制御されます。

可視記事か保留記事を再選択した場合は、サーバーの代わりにキャッシュから取得されます。キャッシュにある記事は期限切れ消去されないので、記事をそれらが属するところに居続けさせている間、それらを保存する方法としてこれは役立つかもしれません。保存したいすべての記事に保留の印を付けるだけで、後は心配無用です。

記事に既読の印が付いたときに、それはキャッシュから削除されるのでしょうか。

記事をキャッシュに入れたりキャッシュから削除することは、変数gnus-cache-enter-articles およびgnus-cache-remove-articles によって制御されます。これらは両方ともシンボルのリストです。前者はディフォルトでは(ticked dormant) で、可視記事と保留記事はキャッシュに入れられます。後者はディフォルトでは(read) で、既読の印が付いた記事はキャッシュから削除されます。おそらくこれら二つのリストに含まれるシンボルはticked, dormant, unread およびread でしょう。

それでは、大規模な記事の取得と格納は、どこで関係してくるのでしょうか。gnus-jog-cache 命令は、すべての購読グループに対して、すべての未読記事を要求し、スコアを付け、キャッシュに保存します。この命令をいつもいつもいつもいつも使うのは、1) NNTP サーバーとの接続が本当に本当に本当に遅くて、2) 本当に本当に本当に巨大なディスクを持っているときだけにするべきです。これは真面目に言っています。ダウンロードされる記事の数を控える一つの方法は、欲しくない記事のスコアを低くして、それらに既読の印を付けることです。そうすれば、それらはこの命令ではダウンロードされません。

すべてのグループではキャッシュをしたくないというのは良くあることです。例えばnnml のメールがホームディレクトリーにあるのなら、それをホームディレクトリーの別の場所にキャッシュするのは意味がありません。二倍の容量を使う方が良いと思うのでなければ。

キャッシュを制限するには、gnus-cacheable-groups を例えば‘^nntp’ のようなキャッシュするグループの正規表現に設定するか、または正規表現gnus-uncacheable-groups を例えば‘^nnml’ に設定してください。両方の変数ともにディフォルトはnil です。もしグループが両方の変数に合致すると、そのグループはキャッシュされません。

キャッシュは、それがどの記事を含んでいるかの情報を、そのアクティブファイル(gnus-cache-active-file) に格納します。このファイル(もしくはキャッシュの他の部分) が何らかの理由でぐちゃぐちゃになってしまった場合、Gnus はものごとを正しくするための二つのコマンドを提供します。*M-x gnus-cache-generate-nov-databases* はすべてのNOV ファイルを(再) 作成し、*M-x gnus-cache-generate-active* はアクティブファイルを(再) 作成します。

gnus-cache-move-cache コマンドは、すべてのgnus-cache-directory をどこか別の場所に移動します。あなたはどこに移動させるかを尋ねられます。それってかっこいいでしょ?

4.13 永続記事

記事のキャッシュと近い関係にあるものに「永続記事」があります。実際それはキャッシュを見るための別の方法で、私に言わせればはるかに役に立ちます。

例えば、ニュースグループを読んでいて、永久に秘蔵しておく価値のある宝石に出会ったとしましょう。普通はそれをファイルに保存します(多くの保存命令の一つを使って)。問題は、単にあの、嫌なだけです。理想的には、記事はグループで見つけた場所に永遠に残っていることが好ましいでしょう。ニュースサーバーにおける期限切れ消去には影響されないで。

これが「永続記事」です---記事は削除されません。それは普通のキャッシュ命令を使って実装されていますが、永続記事の管理をするために二つの明示的な命令を使います:

`*` 現在の記事を永続にします(gnus-cache-enter-article)。

`M-*` 現在の記事を永続記事から取り除きます(gnus-cache-remove-articles)。これは普通は記事を削除します。

これらの命令は両方ともプロセス/接頭引数の習慣を理解します。

永続記事にだけ興味があるのなら、可視記事(やその他のもの) がキャッシュに入るのを避けるために、gnus-use-cache をpassive に設定するのが良いでしょう:

```
(setq gnus-use-cache 'passive)
```

4.14 粘着記事

記事を選択するとき、変数gnus-single-article-buffer の値によっては現在の記事バッファーが再利用されます。それがnil 以外の値だと、すべての記事が同じ記事バッファーを再利用しますが、nil だった場合はグループ毎に独自の記事バッファーを持ちます。

このことは、あるグループで同時に一つより多い記事バッファーを持つことはできないことを意味します。でも、時には今度のクリスマス・パーティーの段取りのために、お母さん、お父さん、叔母さん、叔父さん、さらに 17 人のいとこから最近届いたすべての電子メールを表示したいこともあるでしょう。

こんなときに粘着記事が役に立ちます。粘着記事バッファーは原理的には普通の記事バッファーなのですが、他の記事を選択しても再利用されません。記事を粘着質にするには、このコマンドを使ってください:

A S 現在の記事を粘着質にします。接頭引数付きで呼ぶと、この粘着記事バッファーの名前を尋ねます。

粘着記事バッファーを閉じるには、次のコマンドを使ってください:

q この粘着記事バッファーを、すべてのバッファーのリストの最後尾に置きます。

k この粘着記事バッファーを削除します。

すべての粘着記事バッファーを削除するには、このコマンドを使えば良いでしょう:

`gnus-kill-sticky-article-buffers` *ARG* [関数]
すべての粘着記事バッファーを削除します。接頭引数を与えると、確認を求めます。

4.15 記事のバックログ

回線が遅いために、キャッシュを使うという発想があまり魅力的ではないとき(実際そうなのですが)、「バックログ」に切り替えることによって状況を何とかすることができます。これはすでに読んだ記事を再取得しなくても良いように、すでに読んだ記事を Gnus が一時保存しておくところです。これはもちろん、あなたに最近読んだ記事を再び選択する癖があるときだけ役立ちます。絶対にそれをしない人にとっては、バックログを on にすることは Gnus を少し遅くし、メモリーの使用量をいくらか増やすだけのことです。

`gnus-keep-backlog` を数値*n* に設定すると、Gnus は最大で*n* 個の古い記事を後の再取得のためにバッファーに溜めておきます。この変数が`nil` ではなく、数値でもない場合、Gnus はすべて の既読記事を蓄えます。それは Emacs が爆発するまで際限なく膨れ上がって、マシンがあなたもろとも落ちてしまうということです。私はみなさんがいつも注意を怠らないようにするために、ここに書き加えました。

ディフォルト値は 20 です。

4.16 記事の保存

Gnus はたくさんの方法で記事を保存することができます。以下のものは非常に率直な方法(すなわち記事が保存する前にほとんど何もなされない) で記事を保存するための説明です。異なる手続き(uudecode, unshar) のためには`gnus-uu` を使うのが良いでしょう(see Section 4.17 [Decoding Articles], p. 86)。

ここに列挙されているコマンドは対象がファイルです。ディレクトリー名('`/`' で終わるもの) を指定すると、その下にあるファイルが対象になります。グループに保存したい場合は*B c* (`gnus-summary-copy-article`) コマンドを参照してください(see Section 4.26 [Mail Group Commands], p. 114)。

`gnus-save-all-headers` が`nil` でないと、Gnus は記事を保存する前に不要なヘッダーを消去しません。

もし上記の変数が`nil` であると、正規表現`gnus-saved-headers` に合致するすべてのヘッダーが残される一方、残りのものは保存する前に削除されます。

O o
o ディフォルトの記事を保存する手段を用いて現在の記事を保存します(`gnus-summary-save-article`)。

O m 現在の記事を Unix メール(mbox) ファイルに保存します(`gnus-summary-save-article-mail`)。

O r 現在の記事を Rmail の様式で保存します(`gnus-summary-save-article-rmail`)。これは Emacs 23 から mbox になります。旧バージョンでは Babyl でした。

O f 現在の記事を普通のファイル(plain file) 様式で保存します(gnus-summary-save-article-file)。

O F 現在の記事を普通のファイル様式で保存し、以前のファイルの内容を上書きします(gnus-summary-write-article-file)。

O b 現在の記事の本文を普通のファイル様式で保存します(gnus-summary-save-article-body-file)。

O h 現在の記事を mh のフォルダーの様式で保存します(gnus-summary-save-article-folder)。

O v 現在の記事を VM フォルダーに保存します(gnus-summary-save-article-vm)。

O p
| 現在の記事をパイプに保存します。うーんと、あのぉ、私が言おうとしていることは---現在の記事をプロセスにパイプするということです(gnus-summary-pipe-output)。シンボル接頭引数(see Section 10.3 [Symbolic Prefixes], p. 272)が与えられると、パイプへの出力に完全なヘッダーを含めます。シンボル接頭引数r は特別で、それはすべてのヘッダーを含む生の(デコードされていない)記事をパイプに送ります。gnus-summary-pipe-output-default-command 変数をディフォルトのコマンドと引数を含む文字列に設定することができます(ディフォルトはnil)。

O P 現在の記事を muttprint に保存します。つまり、外部プログラムMuttprint (https://muttprint.sourceforge.net/) を使って記事を印刷するということです。プログラム名と使用するオプションは、変数gnus-summary-muttprint-program で指定されます。(gnus-summary-muttprint)。

すべてのこれらの命令はプロセス/接頭引数の習慣を使います(see Section 10.1 [Process/Prefix], p. 271)。もしこれらの関数を使ってたくさんの記事を保存した場合、それぞれのすべての記事に対してファイル名の入力を要求されることに飽き飽きするでしょう。入力を求める動作は変数gnus-prompt-before-saving によって制御されます。これはディフォルトではalways で、あなたが嫌な思いを味わっている、過剰な入力要求をします。代わりにこの変数をt に設定すると、保存するそれぞれの一連の記事に対して一回だけ入力を要求します。本当に Gnus にすべての判断を任せてしまいたいのであれば、この変数をnil にすることさえできます。そうすれば、記事を保存するためのファイルを促されることはありません。Gnus は単純にすべての記事をディフォルトのファイルに保存します。

Gnus を思い通りに動作させるために、変数gnus-default-article-saver をカスタマイズすることができます。下の八つの既製の関数を使うことができ、また自分自身の関数を作ることもできます。

gnus-summary-save-in-rmail
これがディフォルトで、Rmail パッケージで使われているものです。Emacs 23 から Rmail は標準の mbox 様式を使うようになりました。それ以前には*Babyl* 様式というものが使われていました。そのため、既存の Babyl ファイルに追加するのでなければ、Emacs 23 からこのコマンドは mbox 様式で書きます。古いバージョンの Emacs では、常に Babyl 様式を使います。変数gnus-ramil-save-name に設定されている関数を、記事を保存するファイルの名前を取得するために使います。ディフォルトはgnus-plain-save-name です。

`gnus-summary-save-in-mail`
: Unix メール(mbox) ファイルに保存します。変数`gnus-mail-save-name` に設定されている関数を、記事を保存するファイルの名前を取得するために使います。ディフォルトは`gnus-plain-save-name` です。

`gnus-summary-save-in-file`
: 記事を通常のファイルの後に追加します。変数`gnus-file-save-name` に設定されている関数を、記事を保存するファイルの名前を取得するために使います。ディフォルトは`gnus-numeric-save-name` です。

`gnus-summary-write-to-file`
: 記事をストレートに通常のファイルに保存します。そのファイルが存在していたら上書きされます。変数`gnus-file-save-name` に設定されている関数を、記事を保存するファイルの名前を取得するために使います。ディフォルトは`gnus-numeric-save-name` です。

`gnus-summary-save-body-in-file`
: 記事の本文を通常のファイルの後に追加します。変数`gnus-file-save-name` に設定されている関数を、記事を保存するファイルの名前を取得するために使います。ディフォルトは`gnus-numeric-save-name` です。

`gnus-summary-write-body-to-file`
: 記事の本文をストレートに通常のファイルに保存します。そのファイルが存在していたら上書きされます。変数`gnus-file-save-name` に設定されている関数を、記事を保存するファイルの名前を取得するために使います。ディフォルトは`gnus-numeric-save-name` です。

`gnus-summary-save-in-folder`
: MH ライブラリーの`rcvstore` を使って、記事を MH フォルダーに保存します。変数`gnus-folder-save-name` に設定されている関数を、記事を保存するファイルの名前を取得するために使います。ディフォルトは`gnus-folder-save-name` ですが、`gnus-Folder-save-name` も使うことができて、こちらは先頭が大文字、残りが小文字になった名前を作ります。

`gnus-summary-save-in-vm`
: 記事を VM フォルダーに保存します。この設定を使うためには VM メールリーダーが必要です。

`gnus-summary-save-in-pipe`
: 記事をシェルコマンドにパイプします。この関数は二つの引数 COMMAND および RAW を受け付けます(無くても構いません)。有効な COMMAND の値は次の通りです:

  - 文字列
    実行可能なコマンド名と、もしあれば引数です。
  - `nil`
    ミニバッファーでコマンドを入力します。
  - シンボル`default`
    `gnus-summary-pipe-output-default-command` 変数が持っている値、または最後に保存のために使われたコマンドで置き換えられます。

RAW にnil 以外の値を与えると:decode および:headers 属性(後述) が無視され、すべてのヘッダーを含む生の(デコードされていない) 記事がパイプに渡されます。

それぞれの関数シンボルは以下の属性(property) を持つことができます:

:decode nil ではない値が設定されているとデコードした記事を保存します。gnus-summary-save-in-file、gnus-summary-save-body-in-file、gnus-summary-write-to-file、gnus-summary-write-body-to-file およびgnus-summary-save-in-pipe でだけ、これを設定する意義があります。

:function
記事をファイルに上書するのではなく、追加するための代わりの関数を指定します。これを設定すると、複数の記事を一度に保存するときにgnus-prompt-before-saving がt に束縛され、すべての記事が単一のファイルに保存されます。gnus-summary-write-to-file およびgnus-summary-write-body-to-file でだけ、これを設定する意義があります。

:headers 保存されるヘッダーを指定する変数のシンボルをこれで設定します。省略された場合はgnus-save-all-headers とgnus-saved-headers が、どのヘッダーを保存するかを制御します。

これらのすべての関数は最後の一つを除いて、環境変数SAVEDIR によって初期化されるgnus-article-save-directory に記事を保存します。これはディフォルトでは~/News/ です。

上で述べたように、記事を保存するためのファイルの適切な名前を見つけるために、それらは違った関数を用います。以下は名前を生成するために使うことができる関数のリストです:

gnus-Numeric-save-name
~/News/Alt.andera-dworkin/45 のようなファイル名。

gnus-numeric-save-name
~/News/alt.andera-dworkin/45 のようなファイル名。

gnus-Plain-save-name
~/News/Alt.andera-dworkin のようなファイル名。

gnus-plain-save-name
~/News/alt.andera-dworkin のようなファイル名。

gnus-sender-save-name
~/News/larsi のようなファイル名。

連想リストgnus-split-methods に正規表現を放り込むことによって、Gnus に記事を保存する場所をほのめかすことができます。例えば Gnus に関連する記事をgnus-stuff ファイルに、VM に関連する記事をvm-stuff ファイルに保存したければ、この変数を以下のようにすれば良いでしょう:

```
((("^Subject:.*gnus\\|^Newsgroups:.*gnus" "gnus-stuff")
 ("^Subject:.*vm\\|^Xref:.*vm" "vm-stuff")
 (my-choosing-function "../other-dir/my-stuff")
 ((equal gnus-newsgroup-name "mail.misc") "mail-stuff"))
```

これはそれぞれの要素が、二つの要素---「合致」と「ファイル」を持つリストであるリストであるということがわかります。合致は文字列(この場合は記事のヘッダーに合致する正規表現として使われます)、シンボル(グループ名を引数として、関数として呼ばれます)およびリスト(これは評価(eval) されます) のどれかであることができます。これらの動作の一つでもnil でない結果を返すと、入力を求めるときのディフォルトとして「ファイル」が使われます。加えて、呼ばれた関数か式が文字列か文字列のリストを返したときは、演算の結果自体が使われます。

基本的には、現在の記事を保存するのに使われる可能性のあるファイル名のリストを手に入れることになります。(すべての『合致』が使われます。) そして、実際に名前として使いたいものの入力を促されますが、その際、この変数を適用して得られた結果が、ファイル名を補完するときの候補になります。

この変数はディフォルトでは((gnus-article-archive-name)) で、これは Gnus が保存する記事のArchive-name 行を調べて、それをファイル名の候補として使います。

これはファイル名を多少きれいにする関数の例です。'nnml:mail.whatever' のようなメールグループがたくさんあるとすると、保存するためのファイル名を作る前にそれらのグループ名の最初の方を切り落とす必要があるかもしれません。次の物はまさにそれをします:

```
(defun my-save-name (group)
  (when (string-match "^nnml:mail." group)
    (substring group (match-end 0))))

(setq gnus-split-methods
      '((gnus-article-archive-name)
        (my-save-name)))
```

最後に、gnus-use-long-file-name という変数があります。これがnil であると、すべての上記の関数はグループ名のすべてのピリオド('.') をスラッシュ('/') で置き換えます---つまり、すべてのファイルを一番上のディレクトリーに置くのではなく、それらの関数が階層的なディレクトリーを生成するということです(~/News/alt.andrea-dworkin ではなく~/News/alt/andrea-dworkin のように)。たいていのシステムにおいて、この変数のディフォルトはt です。しかし、歴史的な理由によって Xenix と usg-unix-v マシンではnil がディフォルトになります。

この関数は削除とスコアのファイル名にも影響します。この変数がリストで、そのリストがnot-score という要素を含んでいると、長いファイル名はスコアファイルには使われません。そのリストがnot-save という要素を含んでいると、保存するときに長いファイル名は使われません。また、そのリストがnot-kill という要素を含んでいると、長いファイル名は削除ファイルには使われません。

記事をスプールのような階層に保存したい場合は、次のようにしてください。

```
(setq gnus-use-long-file-name '(not-save)) ; 階層にする
(setq gnus-default-article-saver
      'gnus-summary-save-in-file) ; エンコードしない
```

そうしたならば、o で記事を保存するだけです。すると、階層をnneething 一時グループによって読むことができます---グループバッファーでG D をタイプして、一番上のディレクトリー(~/News/) を引数として渡してください。

4.17 記事のデコード

ときどき利用者は何らかの方法でエンコードされた記事(もしくは一連の記事群) を投稿します。Gnus はそれらをデコードすることができます。

> 訳注: この章では、複数に分割して送信された一つの巨大な記事を、再び一つにまとめ上げてデコードする処理について説明しています。現在では、そのような分割送信をメールサーバーが受け付けない等の理由によって、ほとんど目にすることはありません。分割して送信しないことを確実にするには、以下の設定を行なってください(see Section “メール変数” in *The Message Manual*):
>
> ```
> (setq message-send-mail-partially-limit nil)
> ```

これらすべての関数はプロセス/接頭引数の習慣(see Section 10.1 [Process/Prefix], p. 271) を、『一つの記事』を『一つの群』と解釈する拡張をして、どの記事に操作をするかを見つけるために使います。Gnus は自分自身でどの記事がその群に属しているかを判断し、すべての記事をデコードして、その結果のファイルを展開/表示/保存することができます。

Gnus は以下の簡単な規則に則ってどの記事が群に属するのかを推測します: 表題は行の最後の二つの数字を除いて(ほとんど) 同じである必要があります。(空白は大体無視されますが。)

例えば: ‘`cat.gif (2/3)`’ というような表題を選ぶと、Gnus は正規表現‘`^cat.gif ([0-9]+/[0-9]+).*$`’ に合致するすべての記事を見つけようとします。

‘`cat.gif (2/3) Part 6 of a series`’ のような標準でない表題はどの自動表示命令によっても適切に認識されないため、手で記事に`#` の印を付けなければなりません。

4.17.1 uuencode された記事

X u 現在の群を uudecode します(`gnus-uu-decode-uu`)。

X U 現在の群を uudecode して保存します(`gnus-uu-decode-uu-and-save`)。

X v u 現在の群を uudecode して、表示します(`gnus-uu-decode-uu-view`)。

X v U 現在の記事を uudecode して、表示して保存します(`gnus-uu-decode-uu-and-save-view`)。

これらはすべて、プロセス印が付けられた記事に対して反応するということを覚えておいてください。例えばニュースグループ全体をデコードして保存したいのであれば、例によって*M P* a (`gnus-uu-mark-all`) に続いて*X U* (`gnus-uu-decode-uu-and-save`) を実行してください。

このすべては、白日の下にいちいちキーを打っていたGNUS 4.1 のときの`gnus-uu` の動作とはまったく違っています。一般にこの版の`gnus-uu` は、何かの方法(see Section 4.7.6 [Setting Process Marks], p. 66) で記事に印を付け、それから*X u* を押すことを前提としています。

注意: 定数`gnus-uu-notify-files` (値が‘`[Cc][Ii][Nn][Dd][Yy][0-9]+.\\(gif\\|jpg\\)`’ にハードコードされています) に合致する名前を持つ記事をデコードしようとすると、あなたが問題の記事を今まさに見たことをバラすために、`gnus-uu` は自動的に‘`comp.unix.wizards`’ に記事を投稿します。この機能を使わないようにすることはできません(訳注: そんな Cindy Crawford 嬢の写真がニュースで大量に流れていた、まだ WWW がロクに普及していなかった時代の産物です)。

4.17.2 シェルアーカイブ

シェルアーカイブ(『shar ファイル』) はソースを配布するための人気のある方法でしたが、今日ではそんなに使われていません。とにかくこれらを扱うための命令があります:

X s 現在の群を解凍します(gnus-uu-decode-unshar)。

X S 現在の群を解凍して保存します(gnus-uu-decode-unshar-and-save)。

X v s 現在の群を解凍して表示します(gnus-uu-decode-unshar-view)。

X v S 現在の群を解凍し、表示して保存します(gnus-uu-decode-unshar-and-save-view)。

4.17.3 ポストスクリプトファイル

X p 現在のポストスクリプト群を展開します(gnus-uu-decode-postscript)。

X P 現在のポストスクリプト群を展開して保存します(gnus-uu-decode-postscript-and-save)。

X v p 現在のポストスクリプト群を表示します(gnus-uu-decode-postscript-view)。

X v P 現在のポストスクリプト群を表示して保存します(gnus-uu-decode-postscript-and-save-view)。

4.17.4 他のファイル

X o 現在の記事群を保存します(gnus-uu-decode-save)。

X b 現在の記事群を binhex で解凍します(gnus-uu-decode-binhex)。これはまだ実際には動作しません。

X Y 現在の記事群を yEnc でデコードして保存します(gnus-uu-decode-yenc)。

4.17.5 デコードのための変数

形容詞です。動詞ではありません。

4.17.5.1 規則変数

Gnus はファイルをどうやって表示するかを決めるために「規則変数」を使います。これらの変数はすべて以下のような様式です。

```
(list '(regexp1 command2)
      '(regexp2 command2)
      ...)
```

gnus-uu-user-view-rules
: この変数はファイルを表示するときに最初に調べられます。例えば、もし.au音響ファイルを変換するためにsox を使いたいときは、次のように設定することができます:

```
(setq gnus-uu-user-view-rules
      (list '("\\\\.au$\" "sox %s -t .aiff > /dev/audio")))
```

gnus-uu-user-view-rules-end
: この変数は Gnus が利用者とディフォルトの表示規則から合致するものを見つけることができなかったときに調べられます。

`gnus-uu-user-archive-rules`
: この変数はアーカイブを展開するときにどの命令が使われるべきかを決めるために使うことができます。

4.17.5.2 他のデコードのための変数

`gnus-uu-grabbed-file-functions`
: これは関数のリストです。すぐにファイルを移動したり表示することを可能にし、何かができるようになる前にすべてのファイルがデコードされるのを待つ必要が無いように、それぞれのファイルのデコードに成功した直後にそれらの関数が呼ばれます。このリストに入れることができる既製の関数は以下の通りです:

  `gnus-uu-grab-view`
  : ファイルを表示します。

  `gnus-uu-grab-move`
  : ファイルを移動します(もし保存関数を使っているのであれば)。

`gnus-uu-be-dangerous`
: デコードの最中に異常な状況が起こったときに何をするかを指定します。もし`nil` であると、できるだけ保守的になります。もし`t` であると、動作しないものは無視して、現存するファイルを上書きします。その他の場合は、それぞれのときに尋ねます。

`gnus-uu-ignore-files-by-name`
: この正規表現に合致する名前のファイルは表示されません。

`gnus-uu-ignore-files-by-type`
: この変数に合致するMIME の型を持つファイルは表示されません。Gnus はファイル名に基づいて型を推測していることに注意してください。`gnus-uu` は(まだ) MIME パッケージではないので、これは少々お行儀が悪いものです。

`gnus-uu-tmp-dir`
: `gnus-uu` がその仕事をする場所です。

`gnus-uu-do-not-unpack-archives`
: `nil` でないと、`gnus-uu` は表示するためのファイルを探すためにアーカイブの中身までは見ません。

`gnus-uu-view-and-save`
: `nil` でないと、利用者はファイルを表示した後で常に保存するかどうかを尋ねられます。

`gnus-uu-ignore-default-view-rules`
: `nil` でないと、`gnus-uu` はディフォルトの表示規則を無視します。

`gnus-uu-ignore-default-archive-rules`
: `nil` でないと、`gnus-uu` はディフォルトのアーカイブ展開命令を無視します。

`gnus-uu-kill-carriage-return`
: `nil` でないと、`gnus-uu` は記事からすべてのキャリッジリターンを取り去ります。

`gnus-uu-unmark-articles-not-decoded`
: `nil` でないと、`gnus-uu` はデコードに失敗した記事に未読の印を付けます。

`gnus-uu-correct-stripped-uucode`
: `nil` でないと、`gnus-uu` は後続の空白が削除されてしまっている uuencode されたファイルを修復しようと試み ます。

`gnus-uu-pre-uudecode-hook`
: メッセージを`uudecode` に送る前に実行されるフックです。

`gnus-uu-save-in-digest`
: `nil` でないと、デコードせずに保存することを指示されたときに、`gnus-uu` は要約(digest) を保存します。この変数が`nil` であると、`gnus-uu` は何も加工を施さずにすべてを一つのファイルに保存します。要約の作成は概ね RFC1153 に準拠していますが、意味のある目次を付ける簡単な方法が見つからなかったので、私はそれらを単に落としました。

4.17.5.3 uuencode と投稿

`gnus-uu-post-include-before-composing`
: `nil` でないと、`gnus-uu` は記事を作成する前にエンコードするファイルを尋ねます。この変数が`t` であると、*C-c C-i* によってエンコードされたファイルを取り込むか、記事を投稿するときに取り込むかのどちらかをすることができます。

`gnus-uu-post-length`
: 記事の最大の長さです。エンコードされたファイルは全体のファイルを投稿するのに必要な量のファイルに分割されます。

`gnus-uu-post-threaded`
: `nil` でないと、`gnus-uu` はエンコードされたファイルをスレッドで投稿します。これはあまり賢い方法ではないかもしれません。というのは、今まで私が見た中で uuencode された記事を集めるのに、スレッドを追っていくことのできる他のデコーダーが存在しないからです。(えーと、私はそれをする一つのパッケージを見たことがあります—`gnus-uu` です。しかしどうも、それが数のうちに入るとは思えないのです...) ディフォルトは`nil` です。

`gnus-uu-post-separate-description`
: `nil` でないと、説明文は別の記事で投稿されます。最初の記事は普通(0/x) のように番号が付けられます。もしこの変数が`nil` であると、利用者の書いた説明分は最初のファイルの始めに取り込まれ、(1/x) の番号が付けられます。ディフォルトは`t` です。

4.17.6 ファイルの表示

デコードした後でファイルが何らかのアーカイブである場合、Gnus はアーカイブを展開しようと試み、アーカイブの中に表示できるファイルがあるかどうかを調べます。例えば、gzip された tar ファイル`pics.tar.gz` があって、ファイル`pic1.jpg` と`pic2.gif` を含んでいる場合、Gnus は主ファイルを解凍して tar を展開し、それから二つの絵を表示します。この展開の過程は再帰的なので、アーカイブにアーカイブのアーカイブがあると、それはすべて展開されます。

最後に、Gnus は普通はそれぞれの抽出された記事ごとに「疑似記事」を概略バッファーに挿入します。これらの『記事』に移動した場合は、実行する命令(普通は Gnus が提案をします) を入力するように促され、それからその命令が実行されます。

`gnus-view-pseudo-asynchronously` が`nil` であると、Emacs は先へ進む前に表示の終了を待ちます。

`gnus-view-pseudos` が`automatic` であると、Gnus は概略バッファーに疑似記事を挿入せず、それらをすぐに表示します。この変数が`not-confirm` であると、利用者は表示が済む前に確認さえも求められません。

`gnus-view-pseudos-separately` が`nil` でないと、表示されるそれぞれのファイルにつき一つの疑似記事が作成されます。`nil` であると、同じ表示命令を使うすべての命令がその命令の引数のリストとして渡されます。

`gnus-insert-pseudo-articles` が`nil` でないと、デコードのときに疑似記事を挿入します。ディフォルトでは`t` です。

さて、あなたはそんなふうに仮想サーバー の仮想グループ にある疑似記事 を読むことになるわけです。どうしてすべてが現実ではなくなってしまったんでしょうか? どうしてこんなところに来てしまったんでしょうか?

4.18 記事のトリートメント

この巨大な説明文書を読んできて、人々の著作を読むことがニュースリーダーの本当の目的だったことを、すっかり忘れてしまったかもしれません。記事を読むことです。残念ながら人々は書くことがとても苦手ですが、記事を読みやすくするための関数と変数は山のようにあります。

4.18.1 記事のハイライト

記事バッファーをフルーツサラダのように、いや総天然色のフルーツサラダのようにしたくありませんか。

W H a 現在の記事をもっとハイライトします。この関数は、ヘッダー、引用文、署名をハイライトし、本文とヘッダーにボタンを加えます。

W H h ヘッダーをハイライトします(`gnus-article-highlight-headers`)。ハイライトは変数`gnus-header-face-alist` に従って行なわれ、それはそれぞれの要素が(**正規表現 名前 内容**) という様式のリストです。**正規表現** はヘッダーに合致する正規表現、**名前** はヘッダーの名前をハイライトするのに使われるフェース(see Section 10.7 [Faces and Fonts], p. 282)、**内容** はヘッダーの値をハイライトするフェースです。最初に合致したものが使われます。**正規表現** の先頭に'^' を付けてはいけないことに注意してください---Gnus がそれを付け加えます。

W H c 引用された文をハイライトします(`gnus-article-highlight-citation`)。

引用文のハイライトをカスタマイズする変数は次の通りです:

`gnus-cite-parse-max-size`
: 記事の大きさがこの変数(ディフォルトでは 25000) のバイト数より大きい記事は、引用文のハイライトが行なわれません。

`gnus-cite-max-prefix`
: 引用符の最大の長さです(ディフォルトでは 20 です)。

`gnus-cite-face-list`
: 引用文をハイライトするために使われるフェースのリストです(see Section 10.7 [Faces and Fonts], p. 282)。同じメッセージの中に複数の記事からの引用があると、Gnus はそれぞれの記事からの引用をそれ用のフェースで表示しようとします。これにより、誰が何を書いたかが分かりやすくなるでしょう。

`gnus-supercite-regexp`
: 普通の Supercite 著者行に合致する正規表現です。

`gnus-supercite-secondary-regexp`
: 引き裂かれた Supercite 著者行に合致する正規表現です。

`gnus-cite-minimum-match-count`
: それが引用文であると判定する前に調べなければならない引用符の最小の数です。

`gnus-cite-attribution-prefix`
: 著者行の始まりに合致する正規表現です。

`gnus-cite-attribution-suffix`
: 著者行の終りに合致する正規表現です。

`gnus-cite-attribution-face`
: 著者行に使われるフェースです。その著者が書いた文の引用のためのフェースと融合されます。

`gnus-cite-ignore-quoted-from`
: 非-`nil` だったら、‘`>From` ’ で始まる行で引用文のハイライトは行なわれません。それらの行は、エンベロープ From 行と混同しないように、MTA がクオートした可能性があります。ディフォルト値は`t` です。

W H s
: 署名(signature) をハイライトします(`gnus-article-highlight-signature`)。`gnus-signature-separator` (see Section 4.18.10 [Article Signature], p. 102) の後のすべてのものは署名であると解釈され、`gnus-signature-face` でハイライトされます。それはディフォルトでは`italic` です。

記事を自動的にハイライトする方法についてはSection 5.4 [Customizing Articles], p. 129, を参照してください。

4.18.2 記事中の文の強調表示

(訳注: Fontisize == Fontify + Emphasize)

人々はよくニュースの記事で‘`_ これ_`’ や‘`* これ *`’ または‘`/これ/`’ のようなものを使って単語を強調します。Gnus は記事を*W e* 命令(`gnus-article-emphasize`) にかけることによって素敵に見えるようにできます。

強調がどのように処理されるかは変数`gnus-emphasis-alist` によって制御されます。これは連想リストで、最初の要素は合致するべき正規表現です。二番目の要素は、正規表現の中のどのグループが強調語全体を見つけるために使われるかを示す数値です。三番目は正規

表現のどのグループが表示されハイライトされるかを決める数値です。(この二つのグループの間にあるテキストは隠されます。) 四番目はハイライトさせるためののフェースです。

```
(setq gnus-emphasis-alist
      '(("_\\(\\w+\\)_" 0 1 gnus-emphasis-underline)
        ("\\*\\(\\w+\\)\\*" 0 1 gnus-emphasis-bold)))
```

(訳注: 上記の変数の値は、ディフォルトのままにしておくのが無難です。)

ディフォルトでは七つの規則があり、それらは以下のフェースを用います: `gnus-emphasis-bold`, `gnus-emphasis-italic`, `gnus-emphasis-underline`, `gnus-emphasis-bold-italic`, `gnus-emphasis-underline-italic`, `gnus-emphasis-underline-bold`, `gnus-emphasis-underline-bold-italic`。

これらのフェースを変更したいのであれば、*M-x customize* か`copy-face` を使うことができます。例えば`gnus-emphasis-italic` が代わりに赤のフェースを使うようにしたいのならば、次のようにすれば良いでしょう:

```
(copy-face 'red 'gnus-emphasis-italic)
```

任意の語を強調表示させたいときは、`gnus-group-highlight-words-alist` 変数を使うことができます。これは`gnus-emphasis-alist` と同じ構文を使います。`highlight-words` グループパラメーター(see Section 3.10 [Group Parameters], p. 23) を使うこともできます。

記事を自動的に強調表示させるやり方についてはSection 5.4 [Customizing Articles], p. 129, を参照してください。

4.18.3 記事を隠す

と言うよりはむしろ、記事の中にある特定のものを隠すことです。たいていの記事には、普通はありすぎるくらいのごみがあります。

W W a
: 記事バッファーでたくさんのものを隠します(*gnus-article-hide*)。特にこの関数はヘッダー、PGP、引用文、それに署名を隠します。

W W h
: ヘッダーを隠します(`gnus-article-hide-headers`)。See Section 5.1 [Hiding Headers], p. 125.

W W b
: あまり興味の持てないヘッダーを隠します(`gnus-article-hide-boring-headers`)。See Section 5.1 [Hiding Headers], p. 125.

W W s
: 署名を隠します(`gnus-article-hide-signature`)。See Section 4.18.10 [Article Signature], p. 102.

W W l
: `gnus-list-identifiers` で指定されているメーリングリストの標識を削除します。これらはいくつかのメーリングリストのサーバーがすべての`Subject` ヘッダーの最初に付ける文字列、例えば'`[zebra 4711]`' のようなものです。文字列の初めにある'`Re: `' は、削除を行なう前に跳び越されます。`gnus-list-identifiers` に`\\(..\\)` を含めてはいけません。

  `gnus-list-identifiers`
  : 表題から削除されるべきメーリングリストの標識に合致する正規表現です。これは正規表現のリストであることもできます。

W W P
: 不要なPEM (privacy enhanced messages (プライバシー拡張メッセージ)) の部分を隠します(`gnus-article-hide-pem`)。

W W B banner グループパラメーターで指定されたバナーを取り除きます(gnus-article-strip-banner)。これは主に、いくつかのメーリングリストや司会者付きのグループがすべての記事に追加する、鬱陶しいバナーと/ もしくは署名を隠すために使用されます。この関数を使う方法はbanner グループパラメーター(see Section 3.10 [Group Parameters], p. 23) をバナーを取り除きたいグループに追加することです。パラメーターは、消去されるテキストに合致する正規表現として解釈される文字列か、(最後の) 署名を消去するためのシンボルsignature、またはgnus-article-banner-alist の正規表現に対応した他のシンボルのいずれかであることができます。

例えば:

```
(setq gnus-article-banner-alist
      ((googleGroups .
        "^\n*--~--~---------\\(.+\n\\)+")))
```

グループにかかわらず、記事の送信者がgnus-article-address-banner-alist で設定されている特定のメールアドレスを持っているときだけ、広告のようなものを隠すことができます。

gnus-article-address-banner-alist
: メールアドレスとバナーの連想リストです。それぞれの要素は(*address* . *banner*) の形式を持ち、ここで*address* は From ヘッダーにあるメールアドレスに合致する正規表現です。また、*banner* はシンボルsignature、gnus-article-banner-alist の要素、正規表現またはnil のうちの一つです。*address* が著者のメールアドレスに合致すると、広告のようなものを消します。例えば、送信者が‘hail@yoo-hoo.co.jp’ というメールアドレスを持っていて、彼が送信するすべての記事に‘Do You Yoo-hoo!?’ のようなものがある場合、以下の要素でそれらを消すことができます。

```
("@yoo-hoo\\.co\\.jp\\'" . "\n_+\nDo You Yoo-hoo!\\?\n.*\n.*\n")
```

W W c 引用文を隠します(gnus-article-hide-citation)。隠蔽をカスタマイズするいくつかの変数は:

gnus-cited-opened-text-button-line-format
gnus-cited-closed-text-button-line-format
: Gnus はどこの引用文が隠されているかを示すためにボタンを付け加え、文章の隠蔽を切り替えられるようにします。この変数の様式は、以下のフォーマットのような変数によって指定されます(see Section 10.4 [Formatting Variables], p. 272)。次の指定が有効です:

  ‘b’ 隠された文の最初のポイントです。

  ‘e’ 隠された文の最後のポイントです。

  ‘l’ 隠されたリージョンの文字の数です。

  ‘n’ 隠された文の行の数です。

gnus-cited-lines-visible
: 隠さずに表示しておく、引用文の先頭からの行数です。これは、隠さずに表示する先頭からの行と、隠さずに表示する末尾からの行の、それぞれの数の cons セルであることもできます。

W W C-c 以下の二つの変数に依存して、引用文を隠します(gnus-article-hide-citation-maybe):

gnus-cite-hide-percentage
: 引用文の割合のパーセンテージが、この変数(ディフォルトは 50)より大きかったら、引用文を隠します。

gnus-cite-hide-absolute
: 隠される前に、引用文は少なくともこの長さ(ディフォルトは 10)でなければなりません。

W W C 根本でない記事の引用文を隠します(gnus-article-hide-citation-in-followups)。これは対話的命令としてはあまり役に立たないかもしれませんが、自動的に実行させるには手軽な関数でしょう(see Section 5.4 [Customizing Articles], p. 129)。

これらのすべての『隠蔽』命令は切り替え命令ですが、これらの命令に負の接頭引数を与えると、それらは前に隠されていたものを表示します。正の接頭引数を与えれば、それらは常に隠します。

引用文をカスタマイズするためのさらなる変数について、Section 4.18.1 [Article Highlighting], p. 90, も参照してください。

自動的に記事の要素を隠すための方法はSection 5.4 [Customizing Articles], p. 129, を参照してください。

4.18.4 記事の洗濯

私たちはこれをもっともな理由の下で『記事の洗濯』(article washing) と呼んでいます。*A* キーは使われていたので、代わりに*W* キーを使う必要がありました。

「洗濯」は『何かの何かを何か別のものに変換する』と定義されますが、普通はもっと良く見える何かに落ち着きます。もっときれいになります、たぶん。

Gnus が記事を表示するディフォルトのやり方を変えたいときはSection 5.4 [Customizing Articles], p. 129, を参照してください。

C-u g これは洗濯ではなくて、その逆です。これをタイプすると、ディスクやサーバーにあるがままの記事が見えます。

g 現在の記事の再表示を強制します(gnus-summary-show-article)。これもまた本当の洗濯ではありません。これをタイプすると、以前に適用された対話的な洗濯機能はご破算にされ、すべてのディフォルトのトリートメントを施された記事が表示されます(see Section 5.4 [Customizing Articles], p. 129)。

W l ページの区切りを現在の記事から取り除きます(gnus-summary-stop-page-breaking)。ページの区切りに付いてはSection 5.6 [Misc Article], p. 133, を参照してください。

W r 記事バッファーでカエサル変換(Caesar rotate, rot13) を行ないます(gnus-summary-caesar-message)。カエサル変換か rot13 を用いて読むことを指定する、判読不可能な記事です(典型的には攻撃的な冗談などです。)

普通は“rot13”と呼ばれています。それはアルファベットの位置が 13 個回転するからです。例えば、‘B’ (2 番目の文字) → ‘O’ (15 番目の文字)。これは時々『カエサル変換』と呼ばれることもあります。というのは、カエサルがこの形式の、えーと、いささか貧弱な暗号化を採用したという噂があるからです。

W m 記事バッファーをモールスでデコードします(gnus-summary-morse-message)。

W i 現在の記事にある IDNA エンコードされたドメイン名をデコードします。IDNA エンコードされたドメイン名は‘xn--bar’ のように見えます。これを実行した後で文字列がデコードされないままだったら、おそらくそれは不正な IDNA 文字列でしょう(‘xn--bar’ は不正です)。このコマンドを動かすためには、GNU Libidn (https://www.gnu.org/software/libidn/) をインストールしていなければなりません。

W t

t 記事バッファーにすべてのヘッダーを表示するかどうかを切り替えます(gnus-summary-toggle-header)。

W v 記事バッファーにすべてのヘッダーを永続的に表示するかどうかを切り替えます(gnus-summary-verbose-headers)。

W o オーバーストライクを処理します(gnus-article-treat-overstrike)。

訳注: 以下のような重ね打ちを指示する文字列を bold や underline で表示します。

‘B^HBo^Hol^Hld^Hd’, ‘U^H_n^H_d^H_e^H_r^H_l^H_i^H_n^H_e^H_’

W d マイクロソフト Smart Quotes をgnus-article-smartquotes-map ((gnus-article-treat-smartquotes)) に従って処理します。この関数は文字が Smart Quotes かどうかを推測するので、対話的にのみ使用されるべきであることに注意してください。

Smart Quotes はもっと多くの引用文字を提供するために、マイクロソフトが勝手に文字マップを拡張したものです。もし、アポストロフィ(’) や引用記号(") などがあるべきところに\222 や\264 のようなものが見えてしまったら、洗濯してみてください。

W U 多くの非-ASCII 文字をそれらと等価なASCII 文字に変換します(gnus-article-treat-non-ascii)。これは、フォントに制限があって、アクセント文字や「進んでいる」句読点などを表示しない端末では、たいていの場合に役立ちます。例えば‘≫’ から‘>>’ への変換などを行ないます。

W Y f いかれた Outlook (Express) の記事を完全に醜くないようにします(訳注: de-ugly-fy)。Smart Quote を処理して行の折り返しを解除、著者行の修復と引用文の整頓をします。(gnus-article-outlook-deuglify-article)。

W Y u 折り返された引用行のように見える行の折り返しを解きます。折り返しが解かれた行の最小および最大の長さを表すgnus-outlook-deuglify-unwrap-min およびgnus-outlook-deuglify-unwrap-max を調整することによって、どんな行の折り返しが解かれるかを制御することができます。(gnus-article-outlook-unwrap-lines)。

W Y a 壊れた著者行を修復します。
(gnus-article-outlook-repair-attribution)。

W Y c 壊れた引用文を、テキストを整理し直すことによって修復します。(gnus-article-outlook-rearrange-citation)。

W w 行を折り返します(gnus-article-fill-cited-articles)。
折り返す幅を指定するために、命令に数値接頭引数を与えることができます。

W Q 長い行を折り返します(gnus-article-fill-mode-lines)。
コマンドに数値接頭引数を与えることによって、折り返しの幅を指定することができます。

W C それぞれの文の最初の語を大文字にします(gnus-article-capitalize-sentences)。

W c CRLF の組(すなわち、行の最後の‘^M’) を LF に変換します(これは DOS の行末の世話をします)。そうしてから残りの CR を LF に変換します(これは MAC の行末の世話をします) (gnus-article-remove-cr)。

Quoted-printable を処理します(gnus-article-de-quoted-unreadable)。Quoted-Printable は非-ASCII (すなわち 8-bit) の記事を送るときに使われる一般的なMIME エンコーディングです。それは概して‘déjà vu’ のようなものを‘d=E9j=E0 vu’ に見せるので、とても読み辛くなります。問題の記事が、そのエンコーディングが行なわれたことを示すContent-Transfer-Encoding ヘッダーを持っていれば、通常それは Gnus によって自動的に行なわれることに注意してください。接頭引数が与えられると、文字セットが尋ねられます。

W 6 Base64 をデコードします(gnus-article-de-base64-unreadable)。Base64 は非-ASCII (すなわち 8-bit) の記事を送るときに使われる、一般的なMIME エンコーディングです。問題の記事が、そのエンコーディングが行なわれたことを示すContent-Transfer-Encoding ヘッダーを持っていれば、通常それは Gnus によって自動的に行なわれることに注意してください。接頭引数が与えられると、文字セットが尋ねられます。

W Z HZ または HZP を処理します。HZ (または HZP) は中国語の記事を伝送するときに使われる一般的な符号です。これは‘~{<:Ky2;S{#,NpJ)l6HK!#~}’ のような典型的な文字列を作ります。

W A ANSI SGR シーケンスを overlay または extent に変換します(gnus-article-treat-ansi-sequences)。ANSI シーケンスは中国語のニュースグループで強調表示に使われています。

W u URL に含まれる改行を削除します。いくつかのメイラーは、行を短くするために出ていくメールに改行を挿入しますが、これは長い URL を複数の行に分割してしまいます。改行を削除することによって、それらの URL を復旧させます(gnus-article-unsplit-urls)。

W h HTML を処理します。当該メッセージがHTML であることを示すContent-Type ヘッダーを持っていたならば、それは Gnus によって自動的に行なわれることに注意してください。

接頭引数が与えられると、文字セットを尋ねられます。それがもし数値だったら、gnus-summary-show-article-charset-alist (see Section 4.4 [Paging the Article], p. 56) で定義されている文字セットが使われます。(訳注: 実質的には「文字セット」ではなくてcoding-system です。)

ディフォルトではHTML の変換にmm-text-html-renderer (see Section “表示のカスタマイズ” in *The Emacs MIME Manual*) で設定された関数を使います。使うことができる、あらかじめ用意された関数は以下の通りです:

shr Gnus の簡易 html 描画器(shr) を使います。

`gnus-w3m` w3m を使って Gnus に描画させます。

`w3m` emacs-w3m (`http://emacs-w3m.namazu.org/`) を使います。

`w3m-standalone`
w3m (`https://w3m.sourceforge.net/`) を使います。

`links` Links (`http://links.twibright.com/`) を使います。

`lynx` Lynx (`https://lynx.browser.org/`) を使います。

W D F HTML 記事のフォントをプロポーショナルにするかどうかを切り替えます。これは現在の記事バッファーにおける`shr-use-fonts` 変数を一時的に変更します。

W b クリックできるボタンを記事に加えます(`gnus-article-add-buttons`)。See Section 4.18.6 [Article Buttons], p. 98.

W B クリックできるボタンを記事のヘッダーに加えます(`gnus-article-add-buttons-to-head`)。

W p 署名付きコントロールメッセージの認証を行ないます(`gnus-article-verify-x-pgp-sig`)。`newgroup` や`checkgroups` といったコントロールメッセージは、通常そのニュースグループ階層のメインテイナーによって署名されています。認証を行なうためには、メインテイナーのPGP 公開鍵をあなたのキーリングに追加しなければなりません。[1]

W s 署名されたメッセージ(PGP, PGP/MIME またはS/MIME によって) を検証します(`gnus-summary-force-verify-and-decrypt`)。See Section 4.31 [Security], p. 122.

W a 記事の本文の先頭から`X-No-Archive` ヘッダーのようなヘッダーを取り除きます(`gnus-article-strip-headers-in-body`)。

W E l 記事の先頭にあるすべての空白行を取り除きます(`gnus-article-strip-leading-blank-lines`)。

W E m すべての空白行を空行で置き換えてから、すべての複数の空行を一つの空行で置き換えます(`gnus-article-strip-multiple-blank-lines`)。

W E t 記事の最後にあるすべての空白行を取り除きます(`gnus-article-remove-trailing-blank-lines`)。

W E a 上の三つの命令をすべて実行します(`gnus-article-strip-blank-lines`)。

W E A すべての空白行を取り除きます(`gnus-article-strip-all-blank-lines`)。

W E s 記事の本文のすべての行頭にあるすべての空白を取り除きます(`gnus-article-strip-leading-space`)。

W E e 記事の本文のすべての行末にあるすべての空白を取り除きます(`gnus-article-strip-trailing-space`)。

自動的に記事の洗濯を行なわせる方法はSection 5.4 [Customizing Articles], p. 129, を参照してください(訳注: 実は多くの洗濯がディフォルトで自動的に行なわれます)。

[1] 多くのニュースグループ階層のメインテイナーのPGP の鍵は`https://ftp.isc.org/pub/pgpcontrol/README.html` から入手することができます。

4.18.5 記事ヘッダー

これらのコマンドは記事ヘッダーをいろいろに変形させます。

W G u 折り返されたヘッダー行を一行にします(gnus-article-treat-unfold-headers)。

W G n Newsgroups とFollowup-To ヘッダーを折り返します(gnus-article-treat-fold-newsgroups)。

W G f すべてのメッセージヘッダーを折り返します(gnus-article-treat-fold-headers)。

W E w すべてのヘッダーから余分な空白を取り除きます(gnus-article-remove-leading-whitespace)。

4.18.6 記事のボタン

人々はよく記事の中に他の資料を参照するための案内を入れることがありますが、それらの参照への案内の上でRET を打つか、マウスの真中のボタンを使ったときに、彼らが話題にしているのが何であれ、最小限の曖昧さで Gnus が取得することができれば素敵でしょう。

特定の標準的な参照に、Gnus はディフォルトで「ボタン」を付けます: ちゃんとした URL、メールアドレス、Message-ID、Info へのリンク、man ページ、それに関連する Emacs または Gnus の参考文献です。これは二つの変数によって制御されていて、その一つは記事の本文を扱い、もう一つは記事のヘッダーを扱います。

gnus-button-alist
それぞれの要素が次のような様式を持つ連想リストです:

(*regexp button-par use-p function data-par*)

regexp この正規表現(大文字と小文字は区別されません) に合致するすべてのテキストは、外部への参照であるとみなされます。これは埋め込まれた URL に合致する典型的な正規表現です: '<URL:\\([^\n\r>]*\\)>'。これはまた正規表現の値を持つ変数であってもよく、有用な変数としてgnus-button-url-regexp およびgnus-button-mid-or-mail-regexp があります。

button-par
Gnus は合致したもののどの部分がハイライトされるのかを知らなければなりません。これは正規表現のどの副表現がハイライトされるかを指定する番号です。すべてをハイライトしたいのなら、ここで 0 を使ってください。

use-p この式は**評価** され、結果がnil でなかったら、これは合致であるとみなされます。これは間違った合致を避けるために特別な選別をしたいときに役に立ちます。ここではしばしばgnus-button-*-level のような名前の変数が使われますが、See Section 4.18.7 [Article Button Levels], p. 100, 他のどんな形式でも使うことができます。

function この関数が、このボタンをクリックしたときに呼ばれます。

data-par *button-par* のように、これは部分表現の番号ですが、これは合致のどの部分が*function* にデータとして送られるかを指定します。

したがって URL をボタンにする完全な要素は、こうなります。

```
("<URL:\\([^\n\r>]*\\)>" 0 t gnus-button-url 1)
```

`gnus-header-button-alist`
: これは他の連想リストと同じようなものですが、記事のヘッダーだけに適用されることと、それぞれの項目がどのヘッダーにボタンを付けるかを指示するための追加の要素を持っていることが異なります:

```
(header regexp button-par use-p function data-par)
```

header は正規表現です。

4.18.6.1 関連する変数と関数

`gnus-button-*-level`
: Section 4.18.7 [Article Button Levels], p. 100, を参照してください。

`gnus-button-url-regexp`
: 埋め込まれた URL に合致する正規表現です。上述の変数のディフォルトの値で使われます。

`gnus-button-man-handler`
: Man ページの表示に使う関数です。少なくとも一つの引数として Man ページの名前の文字列を受け付けなければなりません。

`gnus-button-mid-or-mail-regexp`
: Message-ID かメールアドレスに合致する正規表現です。

`gnus-button-prefer-mid-or-mail`
: この変数は‘`foo123@bar.invalid`’ のような文字列のボタンが押されたときに、何を行なうかを決める変数です。このような文字列は Message-ID かメールアドレスのいずれかです。もし`mid` か`mail` というシンボルのうちの一つだったら、Gnus は常にそれぞれ文字列が Message-ID またはメールアドレスであると仮定します。この変数が`ask` というシンボルに設定されると、Gnus はいつも利用者が何をしたいかを尋ねます。それが関数だった場合、たった一つの文字列を引数として呼ばれます。その関数は`mid`、`mail`、`invalid` または`ask` を返さなければなりません。ディフォルト値は関数`gnus-button-mid-or-mail-heuristic` です。

`gnus-button-mid-or-mail-heuristic`
: その引数がメッセージの ID かメールアドレスであるかを推定する関数です。メッセージの ID だったら`mid` を、メールアドレスだったら`mail` を、不確かだったら`ask` を、そして無効な文字列だったら`invalid` を返します。

`gnus-button-mid-or-mail-heuristic-alist`
: 関数`gnus-button-mid-or-mail-heuristic` で使われる`(RATE . REGEXP)` 対の連想リストです。

`gnus-article-button-face`
: ボタンに使われるフェースです。

`gnus-article-mouse-face`
: マウスのカーソルがボタンの上にあるときに使われるフェースです。

記事に自動的にボタンを付ける方法は、Section 5.4 [Customizing Articles], p. 129, を参照してください。

4.18.7 記事ボタンのレベル

変数gnus-button-*-level の値が高いほど、より多くのボタンが現れます。レベルがゼロだったらボタンは表示されません。ディフォルト値(それは 5) では、とてもたくさんのボタンをすでに見ているはずです。高いレベルではより多くのボタンを見ることになりますが、多くの要らないものも現れるかもしれません。それらを避けるために、特定のグループに対して変数gnus-button-*-level を設定しても良いでしょう(see Section 3.10 [Group Parameters], p. 23)。gnus-parameters 変数の例です:

```
;; いくつかのグループで gnus-button-*-level を増やす:
(setq gnus-parameters
      '(("\\<\\(emacs\\|gnus\\)\\>" (gnus-button-emacs-level 10))
        ("\\<unix\\>"               (gnus-button-man-level 10))
        ("\\<tex\\>"                (gnus-button-tex-level 10))))
```

gnus-button-browse-level
: Message-ID、メールアドレスおよびニュースの URL を参照する案内の表示を制御します。関連する変数と関数にはgnus-button-url-regexp、browse-url およびbrowse-url-browser-function があります。

gnus-button-emacs-level
: Emacs または Gnus への参照の表示を制御します。関連する関数は、gnus-button-handle-custom, gnus-button-handle-describe-function, gnus-button-handle-describe-variable, gnus-button-handle-symbol, gnus-button-handle-describe-key, gnus-button-handle-apropos, gnus-button-handle-apropos-command, gnus-button-handle-apropos-variable, gnus-button-handle-apropos-documentation およびgnus-button-handle-library です。

gnus-button-man-level
: (Unix の) man ページへの参照の表示を制御します。gnus-button-man-handler を見てください。

gnus-button-message-level
: Message-ID、メールアドレスおよびニュースの URL の表示を制御します。関連する変数と関数にはgnus-button-mid-or-mail-regexp, gnus-button-prefer-mid-or-mail, gnus-button-mid-or-mail-heuristic およびgnus-button-mid-or-mail-heuristic-alist があります。

4.18.8 記事の日付

日付は聞いたことの無い何か辺鄙なタイムゾーンで作成されていることが良くあるので、記事が送られたときに何時だったかを知ることができるのはとても良いことです。

W T u
: UT (いわゆる GMT, ZULU) で日付を表示します(gnus-article-date-ut)。

W T i
: 日付を国際的な形式である ISO 8601 で表示します(gnus-article-date-iso8601)。

W T l
: 日付をローカル・タイムゾーンで表示します(gnus-article-date-local)。

W T p
: 日付を英語で楽に発音できる形式で表示します(gnus-article-date-english)。

W T s
: 日付を利用者定義の様式を使って表示します(gnus-article-date-user)。その様式は変数gnus-article-time-format で指定される、format-time-string

に渡される文字列です。指定することができる様式の一覧は、変数の説明文を見てください。

W T e 記事が投稿されてから今までどれくらいの時間が経過したかを表示します(gnus-article-date-lapsed)。こんなふうに。

```
Date: 6 weeks, 4 days, 1 hour, 3 minutes, 8 seconds ago
```

この行を絶えず更新させるためには、頻度を秒数でgnus-article-update-date-headers 変数に設定してください(ディフォルトはnil です)。

W T o 本来の日付を表示します(gnus-article-date-original)。これはあなたが普段は他の変換関数を使っていて、それが完全に間違ったことをしているのではないかと心配になったときに役に立ちます。例えば、記事が 1854 年に投稿されたと主張したとしましょう。しかし、そのようなことは完全に 不可能です。私が信用できませんか? * くすくす *

好みの書式で自動的に日付を表示する方法はSection 5.4 [Customizing Articles], p. 129, を参照してください。

4.18.9 記事に表示するもの

これらのコマンドは、いろんな取るに足らないギミック(gimmicks) の表示を、それらをサポートしている Emacs の記事バッファーに追加します。

X-Face ヘッダーは小さな白黒画像で、メッセージヘッダーから持ってきます(see Section 10.16.1 [X-Face], p. 287)。

Face ヘッダーは小さなカラー画像で、メッセージヘッダーから持ってきます(see Section 10.16.2 [Face], p. 288)。

スマイリーは、人々がメッセージに散らかしたがる小さな‘:-)’ シンボルです。

一方 Picon はあなた自身のシステムに依存し、Gnus はヘッダーに合致するあなたの持ち物を探してみます(see Section 10.16.4 [Picons], p. 289)。

Gravatars はhttps://en.gravatar.com/ からオンラインで取得します(see Section 10.16.5 [Gravatars], p. 290)。

これらすべての機能はトグルです。もしすでにそれらが存在していたならば、それらは削除されます。

W D x X-Face をFrom ヘッダーに表示します(gnus-article-display-x-face)。

W D d Face をFrom ヘッダーに表示します(gnus-article-display-face)。

W D s スマイリーを表示します(gnus-treat-smiley)。

W D f From ヘッダーを Picon 化します(gnus-treat-from-picon)。

W D m すべてのメールヘッダー(すなわちCc、To) を Picon 化します(gnus-treat-mail-picon)。

W D n すべてのニュースヘッダー(すなわちNewsgroups とFollowup-To) を Picon 化します(gnus-treat-newsgroups-picon)。

W D g From ヘッダーをアバター化します(gnus-treat-from-gravatar)。

W D h メールヘッダーぜんぶ(つまりCc、To) をアバター化します(gnus-treat-from-gravatar)。

`W D e` 一部のシンボルには絵文字ではない表現と絵文字表現の両方があります。このコマンドは Gnus に絵文字の表現を選択させます(gnus-article-emojize-symbols)。

`W D D` 記事バッファーからすべての画像を削除します(gnus-article-remove-images)。

`W D W` gnus-article-html で描画したHTML 記事を読んでいるときに、阻止された画像をこのコマンドでバッファーに挿入することができます(gnus-html-show-images)。

4.18.10 記事の署名

それぞれの記事は二つの部分に分けられます---ヘッダーと本文です。本文は署名部分と文章部分に分けることができます。どれが署名とみなされるかを決める変数はgnus-signature-separator です。これは普通は RFC 5536 で規定されている標準の'^-- $' です。しかし、多くの人が標準ではない署名セパレーターを使うので、この変数は一つ一つ試される、正規表現のリストであることもできます。(探索は本文の最後から始めへとなされます。) よくありそうな値は:

```
(setq gnus-signature-separator
      '("^-- $"         ; 標準
        "^-- *$"        ; 普通の崩し方
        "^-------*$"    ; 多くの人は長ーーい横棒の
                        ; 行を使います。みっともない!
        "^ *--------*$" ; 二倍みっともない!
        "^________*$"   ; 下線も人気があります
        "^========*$")) ; 邪道!
```

あなたが寛容であればあるほど、間違った結果を得ることになるでしょう。

gnus-signature-limit は記事を表示するときにどれが署名とみなされるかへの制限を提供します。

1. これが整数であれば、署名はこの整数より(文字数で) 長くなっていてはいけません。
2. これが浮動小数点数であれば、署名はその数値より(行数で) 長くなっていてはいけません。
3. これが関数であれば、その関数は引数なしで呼ばれ、それがnil を返せば、そのバッファーには署名がありません。
4. これが文字列であれば、それは正規表現として使われます。もしそれが合致すれば、当のその文字列は署名ではありません。

この変数は、要素が上に列挙された型のリストであることもできます。例です:

```
(setq gnus-signature-limit
      '(200.0 "^---*Forwarded article"))
```

これは署名セパレーターの後に 200 を超える行があるか、セパレーターの後のテキストが正規表現'^---*Forwarded article' に合致すれば、結局それは署名ではないということです。

4.18.11 記事いろいろ

`A t` 記事をある言語から別のものへ変換します(gnus-article-babel)。

4.19 MIME コマンド

以下のコマンドはすべて数値接頭引数を理解します。例えば*3 K v* は「三番目のMIME パートを表示する」という意味です。

b

K v MIME パートを表示します。

K o MIME パートを保存します。

K O ファイル名の入力を求めてからMIME パートを保存し、それを記事から取り除きます。取り除かれたMIME オブジェクトは message/external-body 型のMIME 形式として参照されるようになります。

K r MIME パートを外部にある本体で置き換えます。

K d MIME パートを削除して、削除したことの案内を追加します。

K c MIME パートをコピーします。

K e MIME パートを外部コマンドで表示します。

K i MIME パートをバッファー内に表示します。

K | MIME パートを外部コマンドにパイプします。

以降のMIME コマンドの残りは、数値接頭引数を同じやり方では使いません:

K H 現在の記事の‘`text/html`’ パートを WWW ブラウザーで見ます。メッセージに`cid` 形式で埋め込まれたインライン画像は一般に安全だと考えられているので適切に処理されます。接頭引数が与えられなければ、すべてのHTML パートの先頭に記事のヘッダーが加えられます。

警告: Spammers はHTML 記事中の画像への(`http` 形式の) リンクを、あなたがその記事を読んだかどうかを確かめるために使います。このコマンドはHTML 記事からその手の“web bugs” を取り除かないでブラウザーに渡すので、このコマンドは信頼できる送信者からのメールでだけ使うべきです。

このコマンドは画像を含むHTML コンテンツをブラウザーに渡すために一時ファイルを作り、グループを抜け出るときに(もしあなたが望めば) それらを削除します。

K b すべてのMIME パートの先頭にボタンを付加します。埋め込まれたパートをセーブ(または他の動作を実行) しようとするときに、たいてい便利です。

W M h MIME パート・ボタンを記事のヘッダーの最後に表示します(`gnus-mime-buttonize-attachments-in-header`)。このコマンドは表示をトグルで切り替えます。ヘッダーに加えられるボタンは記事のボディーでインライン表示できないものだけです。ボタンを常に表示したいなら`gnus-mime-display-attachment-buttons-in-header` を`nil` 以外の値にしてください。ディフォルトは`t` です。ボタンの見栄えを変えるには、`gnus-header-face-alist` をカスタマイズしてください。

K m ときたま、ヘッダーが無かったり間違ったヘッダーを持つマルチパートのメッセージが送信されてきます。このコマンドは、それらのメッセージがより快適に表示されるように「修復」を試みます(`gnus-summary-repair-multipart`)。

X m MIME タイプに合致するすべてのパートを、ディレクトリーにセーブします(gnus-summary-save-parts)。プロセス/接頭引数の習慣を理解します(see Section 10.1 [Process/Prefix], p. 271)。

M-t 記事バッファーにボタンを表示するかしないかを切り替えます(gnus-summary-toggle-display-buttonized)。

W M w 記事ヘッダーにある RFC 2047 でエンコードされた語をデコードします(gnus-article-decode-mime-words)。

W M c エンコードされた記事の本文を、文字セットでデコードします(gnus-article-decode-charset)。

このコマンドは、文字セットを決めるためにContent-Type ヘッダーを調べます。記事にそんなヘッダーが無い場合でも、接頭引数を与えることによって、デコードするための文字セットを入力することは可能です。ある共通のエンコーディングを使って(でもMIME ヘッダーは含めずに) 人々が記事を投稿する地域的なグループでは、charset グループ/トピック・パラメーターに必要な文字セットを設定すれば良いでしょう(see Section 3.10 [Group Parameters], p. 23)。

W M v 現在の記事にある、すべてのMIME パートを表示します(gnus-mime-view-all-parts)。

関連する変数:

gnus-ignored-mime-types
これは正規表現のリストで、これに含まれている正規表現に合致するMIME タイプは、Gnus によって完全に無視されます。ディフォルト値はnil です。

すべての Vcard を無視させるには、こんなふうにしてください:

```
(setq gnus-ignored-mime-types
      '("text/x-vcard"))
```

gnus-article-loose-mime
非-nil だったら、Gnus は記事をMIME メッセージとして解読する前に、'MIME-Version' があることを必要としません。これは、ある壊れたメール・ユーザー・エージェントからのメッセージを読むときに役立ちます。ディフォルトはt です。

gnus-article-emulate-mime
MIME ではない別のエンコーディングの手法があります。最も一般的なのは'uuencode' ですが、yEncode も普及してきています。この変数が非-nil になっていると、Gnus はメッセージの本文にそれらのエンコーディングが見つかるかどうかを調べ、もしあったならば、それらを Gnus のMIME 機構で処理します。ディフォルトはt です。デコードできるのは単一の yEnc でエンコードされたパートだけです。Gnus はエンコードについてはサポートしません。

gnus-unbuttonized-mime-types
これは正規表現のリストで、これに含まれている正規表現に合致するMIME タイプには、ボタンが付加されません。ただし、それらが表示されないか、gnus-buttonized-mime-types 変数の方が優先される場合を除いて、ですが。ディフォルト値は(".*/.*") です。この変数はgnus-inhibit-mime-unbuttonizing がnil のときだけ使われます。

`gnus-buttonized-mime-types`
: これは正規表現のリストで、これに含まれている正規表現に合致するMIME タイプには、それらが表示されない場合を除いて、ボタンが付加されます。この変数は`gnus-unbuttonized-mime-types` よりも優先されます。ディフォルト値は`nil` です。この変数は`gnus-inhibit-mime-unbuttonizing` が`nil` のときだけ使われます。

  例えば、セキュリティーのボタンだけを表示して、他のボタンを表示しないようにするには、この変数を`("multipart/signed")` に設定して、`gnus-unbuttonized-mime-types` はディフォルト値のままにしておいてください。

  また、このリストに`"multipart/alternative"` を加えることによって、そういうメールに含まれている二つのメディア・タイプのうちの一つを選ぶことができる、ラジオボタンを表示させることができます。`mm-discouraged-alternatives` も参照してください(see Section “表示のカスタマイズ” in *The Emacs MIME Manual*)。

`gnus-inhibit-mime-unbuttonizing`
: これが非-`nil` だと、すべてのMIME パートにボタンを付加します。ディフォルト値は`nil` です。

`gnus-article-mime-part-function`
: それぞれのMIME パートに対して、この関数がMIME ハンドル(訳注: パートのタイプや内容物を表現するために、Gnus の内部で使われるデータの構造体)を引数にして呼ばれます。この関数は、利用者が記事から情報を集め(例えばVcard の情報を bbdb のデータベースに加え) たり、パートに基づいて何かを起動(例えば、すべての jpeg をあるディレクトリーにセーブ) するために使われることが意図されています。

  後者を行なう関数の例です:

```
(defun my-save-all-jpeg-parts (handle)
  (when (equal (car (mm-handle-type handle)) "image/jpeg")
    (with-temp-buffer
      (insert (mm-get-part handle))
      (write-region (point-min) (point-max)
                    (read-file-name "Save jpeg to: ")))))
(setq gnus-article-mime-part-function
      'my-save-all-jpeg-parts)
```

`gnus-mime-multipart-functions`
: MIME マルチパートの型と、それらを扱う関数の連想リストです。

`gnus-mime-display-multipart-alternative-as-mixed`
: "multipart/alternative" のパートを"multipart/mixed" であるものとして表示します。

`gnus-mime-display-multipart-related-as-mixed`
: "multipart/related" のパートを"multipart/mixed" であるものとして表示します。

  もし‘`text/html`’ を表示するのが気に入らないのなら、`mm-discouraged-alternatives` を参照してください。ただし(それで"text/html" を表示しない

ように設定して、かつ) この変数がnil だと、"multipart/related" パートの中にある画像や他の資料を見逃してしまうかもしれません。See Section “表示のカスタマイズ” in *The Emacs MIME Manual*.

gnus-mime-display-multipart-as-mixed
: "multipart" のパートを"multipart/mixed" であるものとして表示します。もしtだと、gnus-mime-display-multipart-alternative-as-mixed およびgnus-mime-display-multipart-related-as-mixed がnil であっても、この設定の方が優先されます。

mm-file-name-rewrite-functions
: MIME パートのファイル名を書き換えるために使われる関数のリストです。それぞれの関数はファイル名を受け取って、ファイル名を返します。

  出来合いの関数は mm-file-name-delete-whitespace, mm-file-name-trim-whitespace, mm-file-name-collapse-whitespace および mm-file-name-replace-whitespace です。最後のものはファイル名に含まれるそれぞれの空白文字を、変数mm-file-name-replace-whitespace の値で置き換えます。ディフォルト値は"_" (単一の下線) です。

  標準の関数であるcapitalize, downcase, upcase およびupcase-initials も、役に立つでしょう。

  ファイル名に含まれる空白文字が害をもたらすことは、みんなが知っています。ただし、気にかけない人たちを除いて、ですが。そんな蒙昧の人たちから、たくさんの添付ファイルを受け取るのであれば、こんなものを~/.gnus.el ファイルに追加することによって、安寧な生活を送ることができるでしょう。

```
(setq mm-file-name-rewrite-functions
      '(mm-file-name-trim-whitespace
        mm-file-name-collapse-whitespace
        mm-file-name-replace-whitespace))
```

4.20 文字セット

人々はいろいろな文字セットを使いますが、私たちは彼らが何の文字セットを使っているかを教えてくれるMIME を持っています。あるいはもっと正確に言えば、持っていたらいいなあと思います。多くの人たちがMIME を利用しないか理解しないニュースリーダーとメイラーを使って、何の文字セットを使うかを言わずに、単にメッセージを送出するのですが、これを少しばかり救済するために、いくつかの地域的なニュース階層には、何の文字セットがディフォルトであるかを宣言する取り決めがあります。例えば‘fj’ 階層ではiso-2022-jp を使っています。

この知識はgnus-group-charset-alist 変数にエンコードされています。これは正規表現(グループのフルネームに合致した最初の項目を使います) と、それらのグループを購読するときに使われるディフォルトの文字セットの、連想リストです。

加えて、人々のいくらかはMIME を意識していると自称(soi-disant) しているくせに、実はそうではないエージェントを使っています。それらは、実際にはメッセージがkoi-8なのにiso-8859-1 だと、陽気にメッセージに刻印するのです。ここでは救済のためにgnus-newsgroup-ignored-charsets 変数を使うことができます。そのリストに連ねられた文字セットは無視されます。この変数は、グループパラメーター(see Section 3.10 [Group

Parameters], p. 23) を使って、グループ毎に設定することができます。ディフォルト値は(unknown-8bit x-unknown) で、それはいくつかのエージェントが内蔵し、主張する値を含んでいます。

投稿する場合に、MIME でエンコードしてはいけない文字セットを判定するために、gnus-group-posting-charset-alist が使われます。例えばいくつかの階層では、quoted-printable でヘッダーをエンコードすることは嫌われます。

この変数は正規表現と、投稿に際してエンコードしなくても良いことを許された(またはエンコードすることが嫌われる) 文字セットの連想リストです。それぞれの要素は(*test header body-list*) の形式であり、それらは次の意味を持ちます。

test Newsgroups ヘッダーに合致する正規表現、または変数シンボルのどちらかです。後者の場合は、その値を調べた結果が非-nil だったら、その要素が採用されることになります。

header ヘッダーをエンコードしなくても良い文字セットです(nil は、すべての文字セットをエンコードすることを意味します)。

body-list "Content-Transfer-Encoding: 8bit"でもって本文をエンコードしても良い(または quoted-printable や base64 でエンコードすることが嫌われる) 文字セットのリスト、または特別な値の一つであるnil (常に quoted-printable でエンコードする)、またはt (常に"Content-Transfer-Encoding: 8bit"を使う) です。

メッセージを送信するときに何の文字セットが使われるかを制御する付加的な変数については、See Section "エンコーディングのカスタマイズ" in *The Emacs MIME Manual*, を参照してください(訳注: 特に日本語のメッセージの文字セットについては、例えば変数mm-coding-system-priorities を参照してください)。

Gnus 固有ではないけれど、役に立つかもしれない文字セットに関する他の秘訣:

もし、同一の Emacs の文字セットをエンコードするMIME の文字セットが複数あるのならば、以下の宣言を使うことによって、使う文字セットを選択することができます:

```
(put-charset-property 'cyrillic-iso8859-5
                      'preferred-coding-system 'koi8-r)
```

これは、ロシア語がディフォルトのiso-8859-5 MIME 文字セットの代わりに、koi8-r でエンコードされることを意味します。

メッセージをkoi8-u で読みたいのであれば、以下のように騙すことができます。

```
(define-coding-system-alias 'koi8-u 'koi8-r)
```

これは、ほとんど正しいことをするでしょう。

そして最後に、windows-1251 のような文字セットを読むには、次のように宣言すれば良いでしょう(訳注: Emacs の版によっては、windows-1251 が最初から実装されています)。

```
(codepage-setup 1251)
(define-coding-system-alias 'windows-1251 'cp1251)
```

4.21 記事命令

A P 記事バッファーのポストスクリプト(PostScript) イメージを作成して印刷します(gnus-summary-print-article)。gnus-ps-print-hook がバッファーを印刷する直前に実行されます。他に Muttprint を使って印刷することもできます(see Section 4.16 [Saving Articles], p. 81)。

A C もし<backend>-fetch-partial-articles がnil 以外の値だったら、バックエンドがサポートしていれば Gnus は記事の部分を取り込みます。今のところnnimap だけが行ないます。もし記事の部分を見ているときに、そうではなく完全な記事を見る必要があるならば、*A C* コマンド(gnus-summary-show-complete-article)がそうします。

w
A w 記事バッファーを走査してbrowse-url で閲覧できるリンクを提示します。接頭引数を与えると代わりにbrowse-url-secondary-browser-function で閲覧します。

4.22 概略の並べ替え

私はどうしてあなたがそうしたいのかはわからないのですが、それでもあなたはたくさんの方法で概略バッファーを並べ替えることができます。

C-c C-s C-n
記事番号によって並べ替えます(gnus-summary-sort-by-number)。

C-c C-s C-m C-n
最新の記事番号によって並べ替えます(gnus-summary-sort-by-most-recent-number)。

C-c C-s C-a
著者によって並べ替えます(gnus-summary-sort-by-author)。

C-c C-s C-t
受信者によって並べ替えます(gnus-summary-sort-by-recipient)。

C-c C-s C-s
表題によって並べ替えます(gnus-summary-sort-by-subject)。

C-c C-s C-d
日付によって並べ替えます(gnus-summary-sort-by-date)。

C-c C-s C-m C-d
最新の日付によって並べ替えます(gnus-summary-sort-by-most-recent-date)。

C-c C-s C-l
行数によって並べ替えます(gnus-summary-sort-by-lines)。

C-c C-s C-c
記事の長さ(文字数) で並べ替えます(gnus-summary-sort-by-chars)。

C-c C-s C-m C-m
記事の購読度の印(readedness marks) で並べ替えます(gnus-summary-sort-by-marks)。

C-c C-s C-i
スコアによって並べ替えます(gnus-summary-sort-by-score)。

C-c C-s C-u
ニュースグループで並べ替えます(gnus-summary-sort-by-newsgroups)。

C-c C-s C-x
: 入力したヘッダーで並べ替えます(gnus-summary-sort-by-extra)。そのヘッダーのために定義された並べ替え関数が無い場合はエラーになります。

C-c C-s C-r
: ランダムに並べ替えます(gnus-summary-sort-by-random)。

C-c C-s C-o
: ディフォルトの方法で並べ替えます(gnus-summary-sort-by-original)。

これらの関数はスレッドを使っているときと使っていないときの両方で動作します。後者では、すべての概略行が一行一行並べ替えられます。前者では根本だけに基づいて並べ替えられ、それはあなたが求めていることとは異なっているかもしれません。スレッドを使うかどうかを切り替えるには*T T* を打ってください(see Section 4.9.2 [Thread Commands], p. 75)。

接頭引数を与えると並べ替えの順序が逆になります。

4.23 親記事を探す

^
: 現在の記事の親記事を読みたいのに、それが概略バッファーに表示されていなくても、おそらくそれは可能でしょう。というのは、現在のグループがNNTP で取得されていて、親がまだ期限切れ消去されていない上、現在の記事のReferences がぶち壊されていなければ、ただ^ か*A r* を押せば良いだけですから(gnus-summary-refer-parent-article)。すべてがうまくいけば、親記事を取得できるでしょう。もし親記事がすでに概略バッファーに表示されているのであれば、ポイントがその記事に移動するでしょう。

  正の数値接頭引数を与えられると、その数の祖先たちを遡って取得します。負の数値接頭引数が与えられた場合は、その数の世代だけ前の祖先の記事のみを取得します。ですから*3* ^ とすれば、Gnus は現在の記事の親と祖父母と曾祖父母を取得します。*-3* ^ とすれば、Gnus は現在の記事の曾祖父母だけを取得します。

A R (概略)
: 記事のReferences 欄にあるすべての記事を取得します(gnus-summary-refer-references)。

A T (概略)
: 現在の記事があるスレッドの、全部の記事を表示します(gnus-summary-refer-thread)。ディフォルトでは、このコマンドは現在のグループでのみ記事を検索します。一部のバックエンド(現在はnnimap のみ) は、スレッド内の記事を直接検索する方法を知っています。他の場合には、現在のグループの各ヘッダーを取り込んで検査しなければならないので、普通は時間がかかります。これをしばしば行なうのであれば、gnus-fetch-old-headers をinvisible に設定することを考えたほうが良いでしょう(see Section 4.9.1.2 [Filling In Threads], p. 72)。これは普通は視覚的な効果はありませんが、この命令の動作をかなり速くします。もちろんグループに入るのはいくらか遅くなりますが。

  もしgnus-refer-thread-use-search がnil でなければ、スレッドを直接検索する方法を知っているバックエンドは、現在のグループだけでなく、同じサーバー上のすべてのグループも検索します。

  変数gnus-refer-thread-limit はこの命令を実行するときにどのくらい古い(すなわち、現在のグループで最初に表示されたものよりも前の記事の) ヘッ

ダーを取得するかを指定します。ディフォルトは 200 です。もしt であれば、取得可能なすべてのヘッダーを取得します。A T 命令に数値接頭引数を与えると、代わりにそれが使われます。

ほとんどの場合gnus-refer-thread は見つけた記事を現在の概略バッファーに追加します。(gnus-refer-thread-use-search が真で、最初の参照が仮想ではないグループの概略バッファーから始まる場合、これはできないかもしれません。その場合は、スレッド検索の結果を持つ仮想グループの、新しい概略バッファーが作成されます。) gnus-refer-thread-limit-to-thread がnil でない場合、概略バッファーはスレッド内の記事に限定されます。

*M-^* (概略)

どのグループに属しているかに関わらず、任意の記事を Gnus に要求することができます。*M-^* (gnus-summary-refer-article) はMessage-ID、つまりあの長くてなかなか読むことのできない‘<38o6up$6f2@hymir.ifi.uio.no>’ のようなものをあなたに尋ねます。あなたはすべてを正確に打ち込まなければなりません。残念ながら、あいまいな検索はできないのです。

Gnus はすでに取得してあるヘッダーたちの中でMessage-ID を探しますが、見つからなかったらgnus-refer-article-method に設定されているすべての選択方法を試してもみます。

もしあなたの読んでいるグループがMessage-ID での取得があまり良くできないようなバックエンド(nnspool など) であるのなら、gnus-refer-article-method をNNTP の選択方法に設定すれば良いでしょう。おそらく、あなたが問い合わせるNNTP サーバーがあなたの読んでいるスプールを更新していると最も良いでしょう。しかし、それはどうしても必要なわけではありません。

それは選択方法のリストのみならず、現在の選択方法を意味する特別なシンボルcurrent であることもできます。Gnus は合うものを発見するまでそれらすべての方法を試します。

これは現在の選択方法を試して、それが失敗した場合には Google に訊く設定の例です:

```
(setq gnus-refer-article-method
      '(current
        (nnweb "google" (nnweb-type google))))
```

ほとんどのメールバックエンドはMessage-ID での取得が可能ですが、あまり優雅な方法でやっているわけではありません。nnmbox, nnbabyl, nnmaildir およびnnml がどのグループからでも記事を捜索できるのに対して、nnfolder とnnimap は現在のグループに投稿された記事しか探すことができません。nnmh ではまったく不可能です。

幸いにも特別なバックエンドであるnnregistry は、どんなグループにある記事でもバックエンドに関係無く見つけ出すことができます。See Section 10.20.2 [Registry Article Refer Method], p. 320.

4.24 代替手段

ニュースを読む方法の好みは人それぞれです。これは Gnus なのですから、概略バッファーのためのマイナーモードに少しばかり選択肢を設けます。

4.24.1 選んで読む

いくつかのニュースリーダー(nn や、ええと VM/CMS のNetnews など) は二段階の購読インターフェースを使います。利用者はまず概略バッファーで読みたい記事に印を付けます。それから、記事バッファーだけを表示して記事を読みます。

Gnus はこれをするための概略バッファーマイナーモードを提供します—gnus-pick-mode です。これは、基本的には簡単に印を付けられるように少数のプロセス印命令を一個のキーだけで済む命令にして、概略バッファーへ切り替えるための追加の命令を一つ提供します。

訳注: Pick マイナーモードを有効にするには、以下のフックを使ってください:

```
(add-hook 'gnus-summary-mode-hook 'gnus-pick-mode)
```

そうせずに、概略バッファーに入ってから*M-x gnus-pick-mode* を実行しても、うまくいかないようです。

これらが pick mode で使うことができるキーです:

`.` 現在の行の記事かスレッドを選択、またはすでに選択されている場合は選択を解除します(gnus-pickd-article-or-thread)。変数gnus-thread-hide-subtreeがnil でないならば、このキーがスレッドの最初の記事で使われるとスレッド全体を選択します。そうでなければ、その記事だけを選択します。もし数値接頭引数を与えられると、その番号のスレッドか記事に移動して、それを選択します。(普通は行番号が概略行の最初に表示されます。) gnus-process-mark-toggleがnil である場合は、記事またはスレッドを選択します。

`SPC` 概略バッファーを一ページ次にスクロールします(gnus-pick-next-page)。もしバッファーの最後であれば、選択した記事を読み始めます。

u スレッドか記事を未選択にします(gnus-pick-unmark-article-or-thread)。変数gnus-thread-hide-subtree が非-nil だったら、このキーがスレッドの最初で使われるとそのスレッドを未選択にします。そうでなければ、その記事だけを未選択にします。その行にあるスレッドか記事を未選択にするために、このキーに数値接頭引数を与えることができます。

`RET` 選択された記事を読み始めます(gnus-pick-start-reading)。接頭引数が与えられると、最初にすべての未選択記事に既読の印を付けます。gnus-pick-display-summary がnil でないと、概略バッファーは読んでいる間も表示されます。

すべての普通の概略モード命令は pick-mode でも使用可能ですが、*u* は例外です。それでも、同じ関数gnus-summary-tick-article-forward に割り当てられている*!* を使うことができます。

これが良さそうだと思ったら、次のようにしてください:

```
(add-hook 'gnus-summary-mode-hook 'gnus-pick-mode)
```

gnus-pick-minor-mode-hook は pick マイナーモードのバッファーで実行されます。

gnus-mark-unpicked-articles-as-read が非-nil だったら、選択されなかったすべての記事に既読の印を付けます。ディフォルトはnil です。

pick モードでの概略行の様式は標準の様式とは少し違います。それぞれの行の最初に行数が表示されます。Pick モードの行の様式は変数gnus-summary-pick-line-format で制御されます(see Section 10.4 [Formatting Variables], p. 272)。これはgnus-summary-line-formatと同じ様式指定を受け付けます(see Section 4.1.1 [Summary Buffer Lines], p. 47)。

4.24.2 バイナリーグループ

多くの時間をバイナリーグループで過ごしているのなら、いつも*X u*, *n*, RET を叩くのが嫌になっているでしょう。*M-x gnus-binary-mode* は、単に記事を普通の方法で表示する代わりに、記事を選択するための普通の Gnus の関数を、一連の記事を uudecode してその結果を表示するように変更する、概略バッファーのためのマイナーモードです。

現実には、このモードにしたときに、実際に記事を見るための唯一の命令が*g* です(gnus-binary-show-article)。

gnus-binary-mode-hook がバイナリーマイナーモードのバッファーで呼ばれます。

4.25 木表示

もし普通の Gnus の概略表示を好きでないならば、gnus-use-trees をt に設定してみると良いかもしれません。これは(ディフォルトで) 追加の「木バッファー」(tree buffer) を作成します。木バッファーではすべての概略モード命令を実行することができます。

もちろん、木表示をカスタマイズする変数が少しあります:

gnus-tree-mode-hook
: すべての木モードのバッファーで実行されるフックです。

gnus-tree-mode-line-format
: 木モードのバッファーにおけるモード行のためのフォーマット文字列です(see Section 10.4.2 [Mode Line Formatting], p. 273)。ディフォルトは‘Gnus: %%b %S %Z’ です。使用可能な指定はSection 4.1.3 [Summary Buffer Mode Line], p. 52, を参照してください。

gnus-selected-tree-face
: 木バッファーで選択された記事をハイライトするために使われるフェースです。ディフォルトではmodeline です。

gnus-tree-line-format
: 木の節のためのフォーマット文字列です。でもこれは少し誤った名称です---それは行ではなく、ただ節を定義するだけです。ディフォルトの値は‘%(%[%3,3n%]%)’ で、それは投稿者の名前の最初の三文字を表示します。すべての節が同じ長さであることが重要なので、‘%4,4n’ のような指定を使わなければなりません。

  有効な指定は:

  ‘n’ 投稿者の名前。

  ‘f’ From 欄。

  ‘N’ 記事の番号。

  ‘[’ 開き括弧。

  ‘]’ 閉じ括弧。

  ‘s’ 表題。

  See Section 10.4 [Formatting Variables], p. 272.

  表示に関連した変数は:

  gnus-tree-brackets
  : これは『本当の』記事と『まばら』な記事に違いを付けるために使われます。様式は

```
((本当の開 . 本当の閉)
 (まばら開 . まばら閉)
 (偽の開 . 偽の閉))
```

となっていて、ディフォルトは`((?[ . ?]) (?( . ?)) (?{ . ?}) (?< . ?>))` です。

`gnus-tree-parent-child-edges`
: これは親の節を子に接続するために使われる文字を含むリストです。ディフォルトは`(?- ?\\ ?|)` です。

`gnus-tree-minimize-window`
: もしこの変数が`nil` でないと、他の Gnus ウィンドウがもっと場所を取れるように Gnus は木バッファーをできるだけ小さくします。もしこの変数が数値であると、木バッファーの高さはその数値より大きくなることはありません。ディフォルトは`t` です。フレームでいくつかのウィンドウが横に並んで表示されていて、木バッファーがそのうちの一つである場合、木ウィンドウを最小化することはその隣に表示されているすべてのウィンドウの大きさをも変更することに注意してください。

  以下のフックを追加して、いつでも木ウィンドウを最小化するようにしても良いでしょう。

```
(add-hook 'gnus-configure-windows-hook
          'gnus-tree-perhaps-minimize)
```

`gnus-generate-tree-function`
: 実際にスレッドの木を作成する関数です。二つの定義済みの関数`gnus-generate-horizontal-tree` および`gnus-generate-vertical-tree` (これがディフォルトです) が利用可能です。

水平木バッファー(horizontal tree buffer) の例です:

```
{***}-(***)-[odd]-[Gun]
     |      \[Jan]
     |      \[odd]-[Eri]
     |      \(***)-[Eri]
     |            \[odd]-[Paa]
     \[Bjo]
     \[Gun]
     \[Gun]-[Jor]
```

同じスレッドが垂直木バッファー(vertical tree buffer) で表示されたものです:

```
{***}
  |-------------------------\-----\-----\
(***)                       [Bjo] [Gun] [Gun]
  |--\-----\-----\                      |
[odd] [Jan] [odd] (***)                [Jor]
  |           |     |--\
[Gun]       [Eri] [Eri] [odd]
                        |
                      [Paa]
```

もし水平木を使っているのなら、概略バッファーで木を隣り合わせで表示できれば嬉しいでしょう。次のようなものを~/.gnus.el ファイルに加えることができます:

```
(setq gnus-use-trees t
      gnus-generate-tree-function 'gnus-generate-horizontal-tree
      gnus-tree-minimize-window nil)
(gnus-add-configuration
 '(article
   (vertical 1.0
             (horizontal 0.25
                         (summary 0.75 point)
                         (tree 1.0))
             (article 1.0))))
```

See Section 10.5 [Window Layout], p. 276.

4.26 メールグループ命令

いくつかの命令はメールグループでのみ意味を持ちます。これらの命令が現在のグループで有効でないなら、それらは大騒ぎをしてあなたに知らせるでしょう。

これらすべての命令は(期限切れ消去と編集命令は除く) プロセス/接頭引数の習慣を使います(see Section 10.1 [Process/Prefix], p. 271)。

B e 現在のグループのすべての期限切れ消去可能な記事について、期限切れ消去の処理(gnus-summary-expire-articles) を行ないます。これは、そのグループにしばらく存在していた期限切れ消去可能なすべての記事を消去するということです。(see Section 7.4.9 [Expiring Mail], p. 181)。

B C-M-e グループのすべての期限切れ消去可能な記事を削除します(gnus-summay-expire-articles-now)。これは、現在のグループにあるすべて の期限切れ消去可能な記事が、永遠に空の大きな/dev/null へ消え去るということです。

B DEL メール記事を削除します。これは『あなたのディスクから永久に削除して二度と戻らない』の意味の『削除』です。注意して使ってください(gnus-summary-delete-article)。

B m あるメールグループから別のメールグループへ記事を移動します(gnus-summary-move-article)。gnus-preserve-marks の値がnil でなければ(それがディフォルト)、印は保存されます。

B c あるグループ(メールグループや他のもの) からメールグループに記事をコピーします(gnus-summary-copy-article)。gnus-preserve-marks の値がnil でなければ(それがディフォルト)、印は保存されます。

B B 現在の記事を他のグループにクロスポストします(gnus-summary-crosspost-article)。これは他のグループの記事の新しい複製を作成し、記事の Xref 欄も適切に更新されます。

B i 任意のファイルを現在のメールグループに取り込みます(gnus-summary-import-article)。あなたはファイル名と、From 欄とSubject 欄の入力を促されます。

B I 空の記事を現在のメールグループに作ります(gnus-summary-create-article)。From ヘッダーとSubject ヘッダーの内容を尋ねられます。

B r メール記事をスプールし直します(gnus-summary-move-article)。gnus-summary-respool-default-method が再スプールするときのディフォルトの選択方法として使用されます。この変数はディフォルトではnil で、その場合は現在のグループの選択方法が代わりに使われます。gnus-preserve-marks の値がnil でなければ(それがディフォルト)、印は保存されます。

訳注:「スプールし直す」というのはメールの分割(Section 7.4.3 [Splitting Mail], p. 164, またはSection 7.4.6 [Fancy Mail Splitting], p. 175) の規則に基づいて、メールを適切なグループに入れ直すことです。そのグループに間違って入ってしまったメールを、分割の規則を修正した後で、正しいグループに移動させる場合などに使います。この章の*B q* と*B t* も見てください。

B w
e 現在の記事を編集します(gnus-summary-edit-article)。編集を終了して変更を固定するには*C-c C-c* (gnus-summary-edit-article-done) を打ちます。もし*C-c C-c* 命令に接頭引数を与えると、Gnus は記事を再ハイライトしません。

訳注: 変更しないで編集を終るには、*C-c C-k* をタイプしてください。

B q 記事を再スプールするときは、再スプールをする前にどのグループに記事が移るかを知りたいでしょう。この命令でそれがわかります(gnus-summary-respool-query)。

B t 同様に、この命令は再スプールするときに使われるすべての特級分割方式を、もしあれば表示します(gnus-summary-respool-trace)。

B p 一部の人たちには、あなたが投稿した記事にフォローアップするときに「親切な」複製を送る傾向があります。これらは普通はそこにNewsgroups ヘッダーが付いているのですが、いつもそうであるとは限りません。この命令(gnus-summary-article-posted-p) は現在の記事をあなたのニュースサーバーから(というよりは、むしろgnus-refer-article-method やgnus-select-method から) 取得しようとして、記事を発見できたかどうかを報告します。それが記事を発見しなかったとしても、それはとにかく投稿されているかもしれません---メールの伝達はニュースの伝達よりもずっと速いので、ニュースの複製がまだ到着していないだけかもしれないのです。

訳注: その「親切な」複製が、概略バッファーで独立した記事として見えていないと検査することができないので、そうするために*A D* または*C-d* 命令(see Section 4.27.4 [Really Various Summary Commands], p. 118) を使う必要があるかもしれません。この命令はとにかくすべての選択方法を試すので、特にそれらに遅いものが含まれているときは、注意して使ってください。

K E 記事の本文を暗号化します(gnus-article-encrypt-body)。本文は、変数gnus-article-encrypt-protocol で指定されたプロトコルで暗号化されます。

いつも記事をどこかに移動(もしくは複製) することを習慣にしているのならば、記事をどこに入れれば良いかを Gnus に提案してもらいたいと思うでしょう。gnus-move-split-methods はgnus-split-methods と同じ構文を使う変数です(see Section 4.16 [Saving Articles], p. 81)。あなたが妥当だと思うような提案をするようにその変数をカスタマイズすることができます。(gnus-split-methods がファイル名を使うのに対してgnus-move-split-methods はグループ名を使うことに注意してください。)

```
(setq gnus-move-split-methods
```

```
'(("^From:.*Lars Magne" "nnml:junk")
  ("^Subject:.*gnus" "nnfolder:important")
  (".*" "nnml:misc")))
```

4.27 概略のいろいろなもの

`gnus-summary-display-while-building`
: 非-`nil` だったら、構築中の概略バッファーを更新しながら表示します。`t` だった場合は、行が挿入される度に毎回バッファーを更新します。値が整数n であった場合は、n 行毎に表示を更新します。ディフォルトは`nil` です。

`gnus-summary-display-arrow`
: 非-`nil` だったら、現在の記事を指し示すためにフリンジに矢印を表示します。(訳注: フリンジとは Emacs 21 以上でウィンドウの左右に現れる余白のことです。)

`gnus-summary-mode-hook`
: 概略モードのバッファーを作成するときにこのフックが呼ばれます。

`gnus-summary-generate-hook`
: これはスレッド作成と概略バッファー作成の前に実行する最後のものとして呼ばれます。これはニュースグループの持っているデータに基づいてスレッドの変数をカスタマイズするのに非常に便利です。このフックはほとんどの概略バッファー変数が設定された後に概略バッファーから呼ばれます。

`gnus-summary-prepare-hook`
: これは概略バッファーが作成された後に呼ばれます。例えば、これを何かしら神をも畏れぬ方法で行をハイライトしたり、バッファーの見え方を修正したりするのに使ったりするかもしれません。

`gnus-summary-prepared-hook`
: 概略バッファーが作成された後で一番最後に呼ばれるフックです。

`gnus-summary-ignore-duplicates`
: Gnus が同じ`Message-ID` を持つ二つの記事を発見したときは、何か思い切ったことをしなければなりません。別の記事が同じ`Message-ID` を持つことは許されていませんが、それは何らかのソースからメールを読んでいるときに起こるかもしれません。この変数によって Gnus が何をするかをカスタマイズできるようになっています。`nil` だったら(それがディフォルトです)、Gnus は(表示のためだけに) `Message-ID` を付け替えて、その記事を他の記事と同じように表示します。`t` にすると、それは記事を表示しません---最初から存在しなかったかのように。

`gnus-alter-articles-to-read-function`
: この変数に設定した関数で、選択する記事のリストを変更することができます。関数は二つの引数(グループ名と選択する記事のリスト) を受け付けます。

  例えば以下の関数は、キャッシュされた記事のリストを、あるグループのリストだけに追加します。

```
(defun my-add-cached-articles (group articles)
  (if (string= group "some.group")
```

```
    (append gnus-newsgroup-cached articles)
  articles))
```

gnus-newsgroup-variables
: ニュースグループ(その概略バッファーの) のローカル変数、または変数とそれらの評価されるディフォルトの表現(ディフォルト値がnil でない場合) の cons セルのリストで、その概略バッファーが活きている間はグローバル変数になります。(訳注: いわゆるバッファーローカル変数ではありません。)

  注: ディフォルトの表現は単にローカル変数に設定されるのではなく、その前に(eval 関数を使って) 評価されます。ディフォルトの表現がglobal というシンボルだった場合は評価されず、代わりにそのローカル変数のグローバル値が使われます。

  これらグループパラメーターの値が他のバッファーで行なわれる処理に影響するようになっていても、(訳注: その概略バッファーの) グループパラメーターを設定するために使うことができます。例です:

```
(setq gnus-newsgroup-variables
     '(message-use-followup-to
       (gnus-visible-headers .
        "^From:\\|^Newsgroups:\\|^Subject:\\|^Date:\\|^To:")))
```

  Section 3.10 [Group Parameters], p. 23, も参照してください。

  訳注: もっと良い例が必要です。gnus-newsgroup-variables およびgnus-parameters (see Section 3.10 [Group Parameters], p. 23) の値を次のように設定したとしましょう:

```
(setq gnus-newsgroup-variables '((var . foo)))
(setq gnus-parameters
      '(("^fj\\." (var . bar))
        ("^japan\\." (var . baz))))
```

  こうしておくと変数*var* の値が、'fj' 階層のニュースグループ(の概略バッファー) に入ると*bar* になり、'japan' 階層のグループに入ると*baz* になります。グループを抜けても変数*var* の値は変化しませんが、'fj' または'japan' 階層以外のグループに入ると変数*var* の値は*foo* になります(正確には、*foo*, *bar* または*baz* の値は、本編で説明されているようにeval した結果が使われます)。

  通常のグループパラメーターは、そのグループの概略バッファーでだけ値を知ることができるのに対して、gnus-newsgroup-variables で設定した変数は、同じ Emacs のどのバッファーでも、現在選択されているグループ固有の値を持つ点が違います。異なる複数のグループの概略バッファーを使う場合には、注意する必要があります。

  特別な場合として*foo* がnil で良い場合は、次のように記述することができます:

```
(setq gnus-newsgroup-variables '(var))
(setq gnus-parameters
      '(("^fj\\." (var . bar))
        ("^japan\\." (var . baz))))
```

  gnus-newsgroup-variables およびgnus-parameters はどちらもリストなので、setq よりはむしろadd-to-list やpush などを使って、値を「追加」した方が便利かもしれません。

4.27.1 概略グループ情報

H d 現在のグループの簡潔な説明を表示します(gnus-summary-describe-group)。接頭引数が与えられると、サーバーから強制的に説明の再読み込みをします。

H h 最も重要な概略コマンドの、非常に簡潔な説明を表示します(gnus-summary-describe-briefly)。

H i Gnus の info の節(node) に移動します(gnus-info-find-node)。

4.27.2 記事を探す

M-s M-s それ以降のすべての(生の) 記事を正規表現で検索します(gnus-summary-search-article-forward)。

M-s M-r それ以前のすべての(生の) 記事を正規表現で検索します(gnus-summary-search-article-backward)。

M-S 前回の前方検索を繰り返します(gnus-summary-repeat-search-article-forward)。

M-R 前回の後方検索を繰り返します(gnus-summary-repeat-search-article-backward)。

& この命令は、ヘッダー、そのヘッダーの内容に合致する正規表現、および合致したときに実行されるコマンドの入力を要求します(gnus-summary-execute-command)。ヘッダーが空文字列だったら、記事全体で合致するものを探します。接頭引数を与えられると、代わりに後ろ向きに探します。

例えば*&* RET 何かの.* 文字列RET # は、ヘッダーか本文に'何かの.* 文字列' を持つすべての記事にプロセス印を付けます。

M-& この命令に続けて入力する命令を、プロセス印が付けられているすべての記事で実行します(gnus-summary-universal-argument)。

4.27.3 概略生成命令

Y g 現在の概略バッファーを再作成します(gnus-summary-prepare)。

Y c (現在のグループのために) キャッシュされたすべての記事を概略バッファーに挿入します(gnus-summary-insert-cached-articles)。

Y d (現在のグループのための) すべての保留記事を概略バッファーに挿入します(gnus-summary-insert-dormant-articles)。

Y t (現在のグループのための) すべての可視記事を概略バッファーに挿入します(gnus-summary-insert-ticked-articles)。

4.27.4 本当にいろいろな概略命令

A D
C-d 現在の記事が別の記事を寄せ集めたもの(例えばダイジェスト) であるならば、それらの記事でできているグループに入るためにこの命令を使うことができます(gnus-summary-enter-digest-group)。この命令に接頭引数を与えないとGnus はどのような型の記事が現在表示されているかを推測しようとし、実際にはそれが『ダイジェスト』であるものとして強引に解釈します。基本的に、ある

様式で寄せ集められた別のメッセージを見るときはいつでも、*C-d* を使うことによって、もっと便利なやり方でそれらのメッセージを読むことができます。

変数gnus-auto-select-on-ephemeral-exit はダイジェスト・グループを出た後に、どの記事を選択すべきかを制御します。有効な値は次の通りです:

`next` 次の記事を選択します。

`next-unread`
次の未読記事を選択します。

`next-noselect`
カーソルを次の記事に移動します。これがディフォルトです。

`next-unread-noselect`
カーソルを次の未読記事に移動します。

これら以外の値だったり、次の(未読の) 記事が無かったら、ダイジェスト・グループに入る前に選択されていた記事が現れます。

C-M-d この命令は上のものによく似ていますが、いくつかの文書を一つのおおーきなグループに集めます(gnus-summary-read-read-document)。それを実現するために、この命令はそれぞれの文書のためのnndoc グループを開いてから、それら複数のnndoc グループのてっぺんでnnvirtual グループを開きます。この命令はプロセス/接頭引数の習慣を理解します(see Section 10.1 [Process/Prefix], p. 271)。

C-t 長い概略行を切り詰めるかどうかを切り替えます(gnus-summary-toggle-truncation)。これはおそらく概略バッファーで行を中央に表示する機能を混乱させるので、記事を読んでいるときに行の切り詰めを off にするのは良い考えではないでしょう。

= 概略バッファーのウィンドウを拡大します(gnus-summary-expand-window)。接頭引数を与えられると、記事バッファーのためのウィンドウの配置の設定を強制します(訳注: ディフォルトでは記事バッファーのためのウィンドウの配置の設定には概略バッファーを表示することも含まれているので、普通に記事を読んでいるときと同じになるでしょう)。

C-M-e 現在のグループのグループパラメーター(see Section 3.10 [Group Parameters], p. 23) を編集します(gnus-summary-edit-parameters)。

C-M-a 現在のグループのグループパラメーター(see Section 3.10 [Group Parameters], p. 23) をカスタマイズします(gnus-summary-customize-parameters)。

4.28 概略バッファーを抜ける

概略バッファーから抜けると、普通はグループのすべての情報を更新してグループバッファーに戻ります。

Z Z
Z Q
q 現在のグループを出て、グループのすべての情報を更新します(gnus-summary-exit)。抜け出るための多くの処理を行なう前にgnus-summary-prepare-exit-hook が呼ばれ、それはディフォルトでgnus-summary-expire-articles を呼

びます。抜け出るための処理を終えた後でgnus-summary-exit-hook が呼ばれます。グループモードに戻るときに(未読の) グループが残っていなかったらgnus-group-no-more-groups-hook が実行されます。

Z E
Q グループのどんな情報も更新せずに現在のグループを抜け出ます(gnus-summary-exit-no-update)。

Z c
c グループのすべての可視ではない(unticked) 記事に既読の印を付けてから抜けます(gnus-summary-catchup-and-exit)。

Z C 可視記事さえも含むすべての記事に既読の印を付けてから抜けます(gnus-summary-catchup-all-and-exit)。

Z n すべての記事に既読の印を付けて次のグループへ移動します(gnus-summary-catchup-and-goto-next-group)。

Z p すべての記事に既読の印を付けて前のグループへ移動します(gnus-summary-catchup-and-goto-prev-group)。

Z R
C-x C-s 現在のグループを出て、それから入り直します(gnus-summary-reselect-current-group)。接頭引数が与えられると、既読と未読の両方のすべての記事を選択します。

Z G
M-g グループを抜け、そのグループの新しい記事を調べてから、再びそのグループを選択します(gnus-summary-rescan-group)。接頭引数が与えられると、既読と未読の両方のすべての記事を選択します。

Z N グループを抜けて、次のグループへ移動します(gnus-summary-next-group)。

Z P グループを抜けて、前のグループへ移動します(gnus-summary-prev-group)。

Z s 現在の既読と印付き記事の数をドリブルバッファー(dribble buffer) に保存し、それからドリブルバッファーを保存します(gnus-summary-save-newsrc)。接頭引数が与えられると.newsrc ファイル(と.newsrc.eld ファイル) も保存します。この命令を使うと、更新なしで抜け出ること(*Q* 命令) は意味が無くなります。

グループのすべての情報を「更新」して現在のグループを抜けるときにgnus-exit-group-hook が呼ばれます。例えば*Q* 命令(gnus-summary-exit-no-update) はこのフックを呼びません。

グループを抜けた後でそれを後悔する癖があるのなら、gnus-kill-summary-on-exit をnil に設定と良いかもしれません。そうすると Gnus は抜け出るときに概略バッファーを削除しません。(何という驚き!) 代わりに、それはバッファーの名前を*Dead Summary ... *のようなものに変更して、gnus-dead-summary-mode というマイナーモードを導入します。今やそのバッファーに切り替えると、すべてのキーが関数gnus-summary-wake-up-the-dead に割り当てられていることに気付くでしょう。死んだ概略バッファー(dead summary buffer) でどんなキーでも叩くと、それは生きた普通の概略バッファーになります。

死んだ概略バッファーは同時に一つしか存在することはできません。

概略バッファーを抜け出ると、現在のグループのデータ(どの記事を読んで、どの記事に返答したか、など) は更新されます。もし変数`gnus-use-cross-reference` が`t` であると(それがディフォルトです)、そのグループに相互参照された(cross referenced) 記事には、それがクロスポストされた他の購読しているグループにあっても、既読の印が付きます。この変数が`nil` でも`t` でもなければ、記事には購読しているグループと購読していないグループの両方で既読の印が付きます(see Section 4.29 [Crosspost Handling], p. 121)。

4.29 クロスポストの扱い

クロスポストされた記事に既読の印を付けることによって、同じ記事を二回以上読まないで済むことを保証します。もちろん、だれかがそれを複数のグループに別々に投稿しない限りは。同じ記事を複数のグループに(クロスポストではではなく) 投稿することは*spamming* と呼ばれ、あなたはそのような憎むべき犯罪を行なうものに対して、法律によって不快な記事を送ることが義務づけられています。

覚えておいてください: クロスポストはまあ構いませんが、同じ記事を別々に複数のグループに投稿するのは許されません。大量のクロスポスト(*velveeta* として知られているもの) は何としても避けられるべきで、過剰なクロスポストに対して不満を言うために`gnus-summary-mail-crosspost-complaint` 命令を使うことさえできます。

Gnus にクロスポストを正しく扱えなくさせる原因の一つは、XOVER (これは非常に良いです、というのはそれは速度をとても速くするからです) をサポートしているけれどもNOV 行に`Xref` 欄を含めないNNTP サーバーを使っていることです。これは害悪です。でも、あぁ、悲しいかな、非常に良くあることなのです。Gnus はあなたが読んだすべての記事に`Xref` 行を記録することによって The Right Thing (正しいこと) をしようとしますが、記事を削除したり単に読まないで既読の印を付けると、Gnus がこれらの記事の`Xref` 行をのぞきまわる機会が無くなってしまうので、相互参照(cross-reference) の機構を使えなくなってしまいます。

あなたのNNTP サーバーがその概観ファイル(overview file) に`Xref` 欄を含めるかどうかを調べるには、'`telnet your.nntp.server nntp`' をタイプして、`inn` サーバーでは'`MODE READER`' コマンドを与えてから、'`LIST overview.fmt`' を試してください。これは動作しないかもしれません。しかし、もし動作して、取得した最後の行が'`Xref:full`' でないならば(訳注: 最後の行ではないかもしれません)、ニュースの管理者が概観ファイルに`Xref` 欄を含めるようにしてくれるまで、彼女に向かって叫び、泣き付くべきでしょう。

Gnus にいつでも正しい`Xref` を取得するようにさせたいのであれば、`nntp-nov-is-evil` を`t` にする必要があり、それは非常に速度を遅くします。Section 12.5.1 [Slow/Expensive Connection], p. 359, も参照してください。

ま、人生はそのようなものです。

代替手段に付いてはSection 4.30 [Duplicate Suppression], p. 121, を参照してください。

4.30 重複の抑制

ディフォルトでは Gnus はクロスポスト機構を利用することによって、同じ記事を二回以上読まないようにしようとします(see Section 4.29 [Crosspost Handling], p. 121)。しかし、その単純で効果的な方法は、いろいろな理由により、満足する結果をもたらさないかもしれません。

1. NNTP サーバーは`Xref` 欄の生成に失敗するかもしれません。これは悪いことで、あまり起こりません。

2. NNTP サーバーは.overview データベースにXref 欄を含めるのに失敗するかもしれません。これは悪いことで、非常に良くあることです、あぁ悲しい。
3. 同じグループ(もしくはいくつかの関連したグループ) を違ったNNTP サーバーから読んでいるかもしれません。
4. グループに投稿された記事と重複するメールを受け取ったかもしれません。

Xref の扱いに失敗する状況は確かに他にもありますが、これら四つが最も良くある状況です。

もし、本当にもしもXref の扱いに失敗したら、「重複抑制」に切り替えることを考慮する必要があるかもしれません。そうすれば、Gnus はあなたが読んだすべての記事、あるいは既読の印を付けたすべての記事のMessage-ID を記憶し、そしてまるで魔法のように、以後それらを読むときはいつでも既読の印が付いているようにします—すべて のグループで。この機構を使うのは何だかとても非効率になりそうですが、過度に非効率なわけではありません。同じ記事を二回以上読むよりは、間違い無く望ましいです。

重複抑制はあまり精密な道具ではありません。どちらかというと大槌のようなものです。それは非常に単純なやり方で動作しています---あなたが記事に既読の印を付けると、その Message-ID をキャッシュに加えます。次にその Message-ID に出会うと、'M' 印によって記事に既読の印を付けます。その記事をどのグループで見たかは気にしません。

gnus-suppress-duplicates
: nil でなければ、重複抑制をします。

gnus-save-duplicate-list
: nil でなければ、重複のリストをファイルに保存します。これは起動と終了の時間を長くするので、ディフォルトはnil です。しかし、これは Gnus を一回実行したときに読まれた重複記事だけが抑制されるということです。

gnus-duplicate-list-length
: この変数はどのくらい多くのMessage-ID を重複抑制リストに保っておくかを決定します。ディフォルトは 10000 です。

gnus-duplicate-file
: 重複抑制のリストを格納しておくファイルの名前です。ディフォルトは~/News/suppression です。

何度も Gnus を終了して起動する傾向があるのであれば、おそらくgnus-save-duplicate-list をt にするのは良い考えでしょう。もし Gnus を続けて何週間も走らせておくのであれば、それをnil にした方が良いかもしれません。一方、リストを保存することは起動と終了をずっと遅くするので、頻繁に Gnus を終了して起動するのであれば、gnus-save-duplicate-list をnil に設定するべきです。うーむ。私はあなたがどうするかに任せようと思います。

4.31 セキュリティー

Gnus は署名されたメッセージを検証したり、暗号化されたメッセージをデコードすることができます。PGP, PGP/MIME およびS/MIME の形式をサポートしますが、それらを動作させるためには、いくつかの外部プログラムを必要とします:

1. PGP とPGP/MIME のメッセージを扱うには、OpenPGP の実装である GnuPG のようなものをインストールしなければなりません。Emacs に含まれている GnuPG への

インターフェースは EasyPG というもの(see Section “EasyPG” in *EasyPG Assistant user's manual*) ですが、Mailcrypt もサポートします。

2. S/MIME のメッセージを扱うには、OpenSSL をインストールする必要があります。OpenSSL 0.9.6 か、それより新しいものがお勧めです。

以下は、メッセージを読む、または作成する場合に、セキュリティーの機能を制御するための変数です:

`mm-verify-option`
: 署名されたパートを検証するためのオプション。`never` は検証しない、`always` はいつも検証する、`known` は知られたプロトコルの場合だけ検証する、の意味です。それら以外の場合は、どうするかを利用者に尋ねます。

`mm-decrypt-option`
: 暗号化されたパートをデコードするためのオプション。`never` はデコードしない、`always` はいつもデコードする、`known` は知られたプロトコルの場合だけデコードする、の意味です。それら以外の場合は、どうするかを利用者に尋ねます。

`mm-sign-option`
: 署名されたパートを作成するためのオプション。`nil` ではディフォルトの署名のための鍵を使い、`guided` では署名のための鍵をメニューから選びます。

`mm-encrypt-option`
: 暗号化されたパートを作成するためのオプション。`nil` では‘`From:`’ ヘッダーを受取人として最初に合致する公開鍵を使い、`guided` では受取人のキーをメニューから選びます。

`mml1991-use`
: PGP のメッセージのための、OpenPGP の実装への elisp インターフェースを示すシンボルです。ディフォルトは`epg` ですが、推奨はしないものの`mailcrypt` もサポートします。ディフォルトでは、Gnus はこの順番で最初に見つかるインターフェースを使います。

`mml2015-use`
: PGP/MIME のメッセージのための、OpenPGP の実装への elisp インターフェースを示すシンボルです。ディフォルトは`epg` ですが、推奨はしないものの`mailcrypt` もサポートします。ディフォルトでは、Gnus はこの順番で最初に見つかるインターフェースを使います。

ディフォルトではセキュリティーの情報を表示するボタンが現れません。それらは実際にメールを読む際に邪魔になるからです。`K b` をタイプすれば、その情報を表示することができますが。これを恒久的に行なわせるには、`gnus-buttonized-mime-types` および`gnus-unbuttonized-mime-types` 変数を使ってください。これらの変数の詳細と、常にセキュリティーの情報を表示させるためにカスタマイズする方法は、Section 4.19 [MIME Commands], p. 103, を参照してください。

メニュー項目やコマンドから OpenPGP の鍵を取得(snarf) する(すなわち、記事から鍵を鍵束に輸入(import) する) 機能は、明示的にはサポートされません。というよりはむしろ、あなたが適切だと思うどんな動作をも通常のMIME の機構を介して指定できるように、Gnus は‘`application/pgp-keys`’ として鍵を検出し、ラベルを付けます。MIME ボタンをクリック(see Section 5.2 [Using MIME], p. 126) したときに、GNU Privacy Guard を使っ

て鍵を輸入してくれるようにするには、以下のような行を`~/.mailcap` ファイル(see Section "mailcap" in *The Emacs MIME Manual*) に記入してください。

```
application/pgp-keys; gpg --import --interactive --verbose; needsterminal
```

これは、たまたま`mailcap-mime-data` ですでに定義されている、ディフォルトの動作でもあります。

送信するメッセージに署名したり暗号化するために、どうやって設定するかについてのもっと詳しい情報が、message マニュアル(see Section "セキュリティー" in *The Message Manual*) で見つかるでしょう。

4.32 メーリングリスト

Gnus は RFC 2369 で規定された各種のメーリングリストで使われるフィールドを理解します。これを有効にするには概略バッファーで*A M* (`gnus-mailing-list-insinuate`) を使うなどして、`to-list` グループパラメーター(see Section 3.10 [Group Parameters], p. 23) を追加してください。

これによって概略バッファーでの以下の命令が使えるようになります。

C-c C-n h List-Help フィールドがあったら、メーリングリストのヘルプを取り寄せるためのメッセージを送信します。

C-c C-n s List-Subscribe フィールドがあったら、メーリングリストの購読を始めるためのメッセージを送信します。

C-c C-n u List-Unsubscribe フィールドがあったら、メーリングリストの購読をやめるためのメッセージを送信します。

C-c C-n p List-Post フィールドがあったら、メーリングリストに投稿します。

C-c C-n o List-Owner フィールドがあったら、メーリングリストの管理者宛てにメッセージを送信します。

C-c C-n a List-Archive フィールドがあったら、メーリングリストのアーカイブを閲覧します。

5 記事バッファー

記事は一つしかない記事バッファーに表示されます。すべての概略バッファーは(Gnus に指示しない限り) 同じ記事バッファーを共有します。

5.1 余分なヘッダーを隠す

各記事の頭の部分はヘッダー(*head*) と呼ばれます。(残りの部分はボディー(*body*) です。すでにお気づきでしょうが。)

ヘッダーにはたくさんの有益な情報が含まれています。記事を書いた人の名前、それが書かれた日付、および記事の表題です。これはとても良いのですが、ヘッダーには大部分の人にとっては見たくもない情報---記事があなたのところに着くまでにどんなシステムを経由してきたか、`Message-ID`、`References` などなど...もううんざりするくらい---たくさん含まれています。たぶんあなたはこれらの行のいくつかは取り除いてしまいたいと思うでしょう。もしこれらの行をすべて記事バッファー内に残しておきたければ、`gnus-show-all-headers` を`t` に設定してください。

Gnus はヘッダーを選り分けるために二つの変数を用意しています:

`gnus-visible-headers`
: この変数が`nil` 以外であれば、どのヘッダーを記事バッファーに残したいかを指定する正規表現であるとみなされます。この変数に合致しないヘッダーはすべて隠されます。

  例えば、記事を書いた人の名前と表題のみを見たければ、こう指定します:

  ```
  (setq gnus-visible-headers "^From:\\|^Subject:")
  ```

  この変数は、表示させたいヘッダーに合致する正規表現をリストで指定することもできます。

`gnus-ignored-headers`
: この変数は`gnus-visible-headers` の反対です。この変数が設定されていれば(かつ`gnus-visible-headers` が`nil` であれば)、これは隠したいヘッダー行すべてに合致する正規表現であるとみなされます。この変数に合致しないすべてのヘッダー行が表示されます。

  例えば、単に`References` 欄と`Xref` 欄のみを消し去りたければ、以下のようにします:

  ```
  (setq gnus-ignored-headers "^References:\\|^Xref:")
  ```

  この変数は消したいヘッダーに合致する正規表現のリストでも構いません。

  なお、`gnus-visible-headers` が`nil` 以外の場合は、この変数には効果が無いことに注意してください。

Gnus はヘッダーの並べ替え(sort) も行ないます(これはディフォルトで行なわれます)。この並べ替えは`gnus-sorted-header-list` 変数を設定することで制御することができます。これはヘッダーをどういう順序で表示するかを指定する正規表現のリストです。

例えば、記事の著者名を最初に、次に表題を表示したければ、こんな風になるでしょう。

```
(setq gnus-sorted-header-list '("^From:" "^Subject:"))
```

表示するようになっているヘッダーでこの変数に指定されていないものは、この変数に指定されているすべてのヘッダーの後に、適当な順序で表示されるでしょう。

gnus-treat-hide-boring-headers をhead に設定することによって、もっとつまらないヘッダーを隠すことができます。この関数が何をするかはgnus-boring-article-headers 変数に依存します。この変数はリストですが、このリストには実際のヘッダーの名前が入るのではありません。代わりに Gnus がチェックして視界から消し去るためのさまざまな「つまらない条件」(*boring conditions*) のリストを指定します。

この条件には以下のようなものがあります。

`empty` 空のヘッダーをすべて消去します。

`followup-to`
Followup-To 欄がNewsgroups 欄と同一である場合には消去します。

`reply-to` Reply-To 欄がFrom 欄と同じアドレスを示しているか、broken-reply-to グループパラメーターが設定されている場合には消去します。

`newsgroups`
Newsgroups 欄が現在のグループ名しか含んでいない場合には消去します。

`to-address`
To 欄が現在のグループのto-address パラメーターと同じものしか含んでいない場合には消去します。

`to-list` To 欄が現在のグループのto-list パラメーターと同じものしか含んでいない場合には消去します。

`cc-list` Cc 欄が現在のグループのto-list パラメーターと同じものしか含んでいない場合には消去します。

`date` その記事が過去三日以内のものであれば、Date 欄を消去します。

`long-to` To 欄および/またはCc 欄があまりにも長い場合には消去します。

`many-to` To 欄および/またはCc 欄が一つよりも多ければ、それらをすべて消去します。

これらのうちの三つの要素を入れたければ、こんな風になります:

```
(setq gnus-boring-article-headers
      '(empty followup-to reply-to))
```

これはこの変数のディフォルト値でもあります。

5.2 MIME を使う

パントマイム(mime) は、観客があくびをしながらぼんやりしているのにもかかわらず、意味も無く空中で手を振るものの標準として広く知られています。

一方MIME は、そのためにすべてのニュースリーダーが恐怖で死んでしまうのにもかかわらず、意味も無く記事をエンコードする標準です。

MIME は記事がどんな文字セットを使うか、文字をどうエンコードするかを指定することができ、さらには絵やその他のみだらなものを無邪気な格好の記事に埋め込むことさえ可能にします。

Gnus はMIME パートを表示するために、gnus-display-mime-function によってMIME 記事を処理します。これはディフォルトではgnus-display-mime で、MIME オブジェクトを表示し、セーブし、かつ操作するために使うことができる、ひとかたまりのクリック可能なボタンを作成します。

MIME ボタンの上にポイントを置いたならば、以下のコマンドが利用できます:

RET (記事)
BUTTON-2 (記事)
: MIME オブジェクトの表示をトグルで切り替えます(gnus-article-press-button)。そのオブジェクトを内蔵のビューワーで表示できないときは、Gnusはmailcap ファイルにある外部のビューワーに助けを求めます。ビューワーが‘copiousoutput’ 仕様になっている場合は、オブジェクトはインラインで(訳注: Emacs の表示に埋め込まれて) 表示されます。

M-RET (記事)
v (記事)
: 手段を尋ね、その手段を使ってMIME オブジェクトを表示します(gnus-mime-view-part)。

t (記事)
: MIME オブジェクトを、異なるMIME メディア・タイプであるかのように表示します(gnus-mime-view-part-as-type)。

C (記事)
: 文字セットを尋ね、その文字セットを使ってMIME オブジェクトを表示します(gnus-mime-view-part-as-charset)。

o (記事)
: ファイル名を尋ねてMIME オブジェクトをセーブします(gnus-mime-save-part)。

C-o (記事)
: ファイル名を尋ね、MIME オブジェクトをセーブして、それを記事から取り外します(記事を編集することによって行なわれます)。取り外されたMIME オブジェクトは message/external-body MIME タイプとして参照されるようになります(gnus-mime-save-part-and-strip)。

r (記事)
: ファイル名の入力を求めて、MIME オブジェクトを message/external-body 型のMIME 形式のファイルとして参照される外部にある本体で置き換えます。(gnus-mime-replace-part)。

d (記事)
: 記事からMIME オブジェクトを取り外し、取り外したことを表す告知で置き換えます(gnus-mime-delete-part)。

c (記事)
: MIME オブジェクトを新たに作ったバッファーにコピーして、それを表示します(gnus-mime-copy-part)。接頭引数が与えられると、デコードせずに生の内容物をコピーします。数値の接頭引数を与えると、文字セットによるデコードを半手動で切り替えることができます(Section 4.4 [Paging the Article], p. 56, で述べられているgnus-summary-show-article-charset-alist を参照してください)。auto-compression-mode (see Section “Accessing Compressed Files” in *The Emacs Editor*) が設定されていると、.gz や.bz2 のような圧縮されたファイルを自動的に解凍します。

p (記事)
: MIME オブジェクトを印刷します(gnus-mime-print-part)。このコマンドは.mailcap ファイルで定義された‘print=’ 仕様に従います。

i (記事)
: MIME オブジェクトの内容物を、その記事バッファーに‘text/plain’ として挿入します(gnus-mime-inline-part)。接頭引数が与えられると、デコードせずに生の内容物を挿入します。数値の接頭引数を与えると、文字セットによるデコードを半手動で切り替えることができます(Section 4.4 [Paging the Article], p. 56, で述べられているgnus-summary-show-article-charset-alist を参照

してください)。auto-compression-mode (see Section “Accessing Compressed Files” in *The Emacs Editor*) の設定とは無関係に、.gz や.bz2 のような圧縮されたファイルをjka-compr を使って自動的に解凍します。

E (記事) 内部ビューワーでMIME オブジェクトを表示します。内部ビューワーが使えないときは、外部ビューワーを使います(gnus-mime-view-part-internally)。

e (記事) 外部ビューワーでMIME オブジェクトを表示します(gnus-mime-view-part-externally)。

| (記事) MIME オブジェクトをプロセスに出力します(gnus-mime-pipe-part)。

. (記事) MIME オブジェクトをどう処理するかを、対話的に決めて実行します(gnus-mime-action-on-part)。

Gnus はいくつかの種類のMIME オブジェクトを自動的に表示します。どのパートに対してそうするかを Gnus が決めるやり方については、Emacs MIME マニュアルで述べられています。

不愉快なものでびっくりさせられるのを避けるには、トグルで切り替える関数を使うのが最も良いでしょう。(例えば、‘alt.sing-a-long’ グループに入ると、あなたの気づかないうちにMIME は記事中のサウンドファイルをデコードして、何やら怪しげな長い長い歌があなたのスピーカーから大音響で流れ出し、あなたはボリュームボタンを見つけられず、というのはそんなものはもともと付いていないからで、みんなはあなたの方を睨みはじめ、あなたはプログラムを止めようとするけれどもできなくて、ボリュームを制御するプログラムも見つけられなくて、そして部屋中の全員は突然あなたのことを軽蔑の眼差しで見るようになってしまい、あなたはちょっと面白くない思いをする、とか)。

現実の出来事と実在の人物に類似しているかもしれませんが、これはすべてホントのことです。げほげほ。

Section 4.19 [MIME Commands], p. 103, も見てください。

5.3 HTML

Gnus はHTML 記事を素敵に整形して記事バッファーに表示することができます。たくさんの方法がありますが、それらのうちの 2 つがディフォルトの手段であると言えます。

もし Emacs が libxml2 を使うように作られていれば、Gnus は純粋に ELisp で書かれていて Emacs に組み込まれている Simple HTML Renderer shr を使います。[1] shr は Emacs のブラウザーである EWW (see Section “EWW” in *The Emacs Manual*) でも使われます。

Emacs に libxml2 のサポートが無くてもw3m がインストールされていれば、Gnus はHTML メールを描画して、その結果を記事バッファーに表示します(gnus-w3m)。

完全な概要についてはSee Section “Display Customization” in *The Emacs MIME Manual*, を参照してください。この章はディフォルトの機能だけを説明します。

mm-text-html-renderer
もしこれがgnus-article-html に設定されていると、Gnus はw3m を利用するその組み込み機能を使います。

[1] shr はHTML 記事が指定する色を表示しますが、それらを読むことができるように調整しようとします。もっとコントラストを強くしたいならsee [FAQ 4-16], p. 398, を見てください。

`gnus-blocked-images`
: URL がこの正規表現に合致する外部画像は取得も表示もされません。例えば“ads” という文字列をそれらの内に持つすべてのURL を阻止するには、以下のようにしてください:

```
(setq gnus-blocked-images "ads")
```

これは評価されるべき関数でも構いません。その場合、それはグループ名をパラメーターとして呼ばれます。ディフォルト値は`gnus-block-private-groups` で、ニュースグループの記事ではないどんなものに対しても‘`"."`’ を返します。これはメールを読むときに外部画像を取得しないということで、それによってあなたがメールを読んだかどうかを追跡するために誰も web bugs (訳注: インターネットユーザを追跡するためのツール) の類を使えないようにします。

特定のプライベートグループをパブリックグループであるかのように扱いたい場合は、そのグループの名前を`gnus-global-groups` リストに追加してください。

`gnus-inhibit-images` も参照してください(see Section 5.6 [Misc Article], p. 133)。

`gnus-html-frame-width`
: HTML を描画するときの幅です。ディフォルトは 70 です。

`gnus-max-image-proportion`
: 表示される画像が、それらがあるウィンドウに対してどれくらい大きいかを表す値です。0.7 という値(ディフォルト) は、それらがウィンドウの幅と高さの 70% を取ることができることを意味します。画像がこれより大きくて、かつ Emacs がそれを行なうことができるならば(訳注: ImageMagick ライブラリーを使うように構築されているならば)、これらの基準に合うように縮小されます。

`gnus-article-show-cursor`
: これが`nil` 以外の値だったら、記事バッファーが現在バッファーではないときでも、記事バッファーにカーソルを表示します。

これを使うためには`w3m` と`curl` がインストールされている必要があります。もしあればGnus は自動的にHTML を表示するはずです。

5.4 記事のカスタマイズ

記事をどのように見せるかをカスタマイズするためのたくさんの関数が存在しています。これらの関数を対話的に呼ぶこともできるし(see Section 4.18.4 [Article Washing], p. 94)、記事を選択したときに自動的に選択することもできます。

自動的に呼ばれるようにするためには、対応するトリートメント変数を設定しなければなりません。例えばヘッダーを隠すためには、`gnus-treat-hide-headers` を設定します。以下は設定できる変数の一覧ですが、まずこれらの変数の取り得る値について話しましょう。

注意: いくつかの値は、有効な値であってもほとんど意味を無しません。実用的な値は下の一覧を調べてください。

1. `nil`: このトリートメントをしません。
2. `t`: このトリートメントをすべての本文のパートで行ないます。
3. `head`: ヘッダーでそのトリートメントをします。

4. `first`: このトリートメントを最初の本文のパートで行ないます。
5. `last`: このトリートメントを最後の本文のパートで行ないます。
6. 整数: このトリートメントをこの数値より短いすべての本文のパートで行ないます。
7. 文字列のリスト: このリストに含まれている正規表現に合致する名前のグループで読まれた記事の、すべての本文のパートでこのトリートメントを行ないます。
8. 最初の要素が文字列でないリストです:
   リストは再帰的に評価されます。リストの最初の要素は述語です。以下の述語が認識されます: `or`, `and`, `not`, `typep`。例です:

```
(or last
    (typep "text/x-vcard"))
```

9. 関数: 引数を受け取らない関数で、`nil` または non-`nil` を返します。現在の記事は変数`gnus-article-buffer` の値が示すバッファーにあります。

ここで「パート」という語が使われていることに気付いたと思います。これはメッセージにはMIME マルチパート記事があり、いくつかのパートに分割されているかもしれないという事実に関連しています。マルチパートでない記事は一つのパートのみであるとみなされます。

このトリートメントはすべてのマルチパートのパートたちに適用されるのでしょうか? はい、そうしたければそうなります。ですが、ディフォルトでは‘`text/plain`’ パートだけにトリートメントが施されます。これは`gnus-article-treat-types` 変数で制御され、これはパートの型に合致する正規表現のリストです。制御変数の値が、上で説明されているように述語のリストであるときは、この変数は無視されます。

以下のトリートメントのオプションが使用可能です。これをカスタマイズするための最も簡単な方法は`gnus-article-treat` カスタマイズグループを調査することです。丸括弧の中の値は提案されている意味のある値です。他のものも可能ですが、ほとんどの人にとってはおそらくここに一覧表示されているもので十分でしょう。

`gnus-treat-buttonize (t, integer)`
`gnus-treat-buttonize-head (head)`
See Section 4.18.6 [Article Buttons], p. 98.

`gnus-treat-capitalize-sentences (t, integer)`
`gnus-treat-overstrike (t, integer)`
`gnus-treat-strip-cr (t, integer)`
`gnus-treat-emojize-symbols (t, integer)`
`gnus-treat-strip-headers-in-body (t, integer)`
`gnus-treat-strip-leading-blank-lines (t, first, integer)`
`gnus-treat-strip-multiple-blank-lines (t, integer)`
`gnus-treat-strip-pem (t, last, integer)`
`gnus-treat-strip-trailing-blank-lines (t, last, integer)`
`gnus-treat-unsplit-urls (t, integer)`
`gnus-treat-wash-html (t, integer)`
See Section 4.18.4 [Article Washing], p. 94.

`gnus-treat-date (head)`
日付ヘッダーを`gnus-article-date-headers` 変数に応じて変形/追加します。これは表示する Date ヘッダーのリストです。利用可能な形式は:

`ut` 協定世界時。GMT とか ZULU とも言います。

`local` ユーザーのローカル時間帯。

`english` 英文として読める形式。

`lapsed` メッセージが投稿されてからの経過時間。

`combined-lapsed`
元の Date ヘッダーと(短い形式の) 経過時間。

`combined-local-lapsed`
ユーザーのローカル時間帯と(短い形式の) 経過時間。

`original` 元の Date ヘッダー。

`iso8601` ISO8601 形式。つまり“2010-11-23T22:05:21” のようなもの。

`user-defined`
`gnus-article-time-format` 変数に従った形式。

See Section 4.18.8 [Article Date], p. 100.

`gnus-treat-from-picon (head)`
`gnus-treat-mail-picon (head)`
`gnus-treat-newsgroups-picon (head)`
See Section 10.16.4 [Picons], p. 289.

`gnus-treat-from-gravatar (head)`
`gnus-treat-mail-gravatar (head)`
See Section 10.16.5 [Gravatars], p. 290.

`gnus-treat-display-smileys (t, integer)`
`gnus-treat-body-boundary (head)`
ヘッダーと本文の間に境界線を追加します。境界線には`gnus-body-boundary-delimiter` に設定された文字列が使われます。

See Section 10.16.3 [Smileys], p. 289.

`gnus-treat-display-x-face (head)`
See Section 10.16.1 [X-Face], p. 287.

`gnus-treat-display-face (head)`
See Section 10.16.2 [Face], p. 288.

`gnus-treat-emphasize (t, head, integer)`
`gnus-treat-fill-article (t, integer)`
`gnus-treat-fill-long-lines (t, integer)`
`gnus-treat-hide-boring-headers (head)`
`gnus-treat-hide-citation (t, integer)`
`gnus-treat-hide-citation-maybe (t, integer)`
`gnus-treat-hide-headers (head)`
`gnus-treat-hide-signature (t, last)`
`gnus-treat-strip-banner (t, last)`
`gnus-treat-strip-list-identifiers (head)`
See Section 4.18.3 [Article Hiding], p. 92.

`gnus-treat-highlight-citation (t, integer)`
`gnus-treat-highlight-headers (head)`
`gnus-treat-highlight-signature (t, last, integer)`
See Section 4.18.1 [Article Highlighting], p. 90.

`gnus-treat-ansi-sequences (t)`
`gnus-treat-x-pgp-sig (head)`
`gnus-treat-unfold-headers (head)`
`gnus-treat-fold-headers (head)`
`gnus-treat-fold-newsgroups (head)`
`gnus-treat-leading-whitespace (head)`
See Section 4.18.5 [Article Header], p. 98.

もちろん、`gnus-part-display-hook` から呼ばれる自分用の関数を書くこともできます。関数はそのパートに範囲が狭められた状態で呼ばれ、ほとんどなんでも好きなことができます。バッファーに保存しておかなければならない情報はありません---何でも変えることができます。

5.5 記事のキーマップ

概略バッファーにおけるキー操作のほとんどは記事バッファーでも使用できます。これらは概略バッファーでそれらを押したかのように動作するはずです。つまり記事を読んでいる間、実際に概略バッファーを表示させておく必要がありません。すべての操作は記事バッファーから行なうことができるのです。

v キーはユーザー用に予約されています。そのまま何かのコマンドに割り当てても構いませんが、接頭キーとして使う方が良いでしょう。

他にもいくつかのキーが利用できます:

`SPC` 記事を一ページ先にスクロールします。(`gnus-article-next-page`)。*h* `SPC` *h* とまったく同じです。

`DEL` 記事を一ページ前にスクロールします(`gnus-article-prev-page`)。*h* `DEL` *h* とまったく同じです。

C-c ^ カーソルが`Message-ID` の近辺にあるときに*C-c ^* を押すと、Gnus はサーバーからその記事を取ってこようとします(`gnus-article-refer-article`)。

C-c C-m カーソルの近くにあるアドレスに返信を送ります(`gnus-article-mail`)。接頭引数を与えると、そのメールを引用します。

s バッファーを再配置して、概略バッファーが見えるようにします(`gnus-article-show-summary`)。

? 利用できるキー操作のごく簡単な説明を出します(`gnus-article-describe-briefly`)。

`TAB` 次のボタンがあればそこに移動します(`gnus-article-next-button`)。これは記事にボタンを付ける機能をオンにしているときのみ意味を持ちます。

M-TAB 一つ前のボタンがあればそこに移動します(`gnus-article-prev-button`)。

R 現在の記事に元記事を含んだ返答のメールを送ります(`gnus-article-reply-with-original`)。もし領域が活性化されていたならば、その領域にあるテキストだけを yank します。

S W 現在の記事に元記事を含んだ広い返答のメールを送ります(gnus-article-wide-reply-with-original)。もし領域が活性化されていたならば、その領域にあるテキストだけを yank します。

訳注:「広い返答」とはヘッダーのTo, From, (もしくはReply-To) とCc) のすべての人に返答をすることです。Mail-Followup-To があれば、代わりにそれが使われます。

F 現在の記事に元記事を含んでフォローアップをします(gnus-article-followup-with-original)。もし領域が活性化されていたならば、その領域にあるテキストだけを yank します。

5.6 記事のその他

gnus-single-article-buffer
: nil 以外であれば、すべてのグループに対して同じ記事バッファーを使用します(これはディフォルトです)。nil であれば、各グループ毎の固有の記事バッファーを持つようになります。

gnus-widen-article-window
: もしnil でなかったら*h* コマンドで記事を選ぶときに記事ウィンドウをフレーム全体を占めるように拡大します。

gnus-article-decode-hook
: MIME 記事をデコードするときに使用されるフックです。ディフォルト値は(article-decode-charset article-decode-encoded-words) です。

gnus-article-prepare-hook
: このフックは記事が記事バッファーに挿入された直後に呼び出されます。これは主に、何か記事の内容に依存する処理をする関数のために用意されています。記事バッファーの内容を変更するような目的で使うべきではないでしょう。

gnus-article-mode-hook
: 記事モードのバッファーで呼び出されるフックです。

gnus-article-mode-syntax-table
: 記事バッファーで用いられる構文テーブル(syntax table) です。これはtext-mode-syntax-table をもとに初期化されます。

gnus-article-over-scroll
: 非-nil にすることによって、それ以上スクロールする新しいテキストが無くても記事バッファーをスクロールできるようにします。ディフォルトはnil です。(訳注: 記事の最下行が見えているときに、nil だとRET キーでそれ以上スクロールしませんが、非-nil にすると記事が見えなくなるまでスクロールします。)

gnus-article-mode-line-format
: この変数はgnus-summary-mode-line-format と同じ仕様に沿った様式文字列です(see Section 4.1.3 [Summary Buffer Mode Line], p. 52)。これは、その変数と同じ様式指定および二つの拡張を受付けます。

  ‘w’ 記事の「洗濯状態」(*wash status*)。これは記事に対して行なわれたであろう洗濯操作を、それぞれ一文字で示す短い文字列になります。文字とそれらの意味は次の通りです:

‘c’ 記事バッファーにおいて、引用された文が隠されているかもしれない場合に表示されます。

‘h’ 記事バッファーにおいて、ヘッダーが隠されている場合に表示されます。

‘p’ 記事が電子署名または暗号化されていて、Gnus がセキュリティーのためのヘッダーを隠していると表示されます。(注: 署名が正しいか間違っているかを表すものではありません。)

‘s’ 記事バッファーにおいて、署名が隠されている場合に表示されます。

‘o’ 記事バッファーにおいて、Gnus が重ね打ち文字のトリートメントを行なった場合に表示されます。

‘e’ 記事バッファーにおいて、Gnus が強調された文字のトリートメントを行なった場合に表示されます。

‘m’ 記事のMIME パートの数です。

`gnus-break-pages`
改ページ(*page breaking*) を行なうかどうかを制御します。この変数が`nil` 以外であれば、記事中にページ区切り文字が現れるごとにページ分割をします。この変数が`nil` であればページ分けは行なわれません。

`gnus-page-delimiter`
これが上で触れた区切り文字です。ディフォルトでは‘^L’ (フォームフィード) です。

`gnus-use-idna`
この変数は‘`From:`’、‘`To:`’ および‘`Cc:`’ ヘッダーにある国際化ドメイン名を、Gnus が IDNA デコードするかどうかを制御します。そのようなメッセージの作り方についてはSee Section “国際化ドメイン名” in *The Message Manual*, を参照してください。これにはGNU Libidn (`https://www.gnu.org/software/libidn/`) が必要で、この変数はそれをインストールしてある場合だけ有効になります。

`gnus-inhibit-images`
もし`nil` でなければ、記事のボディーでの画像のインライン表示を禁止します。これはMIME パートとして記事にある画像、および`mm-text-html-renderer` (see Section “表示のカスタマイズ” in *The Emacs MIME Manual*) が`shr` または`gnus-w3m` である場合に表示されたHTML 記事の画像に対して有効です。

6 メッセージの作成

すべての投稿とメールを送るためのコマンドは、あなたをメッセージバッファーに導きます。そこでは`C-c C-c` を押すことによって記事を送信する前に、記事を好きなように編集することができます。See Section “概要” in *The Message Manual*. メッセージはあなたの設定に基づいて投稿またはメールとして送信されます(see Section 6.2 [Posting Server], p. 135)。

投稿するべきでなかった記事を削除するための情報についてSection 4.5.4 [Canceling and Superseding], p. 60, も参照してください。

6.1 メール

出て行くメールをカスタマイズする変数です:

`gnus-uu-digest-headers`
: 要約メッセージ(digested message) に含まれるヘッダーに合致する正規表現のリストです。ヘッダーは合致した順に取り込まれます。`nil` だったら、すべてのヘッダーを含みます。

`gnus-add-to-list`
: `nil` でなければ、`a` を押したときに、`to-list` グループパラメーターをそれの無いメールグループに付け加えます。

`gnus-confirm-mail-reply-to-news`
: 非-`nil` だったら、あなたがニュース記事への返答をメールでしようとするとGnus は確認を求めます。`nil` ならば、あなたがやりたいことに何も口出ししません。これは関数か正規表現であることもできます。関数は唯一のパラメーターとしてグループ名を受け取り、確認する必要がある場合に非-`nil` を返します。これを正規表現にすると、それに合致する名前のグループで確認を求めます。

  メールで返信する気は無いのに時たまぞんざいに R を押してしまう癖があるならば、この変数はそんなあなたのためにあります。

`gnus-confirm-treat-mail-like-news`
: 非-`nil` だったら、Gnus はメールに返信する時にも`gnus-confirm-mail-reply-to-news` に基づいた確認を求めます。これはメーリングリストをニュースグループのように扱うのに便利です。

6.2 投稿するサーバー

最新の(もちろん、非常に知的な) 記事を送り出すために、あの魔法のような`C-c C-c` キーを押した時、それはどこにいくのでしょう?

尋ねてくれてありがとう。あなたを恨みます。

それは非常に複雑になり得ます。

ニュースを投稿するとき、通常 Message は`message-send-news` を呼び出します(see Section “ニュース変数” in *The Message Manual*)。普通は、Gnus は購読用と同じ選択方法を使って投稿します(このことは、あなたがたくさんのグループを異なったサーバーで購読している場合に、たぶん都合が良いのです)。しかし、あなたが購読しているそのサーバーが投稿を許可せず、読むことのみを許可しているのならば、おそらくあなたの(非常に知的

でとんでもなく興味深い) 記事を投稿するために、他のサーバーを使いたいと思うでしょう。そうならばgnus-post-method を他の方法に設定することができます:

```
(setq gnus-post-method '(nnspool ""))
```

さて、この設定をした後でサーバーがあなたの記事を拒否したり、サーバーが落ちていたりしたら、どうしたらよいのでしょう? この変数よりも優先させるために*C-c C-c* 命令にゼロでない数の接頭引数を与えることによって、投稿に“current”(現在の) サーバーを使わせること、すなわちディフォルトの動作(訳注: gnus-post-method のディフォルト値はcurrent) に戻すことができます。

もし、ゼロを接頭引数としてその命令に与えたなら(すなわち、*C-u 0 C-c C-c*)、Gnus は投稿にどの方法を使うかをあなたに尋ねます。

gnus-post-method を選択方法のリストにすることもできます。その場合は、Gnus は常に投稿にどの方法を使うかをあなたに尋ねます。

最後に、あなたがいつでも基本の選択方法を使って投稿したいのならば、この変数をnative にしてください。

メールを送信するときに、Message はmessage-send-mail-function 変数によって指定される関数を呼び出します。Gnus はそれを、あなたのシステムに適合する値に設定しようとします。詳しくは: See Section “メール変数” in *The Message Manual*.

6.3 POP before SMTP

あなたのISP はPOP-before-SMTP 認証を使いますか? この認証手順は、電子メールを送信する前にPOP サーバーと通信することを要求します。それを行なうには~/.gnus.el ファイルに以下の行を入れてください:

```
(add-hook 'message-send-mail-hook 'mail-source-touch-pop)
```

メールを送信する直前に、mail-source-touch-pop 関数はメールを取得せずにmail-sources の値に従ってPOP 認証を行ないます。See Section 7.4.4 [Mail Sources], p. 166.

もしmail-sources に二つ以上のPOP メールサーバーを設定しているならば、それらの一つをPOP-before-SMTP 認証に使われるPOP メールサーバーとしてmail-source-primary-source に設定する必要があるでしょう。それが第一POP メールサーバーならば(すなわち、主にそのサーバーからメールを取得しているならば)、それを以下のように恒久的に設定することができます:

```
(setq mail-source-primary-source
      '(pop :server "pop3.mail.server"
            :password "secret"))
```

さもなければ、POP-before-SMTP 認証を行なうときだけ、それを以下のように動的に束縛してください:

```
(add-hook 'message-send-mail-hook
          (lambda ()
            (let ((mail-source-primary-source
                   '(pop :server "pop3.mail.server"
                         :password "secret")))
              (mail-source-touch-pop))))
```

6.4 メールと投稿

これはメールの送信とニュースの投稿の両方に関連する変数のリストです:

`gnus-mailing-list-groups`
: あなたのニュースサーバーが、本当にメーリングリストの記事をNNTP サーバーに流し込むゲートウェイによって、それらがニュースグループの記事として見えるようにしているのであれば、それらのグループは問題なく読めるでしょう。しかしいくらか面倒なことを克服すること無しに、それらに投稿またはフォローアップすることはできません。一つの解決法は、グループパラメーター(see Section 3.10 [Group Parameters], p. 23) に`to-address` を加えることです。簡単にできるのは、`gnus-mailing-list-groups` を、本当はメーリングリストであるグループに合致する正規表現に設定することです。そうすれば、少なくともメーリングリストへのフォローアップはたいていのときに行なうことができるでしょう。これらのグループに投稿すること(`a`) は、それでも苦痛を引き起こすでしょうけれど。

`gnus-user-agent`
: この変数は、どの情報が User-Agent ヘッダーに陳列されるかを制御します。シンボルのリスト、または文字列です。有効なシンボルは`gnus` (Gnus のバージョン) および`emacs` (Emacs のバージョン) です。Emacs のバージョンには`config` (`system-configuration` の値) か`type` (`system-type` の値) を加えることができます。これを文字列にするときは、正しいフォーマットを使ってください(RFC2616 参照)。

あなたは自分が送るメッセージで、綴りをチェックしたいかもしれません。もしくは手で綴りのチェックをしたくないのであれば、自動綴りチェックを`ispell` パッケージを使うことによって付け加えることができます:

```
(add-hook 'message-send-hook 'ispell-message)
```

`ispell` の辞書をグループに応じて切り替えたいならば、以下のようにすれば良いでしょう。

```
(add-hook 'gnus-select-group-hook
          (lambda ()
            (cond
             ((string-match
               "^de\\." (gnus-group-real-name gnus-newsgroup-name))
              (ispell-change-dictionary "deutsch"))
             (t
              (ispell-change-dictionary "english")))))
```

あなたの必要に応じて変更してください。

`gnus-message-highlight-citation` を`t` に設定すれば、message モードのバッファーでも記事バッファーと同様に、引用された文のレベルの違いに応じたハイライトが行なわれます。

6.5 メッセージの保管

Gnus はあなたが送ったメールとニュースを貯めておくためのいくつかの違った方法を提供します。ディフォルトの方法はメッセージを保存するために「アーカイブ仮想サーバー」を

使うことです。これを完全に禁止したいのであれば、変数gnus-message-archive-groupをnil にしなければなりません。ディフォルトは"sent.%Y-%m" で、これは月が変わる毎に一つのアーカイブを作ります。

グループで読んだ興味のあるメッセージの保存については、*B c* (gnus-summary-copy-article) コマンドを参照してください(see Section 4.26 [Mail Group Commands], p. 114)。

gnus-message-archive-method は、送ったメッセージを保存するためにどの仮想サーバーを Gnus が使うかを指定します。ディフォルトは"arhive" で、実際に使われるときに以下の方法に展開されます:

```
(nnfolder "archive"
          (nnfolder-directory   "~/Mail/archive")
          (nnfolder-active-file "~/Mail/archive/active")
          (nnfolder-get-new-mail nil)
          (nnfolder-inhibit-expiry t))
```

注: このようなサーバーは、"archive" という名前のサーバーの実際の選択方法としてその後ずっと使えるようにするために(つまりgnus-message-archive-method が"archive" に設定された場合のために)、最初に~/.newsrc.eld ファイルに保存されます。もしそれが一度保存されると、gnus-message-archive-method の値を後で変更しても、ディフォルトではそれは更新されません。したがって"archive" というサーバーが、いつでもこのようなnnfolder サーバーを意味するとは限りません。保存される選択方法が常にgnus-message-archive-method の値を反映するようにしたい場合は、gnus-update-message-archive-method 変数をnil 以外の値に設定してください。この変数のディフォルト値はnil です。

nnfolder はこのようなことをするのには極めて適した選択方法なのですが、これに限らずnnml やnnmbox などの、どのメール選択方法でも使うことができます。ディフォルトで選択されるディレクトリーが気にいらなければ、次のようにすることができます:

```
(setq gnus-message-archive-method
      '(nnfolder "archive"
                 (nnfolder-inhibit-expiry t)
                 (nnfolder-active-file "~/News/sent-mail/active")
                 (nnfolder-directory "~/News/sent-mail/")))
```

訳注: 上記のような例は「意図した通りに動作しない」FAQ のネタになりつつあり、不具合の原因が特定できない事例が少なくありません。例えば、同じ"archive" という名前の仮想サーバーを過去に使ったことがあると、それが~/.newsrc.eld ファイルの中でgnus-server-alist 変数に登録されているかもしれません。あるいは単に、同名の仮想サーバーを現在も使っているかもしれません。そのような場合は、別の名前を使う必要があります。

Gnus は外へ出て行くすべてのメッセージに、gnus-message-archive-method で指定されたアーカイブ仮想サーバーにある(あるいはそれ以外のサーバーにある) 一つかそれ以上のグループに保存することを意図したGcc 欄を挿入します。どのグループを使うかは変数gnus-message-archive-group によって決まります。

この変数(gnus-message-archive-group) は次のようなことをするために使うことができます:

文字列　　メッセージはそのグループに保存されます。

グループ名に選択方法を含めることができますが、そうするとそのメッセージはgnus-message-archive-method で指定した選択方法ではなくて、代わりにグループ名の選択方法で保存されることに注意しましょう。gnus-message-archive-method は、上に示したようなディフォルト値を持つためものであると考えてください。ですからgnus-message-archive-group を"foo" にしておけば、外へ出て行くメッセージは‘nnfolder+archive:foo’ に保存されますが、"nnml:foo" という値を使うと、外へ出て行くメッセージは‘nnml:foo’ に保存されるでしょう。

文字列のリスト
: メッセージはそれらのすべてのグループに保存されます。

正規表現、関数、Lisp フォームの連想リスト
: キーが『合致』すると、その結果が使われます。

  訳注: 正確には以下の三種類です。

  - 正規表現とグループ名(または複数のグループ名リスト) の連想リスト。最初に正規表現が合致した要素のグループ名(またはグループ名のリスト) が使われます。
  - 関数のリスト。それぞれの関数には現在のグループ名が引数として与えられ、最初に返ってきたnil 以外の値が使われます。
  - Lisp フォームのリスト。それぞれのフォームが評価され、最初に返ってきたnil 以外の値が使われます。

nil
: メッセージの保存は行なわれません。

例をあげてみましょう:

‘MisK’ という単一のグループに保存するだけならば:

```
(setq gnus-message-archive-group "MisK")
```

二つのグループ、‘MisK’ と‘safe’ に保存するならば:

```
(setq gnus-message-archive-group '("MisK" "safe"))
```

どのグループにいるかによって違ったグループに保存するなら:

```
(setq gnus-message-archive-group
      '(("^alt" "sent-to-alt")
        ("mail" "sent-to-mail")
        (".*" "sent-to-misc")))
```

もっと複雑なもの:

```
(setq gnus-message-archive-group
      '((if (message-news-p)
            "misc-news"
          "misc-mail")))
```

すべてのニュースメッセージを一つのファイルに保存して、メールメッセージを一月につき一つのファイルに保存するというのはどうでしょう:

```
(setq gnus-message-archive-group
      '((if (message-news-p)
            "misc-news"
          (concat "mail." (format-time-string
```

```
                                        "%Y-%m" (current-time))))))
```

さあ、メッセージを送ると適切なグループに保存されるようになりました。(もし特定のメッセージを保存をしたくないのであれば、挿入されたGcc 欄を取り除いてください。) 保管グループは次に Gnus を起動したときか、次にグループバッファーで*F* を押したときにグループバッファーに現れます。他のグループと同じように、そのグループに入って記事を読むことができます。そのグループが本当に大きくて悩ましくなったら、なにか良いものにその名前を変更することができます(グループバッファーで*G r* を使うことによって)—‘misc-mail-september-1995’ その他何でも。新しいメッセージは古い(今は空になった) グループに溜められます。

gnus-gcc-mark-as-read
: もし非-nil なら、Gcc の記事に既読の印を付けます。

gnus-gcc-externalize-attachments
: nil だったら、ファイルを通常のパートとして Gcc で保存する記事のコピーに添付します。それが正規表現で Gcc のグループ名に合致する場合は、外部パートとしてファイルを添付します。all だったらローカルファイルを参照する外部パートとして添付します。それが別の非-nil だった場合の動作はall のときと同じですが、将来は変わるかもしれません。

  (訳注: 送信したメッセージと同じものを Gcc で保存する代わりに、添付ファイルをメッセージから切り離して、別にセーブするかどうかを制御する変数です。)

gnus-gcc-self-resent-messages
: gcc-self グループパラメーターのようなものですが、gnus-summary-resend-message (see Section 4.5.1 [Summary Mail Commands], p. 57) が再送する変更されていないメッセージだけに適用されます。この変数のnil 以外の値は、どんなGcc ヘッダーよりも優先します。

  これがもしnone だったらGcc コピーは行なわれません。もしt だったら、再送するメッセージは現在のグループにGcc コピーされます。もしこれが文字列だったら、それで再送するメッセージがコピーされるグループを指定します。nil だったら、もしあれば既存のGcc ヘッダーに従ってGcc が行なわれます。これがもしno-gcc-self だったら(それがディフォルト)、再送するメッセージは既存のGcc ヘッダーが指定するものから現在のグループを除いたグループにGcc コピーされます。

gnus-gcc-pre-body-encode-hook
gnus-gcc-post-body-encode-hook
: これらの hook は送信したメッセージをGcc コピーする際、そのボディーをエンコードする前後に実行されます。Hook が実行されるときのバッファーには、ヘッダーを含めたそのメッセージがあります。そのメッセージに対して行なわれる変更は、元のメッセージではなくGcc コピーだけに作用します。これらの hook はコピーを編集するため(そして続いて起こる何らかの変換に影響を及ぼすため)、例えば MML のセキュリティー・タグ(see Section 6.9 [Signing and encrypting], p. 144) を取り除くような用途に使うことができます。

6.6 投稿様式

それらはすべて変数で、私の頭をくらくらさせます。

投稿するグループによって違ったOrganization と署名を付けたいんですか? そして、家のマシンと職場のマシンの両方から投稿するけれども、違ったFrom 行などを使いたいんですか? そんなこと、どうでもいいじゃありませんか。

そのようなことをする方法の一つは、変更する必要のある変数を変更する賢いフックを書くことです。それは少し退屈なので、利用者にこれらのことを手軽な連想リストで指定するというすばらしい着想にたどり着いた人がいました。これが変数gnus-posting-styles の例です:

```
((".*"
   (signature "Peace and happiness")
   (organization "What me?"))
 ("^comp"
  (signature "Death to everybody"))
 ("comp.emacs.i-love-it"
  (organization "Emacs is it")))
```

この例から推測されるように、この連想リストはいくつかの「様式」(style) からなっています。それぞれの様式は最初の要素が何らかの形で「合致」したときに適用されます。連想リスト全体は最初から最後まで反復して実行され、それぞれの合致が適用されます。これは、後の様式の属性が前に合致した様式の属性よりも優先されるということです。ですから‘comp.programming.literate’ は、‘Death to everybody’ という署名と‘What me?’ というOrganization ヘッダーを持ちます。

それぞれの様式の最初の要素は**マッチ** (match) と言います。もしそれが文字列であれば、Gnus はそれを正規表現であるものとして、グループ名に合致するかどうかを調べます。(header **合致** **正規表現**) という形式であれば、Gnus は元記事の中からその名前が**合致** であるヘッダーを探し、それを**正規表現** と比較します。**合致** と**正規表現** は文字列です。(元記事とは、あなたがそれに対して返信またはフォローアップしようとしている対象の記事です。返信あるいはフォローアップを作成していなければ、合致するものは何もありません。) もし**マッチ** が関数のシンボルであれば、その関数が引数無しで呼ばれます。それが変数のシンボルであれば、その変数が参照されます。それがリストであれば、そのリストが**評価** されます。どの場合でも、これがnil でない値を返せば、様式は**合致した** と言います。

それぞれの様式は任意の量の「属性」を持つことができます。それぞれの属性は(*name* *value*) の対により成り立っています。加えて(*name* :file *value*) の形式か(*name* :value *value*) の形式を使うこともできます。ここで:file は*value* がファイル名を表して、その内容が属性値として使用されるべきであることを示し、:value は*value* がファイル名を表わさないことを明示的に示します。属性名(*name*) は、以下のどれかであることができます。

- signature
- signature-file
- x-face-file
- address (user-mail-address よりも優先されます)
- name ((user-full-name) よりも優先されます)
- body

signature-file 属性はmessage-signature-directory 変数を見ることに注意してください。

属性名は文字列またはシンボルであることもできます。その場合それはヘッダー名として使われ、その値が記事のヘッダーに挿入されます。もし属性名がnil だったら、そのヘッ

ダー名は削除されます。もし属性名が`eval` だったらその様式が評価され、結果は捨てられます。

属性値は文字列、引数の無い関数(返り値が使われます)、変数(その値が使われます) またはリスト(それは**評価** されて、返り値が使われます) であることができます。関数と S 式(sexp) はセットアップされつつあるメッセージバッファーで呼ばれるか評価されます。

属性値が文字列である場合、もし`match` が正規表現だったら、あるいはそれが`(header` *`match regexp`*`)` の形式だったら、'`gnus-match-substitute-replacement`' はその属性値上で、位置パラメーター'`\n`' (訳注: 何番目の括弧に合致するかを*n* で指定するもの) を対応する合致した括弧で置き換えようとします(see Section "Replacing the Text that Matched" in *The Emacs Lisp Reference Manual*)。

作成しようとしているメッセージがニュース記事かメールメッセージであるかを調べたいときは、関数`message-news-p` と`message-mail-p` の戻り値を調べてください。

そして、これは例です:

```
(setq gnus-posting-styles
      '((".*"
         (signature-file "~/.signature")
         (name "User Name")
         (x-face-file "~/.xface")
         (x-url (getenv "WWW_HOME"))
         (organization "People's Front Against MWM"))
        ("^rec.humor"
         (signature my-funny-signature-randomizer))
        ((equal (system-name) "gnarly")  ;; 様式
         (signature my-quote-randomizer))
        (message-news-p        ;; 関数シンボル
         (signature my-news-signature))
        (window-system         ;; 変数シンボル
         ("X-Window-System" (format "%s" window-system)))
        ;; Lars さんに返事をするときは
        ;; Organization ヘッダーを付けよう。
        ((header "to" "larsi.*org")
         (Organization "Somewhere, Inc."))
        ;; 元のメッセージが送られてきた宛先と同じだが
        ;; そのサブのアドレスから返事をします。
        ((header "x-original-to" "me\\(\\+.+\\)@example.org")
         (address "me\\1@example.org"))
        ((posting-from-work-p) ;; 利用者が定義した関数
         (signature-file "~/.work-signature")
         (address "user@bar.foo")
         (body "お前はクビだ。\n\n 親愛なるボスより。")
         ("X-Message-SMTP-Method" "smtp smtp.example.org 587")
         (organization "Important Work, Inc"))
        ("nnml:.*"
         (From (with-current-buffer gnus-article-buffer
                 (message-fetch-field "to"))))
        ("^nn.+:"
```

```
(signature-file "~/.mail-signature"))))
```

‘nnml:.*’ の規則は、あなたが出すすべての返事のTo アドレスをFrom アドレスとして使うことを意味します。これは、あなたがたくさんのメーリングリストに参加している場合に便利でしょう。代わりにmessage-alternative-emails を使うこともできます。See Section “メッセージヘッダー” in *The Message Manual*.

「業務」様式の中で特に興味深いのは‘X-Message-SMTP-Method’ ヘッダーです。それはE メールをどうやって送るかを指定します。例えば、会社の規則によっては特定の E メールは特定のSMTP サーバーを通す必要があるでしょう。See Section “メール変数” in *Message Manual*.

6.7 下書き

メッセージ(メールもしくはニュース) を書いているときに、オーブンにステーキが入っている(もしくはあなたがとーってもすごい菜食主義者で、何かのペーストがフードプロセッサーに入っている) ことを突然思い出したなら、書いているメッセージを保存する方法があれば良いと思うでしょう。いつか別の日に編集を続けることができ、それが完成したと思ったときに送ることができるように。

えぇ、心配しないでください。メールかニュースを送信するための Gnus の命令を使って何らかのメッセージを書き始めたときにあなたが手にするバッファーは、自動的に特別な*draft* グループに関連付けられます。普通の方法(例えば*C-x C-s*) でバッファーを保存すれば、その記事はそこに保存されます。(自動保存(auto-save) ファイルも下書きグループ(draft group) に行きます。)

下書きグループは‘nndraft:drafts’ と呼ばれる特別なグループです(あなたが絶対に知っていなければならないのであれば、それはnndraft グループとして実装されています)。変数nndraft-directory はnndraft がそのファイルをどこに保管するかを指定します。このグループを特別なものにしているのは、その中の記事に可視や既読の印を付けることができないことです---そのグループのすべての記事は永久に未読です。

もしグループが存在しないと、それは作成され、購読させられます。グループバッファーからそれを消し去る唯一の方法は、それを購読しないようにすることです。下書きグループの特別の特性はグループの特性(see Section 3.10 [Group Parameters], p. 23) によって生じ、それが失われてしまうと他のグループのように振る舞うようになります。これは(グループの特性を消してしまうことは) 以下のコマンドが使えないことを意味します。そのグループの特別の特性を復活させる最も簡単な方法は、*C-k* でそのグループを削除してからGnus を再起動することです。そのグループの内容物は失われません。

記事の編集を続けたいときは、下書きグループに入って*D e* (gnus-draft-edit-message) を押すだけです。編集を中断したときの状態のバッファーに移動します。

送信を拒否された記事も、この下書きグループに入れられます(see Section 6.8 [Rejected Articles], p. 144)。

送信を拒否されたメッセージがたくさんあって、それ以上編集せずにそれらを送信したい場合は、*D s* 命令(gnus-draft-send-message) を使うことができます。この命令はプロセス/接頭引数の習慣を理解します(see Section 10.1 [Process/Prefix], p. 271)。*D S* 命令(gnus-draft-send-all-messages) はバッファーのすべてのメッセージを送り出します。

送りたくないメッセージがいくつかあるのであれば、*D t* 命令(gnus-draft-toggle-sending) を使ってメッセージに送信不可の印を付けることができます。これは切り替え命令です。

最後に。もし下書きの記事を消したいときは、通常の*B* DEL コマンドを使ってください(see Section 4.26 [Mail Group Commands], p. 114)。

6.8 拒否された記事

時々ニュースサーバーは記事を送信することを拒否します。おそらくサーバーはあなたの顔を好きではないのでしょう。おそらく落ち込んでいるのでしょう。おそらく悪魔*(demon)*がいるのでしょう。おそらく引用文を入れすぎたのでしょう。おそらくディスクが一杯だったのでしょう。おそらくサーバーが落ちていたのでしょう。

もちろんこれらの状況は完全に Gnus の扱える範囲外です。(もちろん Gnus はあなたの風貌を愛しているし、いつも機嫌が良いし、中を飛び回る天使がいて、どれくらい引用文が含まれていようと気にせず、一杯になったり、落っこちたりしません。) ですから Gnus はこれらの記事を後でサーバーの機嫌が良くなるまで保存します。

拒否された記事は自動的に特別な下書きグループ(see Section 6.7 [Drafts], p. 143) に入れられます。サーバーが復旧した暁には、普通あなたはそのグループに入って、すべての記事を送ることになるでしょう。

6.9 署名と暗号化

素のPGP 形式、PGP/MIME またはS/MIME を使って、Gnus はメッセージに電子署名したり暗号化することができます。そのようなメッセージのデコードに関しては、`mm-verify-option` オプションおよび`mm-decrypt-option` オプション(see Section 4.31 [Security], p. 122) を参照してください。

署名したメッセージを送ってきた人たちに、署名した返信を返したいことはしばしばあります。さらに暗号化されたメッセージへの返信を暗号化したいことは、もっとたびたびあるかもしれません。Gnus は前者のために`gnus-message-replysign` の機能を、後者のために`gnus-message-replyencrypt` の機能を提供します。さらに`gnus-message-replysignencrypted` を設定することによって(ディフォルトで on になっています)、暗号化したメッセージに自動的に署名もします。

MIME パートに対してセキュリティーの操作を行なうためのMML への指示は、以下のように署名の場合は*C-c C-m s* キーマップを使って、暗号化の場合は*C-c C-m c* キーマップを使って行ないます。

C-c C-m s s
: S/MIME を使って現在のメッセージに電子署名します。

C-c C-m s o
: PGP を使って現在のメッセージに電子署名します。

C-c C-m s p
: PGP/MIME を使って現在のメッセージに電子署名します。

C-c C-m c s
: S/MIME を使って現在のメッセージを電子暗号化します。

C-c C-m c o
: PGP を使って現在のメッセージを電子暗号化します。

C-c C-m c p
: PGP/MIME を使って現在のメッセージを電子暗号化します。

C-c C-m C-n
: メッセージから、セキュリティー関連のMML タグを外します。

もっと詳しいことはSee Section “セキュリティー” in *The Message Manual*, を参照してください。

7 選択方法

「外部グループ」(foreign group) とは、普通(もしくはディフォルト) の方法で読まれないグループのことです。例えばそれは別のNNTP サーバーのグループであったり、仮想グループであったり、個人的なメールグループであったりするでしょう。

外部グループ(あるいは実際にどんなグループでも) は「名前」と「選択方法」で指定されます。先に後者を例に出すと、選択方法はリストで、最初の要素がどのバックエンドを使うか(例えば`nntp`, `nnspool`, `nnml`) を、二つめの要素が「サーバー名」を表します。選択方法には、その当のバックエンドにとって特別の意味を持つ値である追加の要素があるかもしれません。

選択方法とは「仮想サーバー」を定義することだ、と言うことができます---ですから私たちはまさにそれをしました(see Section 7.1 [Server Buffer], p. 146)。

グループの「名前」は、バックエンドがそのグループを認識する名前です。

例えば‘`some.where.edu`’ というNNTP サーバーにある‘`soc.motss`’ グループは、名前‘`soc.motss`’ と選択方法(`nntp "some.where.edu"`) を持ちます。`nntp` バックエンドはこのグループを‘`soc.motss`’ として知っているだけですが、Gnus はこのグループを‘`nntp+some.where.edu:soc.motss`’ と呼びます。

もちろん、違った方法はすべてそれ特有の要素を持っています。

7.1 サーバーバッファー

伝統的に、「サーバー」は誰かがそれに接続して、それからの情報を要求するマシンかソフトウェアの断片です。Gnus は実際のどんなサーバーにも直接には接続せず、何かのバックエンドを通してすべての処理を行ないます。しかしそれはまさしく実際の媒体と Gnus の間に一つ以上の階層を置くことであって、ちょうどそれぞれのバックエンドが疑似的なサーバーに相当すると言っても良いでしょう。

例えば`nntp` バックエンドは、複数の別々に実在するNNTP サーバー、あるいは実在する同じNNTP サーバーの異なるポートに接続するために用いられます。あなたはどのバックエンドを使うか、そしてどんなパラメーターを設定するかを選択方法(*select method*) に設定して Gnus に指示します。

選択方法の指定は、ときに極めて面倒なものになります---えーと、例えば‘`news.funet.fi`’ というNNTP サーバーのポート 13 を読みたいのだけれど、NOV ヘッダーを取り寄せようとすると固まってしまうし、間違った記事を選択してしまうような場合です。うおっほん。とにかくこのサーバーを使うそれぞれのグループについてそういうことを設定しなければならないとしたら、大変な作業になってしまうでしょう。そこで Gnus は、そういう作業をサーバーバッファーで行なうために、選択方法に名前を付ける手段を設けているのです。

サーバーバッファーに入るためには、グループバッファーで`^` (`gnus-group-enter-server-mode`) コマンドを使ってください。

サーバーバッファーを作成するときに`gnus-server-mode-hook` が実行されます。

7.1.1 サーバーバッファーの表示様式

サーバーバッファーの行の外見を、変数`gnus-server-line-format` 変数を変更することによって変えることができます。これは`format` のような変数で、少しばかり単純な拡張がなされています:

‘`h`’ どのようにニュースが取得されるか---バックエンドの名前。

‘n’ サーバーの名前。

‘w’ どこからニュースが取得されるか---アドレス。

‘s’ サーバーの接続の開いた/閉じた/拒否された状態。

‘a’ そのサーバーがエージェント化されているかどうか。

モード行も変数gnus-server-mode-line-format を使うことによってカスタマイズすることができます(see Section 10.4.2 [Mode Line Formatting], p. 273)。

[訳注: 現在この変数は使われていません。]

以下の仕様が理解されます:

‘S’ サーバー名。

‘M’ サーバーの選択方法。

Section 10.4 [Formatting Variables], p. 272, も参照してください。

7.1.2 サーバー命令

サーバーバッファーでは以下のキー割り当てを使うことができます。いくつかのコマンドは、初期設定ファイルで定義したサーバーではなく、あなたがこのインターフェースを通して(a を使って) 追加したサーバーでだけ働くことを念頭に置いてください。

v *v* キーはユーザー用に予約されています。そのまま何かのコマンドに割り当てても構いませんが、接頭キーとして使う方が良いでしょう。

a 新しいサーバーを追加します(gnus-server-add-server)。

e サーバーを編集します(gnus-server-edit-server)。

S サーバーの定義を表示します(gnus-server-show-server)。

SPC 現在のサーバーを眺めます(gnus-server-read-server)。

訳注: 実際にはgnus-server-read-server-in-server-buffer 命令を呼びますが、gnus-server-browse-in-group-buffer の値がディフォルトのnil であればgnus-server-read-server と同じです。gnus-server-browse-in-group-buffer をnil 以外の値にすることはまったくお勧めできませんが、あなたが何をするのも自由です。詳細はソースコードを読むか、実際に試して痛い目に会ってください。;-p

q グループバッファーに戻ります(gnus-server-exit)。

k 現在のサーバーを切り取ります(kill します) (gnus-server-kill-server)。

y 先ほど切られた(killed) サーバーを貼り付けます(yank します) (gnus-server-yank-server)。

c 現在のサーバーを複写します(gnus-server-copy-server)。

l すべてのサーバーの一覧を表示します(gnus-server-list-servers)。

s サーバーにそのソースから新しい記事を調べるように要求します(gnus-server-scan-server)。主にメールサーバーが意味のある動作をします。

g サーバーにすべてのデータ構造を再作成させます(gnus-server-regenerate-server)。これは同期が外れてしまったメールバックエンドがあるときに役に立ちます。

z

現在位置のサーバーのすべてのグループを圧縮します。今のところnnml (see Section 7.4.13.3 [Mail Spool], p. 188) だけに実装されています。これは記事番号のすきまを取り除くので、正しい全記事数を得ることができるようになります。

サーバーを閉じ、禁止し、および再開するための他のコマンドについてはSection 7.1.7 [Unavailable Servers], p. 150。

7.1.3 方法の例

ほとんどの選択方法は、説明する必要が無いくらいにかなり単純です:

```
(nntp "news.funet.fi")
```

直接スプールから読むのはもっと単純です:

```
(nnspool "")
```

見ての通り、選択方法の最初の要素はバックエンドの名前で、二番目は「アドレス」(address)、もしくはそう呼びたいのであれば「名前」です。

これらの二つの要素の後には、任意の数の(**変数 様式**) の対を置くことができます。

最初の例に戻りましょう---そのマシンのポート 15 から読みたいのだと思ってください。これがその時に、そうなるはずの選択方法です:

```
(nntp "news.funet.fi" (nntp-port-number 15))
```

どの変数が関連するかを見つけ出すために、それぞれのバックエンドの説明文書を読むべきでしょうが、これはnnmh の例です。

nnmh はスプールのような構造を読むためのメールバックエンドです。例えばアクセスしたい二つの構造があるとしましょう: 一つはあなたの私的なメールスプールで、他方は公的なものです。これは私的なメールのために使うことができる指定です:

```
(nnmh "private" (nnmh-directory "~/private/mail/"))
```

(それでこのサーバーは‘private’ と呼ばれますが、あなたはすでに推測していたかもしれませんね。)

これは公的なスプールのための方法です:

```
(nnmh "public"
      (nnmh-directory "/usr/information/spool/")
      (nnmh-get-new-mail nil))
```

あなたが防壁(firewall) の中にいて、防壁マシンを通してNNTP サーバーに接続するしかないのであれば、防壁マシンにrlogin して、そこからnetcat (https://netcat.sourceforge.net/) でNNTP サーバーに接続するように Gnus に指示することができます。こんなことをするのはいささかばかげているのですが、でも仮想サーバーの定義はおそらくこのようなものになるはずです:

```
(nntp "firewall"
      (nntp-open-connection-function nntp-open-via-rlogin-and-netcat)
      (nntp-via-address "the.firewall.machine")
      (nntp-address "the.real.nntp.host"))
```

あの素敵なssh プログラムを、モデムを経由する通信を圧縮するために使いたいのならば、上記の例に以下の設定を加えることができます。

```
      (nntp-via-rlogin-command "ssh")
```

nntp-via-rlogin-command-switches も参照してください。間接的に接続する場合の例です:

```
(setq gnus-select-method
      '(nntp "indirect"
             (nntp-address "news.server.example")
             (nntp-via-user-name "intermediate_user_name")
             (nntp-via-address "intermediate.host.example")
             (nntp-via-rlogin-command "ssh")
             (nntp-via-rlogin-command-switches ("-C"))
             (nntp-open-connection-function nntp-open-via-rlogin-and-netcat)))
```

もちろん、自動認証を行なわせるためにはssh-agent を適切に設定しなければなりません。

防壁の中にいたとしても"runsocks" のようなラッパーコマンドを通して外の世界に直接アクセスできるのならば、以下のように socks 化された netcat でニュースサーバーに接続することができるでしょう:

```
(nntp "outside"
      (nntp-pre-command "runsocks")
      (nntp-open-connection-function nntp-open-netcat-stream)
      (nntp-address "the.news.server"))
```

7.1.4 仮想サーバーを作成する

永続記事を使ってたくさんの記事をキャッシュに保存しているのであれば、キャッシュを読むための仮想サーバーを作る必要があるでしょう。

最初に新しいサーバーを追加する必要があります。それをするのはa 命令です。おそらくキャッシュを読むためにはnnml を使うのが一番良いでしょう。nnspool やnnmh も使えるでしょうけれけど。

a nnml RET *cache* RET とタイプしてください。

今やあなたは真新しい‘cache’ というnnml の仮想サーバーを手に入れたはずです。次はそれを編集して、正しい定義を与えましょう。サーバーを編集するには*e* をタイプしてください。あなたは以下のものを含むバッファーに入ります:

```
(nnml "cache")
```

それを次のように変更してください:

```
(nnml "cache"
      (nnml-directory "~/News/cache/")
      (nnml-active-file "~/News/cache/active"))
```

サーバーバッファーに戻るには*C-c C-c* をタイプしてください。今ではこの仮想サーバーでRET を押すと、閲覧バッファーに入って、表示されているどのグループにでも入ることができるはずです。

7.1.5 サーバー変数

変数を(バックエンドと Emacs 一般の両方で) 定義する際の一つのやっかいな点は、いくつかの変数は、概してその変数の定義がロードされるときに他の変数で初期化されることです。「基」になる変数がロードされた後でそれを変更しても、「派生」した変数は変更されません。

これは一般にディレクトリーやファイルの変数に影響します。例えばnnml-directoryはディフォルトでは~/Mail/ で、また、すべてのnnml ディレクトリー変数はその変数によって初期化されるので、nnml-active-file は~/Mail/active になります。新しいnnml 仮想サーバーを定義する場合、nnml-directory を設定するだけでは十分ではありません---あなたはすべてのファイル変数を、そうしたいと望んだ値に明示的に設定しなければなりません。それぞれのバックエンドのための完全な変数のリストを見るには、このマニュアルの後に続くそれぞれのバックエンドの部分を読んでください。でもnnml の定義の例はここにあります:

```
(nnml "public"
      (nnml-directory "~/my-mail/")
      (nnml-active-file "~/my-mail/active")
      (nnml-newsgroups-file "~/my-mail/newsgroups"))
```

サーバー変数はしばしば「サーバーパラメーター」と呼ばれます。

7.1.6 サーバーと選択方法

普通に選択方法を使う(例えば外部サーバーから記事を読むときにグループを選択する手段としてgnus-secondary-select-method 使う) 場面ではどこでも、代わりに仮想サーバーの名前を使うことができます。これによって、たくさんキーボードを叩かなくて済むかもしれません。そして、どんなときでもその方が良いです。

7.1.7 使用不可能なサーバー

あるサーバーに接続することができないように見えるとき、Gnus はそのサーバーに拒否された(denied) ことを記録します。その後でそのサーバーと接続しようとするどんな試みも、単に無視されます。実際にそうかどうかを少しも確かめずに、Gnus は「接続を開くことができません」と(英語で) 告げます。

それはずいぶんお行儀が悪いと思うかもしれませんが、たいていの場合は有意義なのです。例えば‘nephelococcdyia.com’ というサーバーで十個のグループを購読しているとしましょう。サーバーはどこかとても遠いところにあって、そのマシンはとても遅いので、今日それが接続を拒否するかどうかを調べるだけでも一分かかります。もし Gnus がそれを十回試すようになっていたとすると、とても煩わしいでしょう。ですから Gnus はそれを試そうとはしません。一度でも「接続が拒否された」(connection refused) という結果を受け取ったなら、それはサーバーが「落ちている」(down) のだ、とみなします。

では、一時的にそのマシンの機嫌が悪いだけだったら何が起こるのでしょう? マシンが復活したかどうかをどうすれば調べることができるのでしょう?

それには、サーバーバッファーに移動して(see Section 7.1 [Server Buffer], p. 146)、以下の命令で突いてみてください:

O 現在の行のサーバーとの接続を確立しようとします(gnus-server-open-server)。

C サーバーとの接続(もしあれば) を閉じます(gnus-server-close-server)。

D 現在のサーバーに接続不可の印を付けます(gnus-server-deny-server)。これは事実上そのサーバーを使わないようにします。

M-o バッファーにあるすべてのサーバーとの接続を開きます(gnus-server-open-all-servers)。

M-c バッファーにあるすべてのサーバーとの接続を閉じます(gnus-server-close-all-servers)。

R Gnus が接続を拒否されたすべてのサーバーの、すべての印を消去します(gnus-server-remove-denials)。

c サーバーをコピーして新しい名前を付けます(gnus-server-copy-server)。これは、複雑な接続方法の定義がすでにあって、それと同じ定義を異なる(物理)サーバーのために使う必要がある場合に役立つはずです。

L サーバーの状態をオフラインにします(gnus-server-offline-server)。

7.2 ニュースの取得

ニュースリーダーは普通はニュースを読むために使われます。Gnus は現在はニュースを取得するための二つの方法だけを提供しています—NNTP サーバーから、またはローカルスプールから読むことができます。

7.2.1 NNTP

NNTP サーバーから外部グループを購読するのは比較的簡単です。単に選択方法としてnntpを指定し、NNTP サーバーのアドレスを、うーん、アドレスとして指定するだけです。

NNTP サーバーが標準ではないポート(port) に設置されているときは、選択方法の三番目の要素をこのポートの番号に設定すれば、正しいポートに接続することができるでしょう。そのためにはグループ情報を編集しなければなりません(see Section 3.9 [Foreign Groups], p. 21)。

外部グループの名前は基本グループと同じでも構いません。実際、あなたの思うままに同じグループを可能な限りの違ったサーバーから購読することができます。名前の衝突は起こりません。

以下の変数は仮想nntp サーバーを作るために使われます:

nntp-server-opened-hook
: は接続ができた後に実行されます。それはNNTP サーバーに接続した後に、それに命令を送るために使うことができます。ディフォルトではMODE READER 命令が、nntp-send-mode-reader 関数によってサーバーに送られるようになっています。この関数は常にこのフックにあるべきです。

nntp-authinfo-function
: この関数はNNTP サーバーに‘AUTHINFO’ を送るために使われます。ディフォルトの関数はnntp-send-authinfo で、該当するエントリーを探すために~/.authinfo を調べます。もし一つも見つからなかったら、ログイン名とパスワードの入力を要求します。~/.authinfo ファイルの様式はftp のための~/.netrc ファイルと(ほとんど) 同じです。それはftp のマニュアルページで定義されていますが、ここに重要な事実があります:

  1. ファイルは一つ以上の行を含み、それぞれは一つのサーバーを定義します。
  2. それぞれの行は任意の数の標章(token) と値の対を含むことができます。

  有効な標章は‘machine’, ‘login’, ‘password’, ‘default’ です。加えて、Gnus は.netrc/ftp の構文の原型には現れない二つの新しい標章、名付けて‘port’ と‘force’ を導入します。(これが.authinfo ファイルの様式が.netrc フ

ァイルの様式から逸脱する唯一の方法です。) ‘port’ はサーバーのどのポートを認証に用いるかを示し、‘force’ は以下で説明します。

これがそのファイルの例です:

```
machine news.uio.no login larsi password geheimnis
machine nntp.ifi.uio.no login larsi force yes
```

標章と値の対はどんな順番ででも現れることができます。例えば‘machine’ が最初でなければならない必要はありません。

この例では、前者のサーバーにログイン名とパスワードの両方が与えられているのに対して、後者にはログイン名だけがあり、利用者はパスワードの入力を求められるでしょう。後者は‘force’ タグも持っていて、これによって接続時に*nntp* サーバーに認証情報(authinfo) が送られます。ディフォルト(すなわち、‘force’ タグが無いとき) では、*nntp* サーバーが認証情報を尋ねない限りそれを*nntp* サーバーに送りません。

‘machine’ 行に合致しないすべてのサーバーに適用される‘default’ 行を追加することもできます。

```
default force yes
```

これは、それ以前に書かれていないすべてのサーバーに‘AUTHINFO’ 命令を強制的に送ります。

~/.authinfo ファイルを世界中が読めるような設定のままで放置しないように注意してください。

nntp-server-action-alist
: これはサーバーの型に合致する正規表現と、合致が起こったときに取られる動作の連想リストです。例えば、Gnus に innd に接続したときに毎回ビープ音を鳴らしたいのであれば、次のようにすることができます:

```
(setq nntp-server-action-list
      '(("innd" (ding))))
```

まぁ、そんなことをしたいとは思わないでしょうけれどね。

ディフォルトの値は

```
'(("nntpd 1\\.5\\.11t"
   (remove-hook 'nntp-server-opened-hook
                'nntp-send-mode-reader)))
```

で、これは nntpd 1.5.11t にはMODE READER 命令を確実に送らないようにします。なぜなら、その命令はサーバーの息の根を止めると聞いているからです。

nntp-maximum-request
: もしNNTP サーバーがNOV ヘッダーをサポートしていないのであれば、このバックエンドはhead 命令をいくつも送って、ヘッダーを集めます。この動作を速くするために、バックエンドは返答を待たずにこの命令をたくさん送り、それからすべての返答を読みます。これは変数nntp-maximum-request によって制御され、ディフォルトで 400 です。もしネットワークの具合が良くないようなら、この変数を 1 に設定するべきでしょう。

nntp-connection-timeout
: 定期的に接続している外部nntp グループがたくさんあると、ちゃんと応答しなかったり常識的な時間内に返答できないくらいの負荷がかかってい

るNNTP サーバーの問題があるはずです。これはやっかいな問題をもたらしますが、nntp-connection-timeout を設定することによってある程度解消することができます。これは接続を諦める前に、nntp バックエンドが何秒待つかを示す整数です。もしこれがnil であると、それがディフォルトですが、時間切れによる切断は行ないません。

nntp-nov-is-evil
: NNTP サーバーがNOV をサポートしていない場合は、この変数をt に設定すれば良いでしょう。でもnntp は普通はNOV が使えるかどうかを自動的に調べます。(訳注: ですから、わざわざ設定しなくても構いません。)

nntp-xover-commands
: サーバーからNOV 行を取得するための命令として使われる文字列のリストです。この変数のディフォルトの値は("XOVER" "XOVERVIEW") です。(訳注: それらを順に試します。)

nntp-nov-gap
: nntp は、普通はサーバーにNOV 行のための一つの大きな要求を送ります。サーバーは一つの巨大な行のリストで応答します。しかし、グループの 2-5000 の記事を読んだ後で 1 と 5001 を読みたいだけだとしても、nntp は必要の無い 4999 個のNOV 行を取得することになります。この変数は、どれくらい大きな二つの連続した記事群の間の隔たり(gap) までXOVER の要求を分割せずに送るかを決定します。ネットワークが速い場合に、この変数を本当に小さな数値に設定してしまうと、おそらく取得が遅くなることに注意してください。この変数がnil ならば、nntp は要求を分割しません。ディフォルトは 5 です。

nntp-xref-number-is-evil
: ユーザーが指定したMessage-ID を持っている記事、または現在のものの親記事のMessage-ID を持っている記事を参照するとき(see Section 4.23 [Finding the Parent], p. 109)、Gnus はそれがどこにあるかを知るためにNNTP サーバーにHEAD コマンドを送ります。そしてサーバーは、Xref ヘッダーにグループと記事番号の対を含んでいるデータを返します。そのデータが、その記事が現在のグループにあることを示すなら、通常 Gnus はその記事を参照するのに記事番号を使用します、そうでなければMessage-ID を使いますが。ところが、あるニュースサーバー(例えば Diablo を実行するもの) は、同じ記事群を有する複数のエンジンを運転していて、それらの間では記事番号が同期されていません。その場合Xref ヘッダーに現われる記事番号は、どのエンジンが選ばれるかによって変化するので、例えば現在のグループにある親記事を参照することができません。そのようなサーバーに接続するのであれば、この変数をnil ではない値に設定してください。そうすれば Gnus は記事番号を使いません。例えば:

```
(setq gnus-select-method
      '(nntp "newszilla"
             (nntp-address "newszilla.example.com")
             (nntp-xref-number-is-evil t)
             ...))
```

このサーバー変数のディフォルト値はnil です。

nntp-prepare-server-hook
: NNTP サーバーに接続を試みる前に実行するフックです。

`nntp-record-commands`
: これを`nil` でない値にすると、`nntp` はNNTP サーバーに送ったすべての命令を(時刻と共に) `*nntp-log*` バッファーに記録します。これは動作していないように見える Gnus のNNTP 接続をデバッグしているときに役に立ちます。

`nntp-open-connection-function`
: どのようにNNTP サーバーと接続するかをカスタマイズすることができます。`nntp-open-connection-function` パラメーターを設定しておくと、Gnus は接続を確立するためにその関数を使います。そのために七つの関数があらかじめ用意されています。それらは二種類に分類することができ、直接接続するための関数群(四つ) と間接的に接続するためのもの(三つ) があります。

`nntp-never-echoes-commands`
: 非-`nil` でNNTP サーバーがコマンドをエコーバックしないことを意味します。報告によると、ある種のNNTPS サーバーはコマンドをエコーバックしないそうです。したがって、例えば`nntp-open-connection-function` を`nntp-open-tls-stream` に設定してあるそのようなサーバーのための選択方法の中で、この変数を非-`nil` に設定する必要があるでしょう。ディフォルト値は`nil` です。この変数の値`nil` は、`nntp-open-connection-functions-never-echo-commands` 変数で上書きされることに注意してください。

`nntp-open-connection-functions-never-echo-commands`
: コマンドをエコーバックしない関数のリストです。`nntp-open-connection-function` に設定した関数がコマンドをエコーバックしないならば、それをこのリストに加えてください。`nntp-never-echoes-commands` 変数の`nil` でない値が、この変数をくつがえすことに注意してください。ディフォルト値は`(nntp-open-network-stream)` です。

`nntp-prepare-post-hook`
: 記事を投稿する直前に実行されるフックです。もし記事に`Message-ID` ヘッダーが無くてニュースサーバーが推奨 ID を提供してくれるならば、このフックが実行される前にそれが記事に加えられます。これは、もしあなたが Gnus が`Message-ID` ヘッダーを付けないようにしていても、`Cancel-Lock` ヘッダーを作るために利用することができます。それにはこうすれば良いでしょう:

```
(add-hook 'nntp-prepare-post-hook 'canlock-insert-header)
```

すべてのサーバーが推奨 ID をサポートしているわけではないことに注意してください。これは例えば INN 2.3.0 以上で動作します。

`nntp-server-list-active-group`
: もし`nil` だったら‘`LIST ACTIVE`’ の代わりに常に‘`GROUP`’ を使います。これは普通遅いのですが、誤って active ファイルを頻繁に更新しないように設定されているサーバーでは助けになるはずです。

7.2.1.1 直接接続するための関数

これらの関数は、あなたのマシンとNNTP サーバーを接続するために直接呼ばれます。また、それらの動作はそれらが共通に参照する変数に影響されます(see Section 7.2.1.3 [Common Variables], p. 157)。

`nntp-open-network-stream`
: これはディフォルトで、単純に遠隔システムのポートもしくは別のものに接続します。もし Emacs とサーバーの両方がサポートしていれば、その接続は自動的に暗号化されたSTARTTLS 接続に昇格されます。もし悪意のある仲介者がSTARTTLS の使用をブロックする可能性を避けたいのであれば、代わりに`nntp-open-tls-stream` を使ってください。

`nntp-open-plain-stream`
`network-only`
: 上記に同じ。ただし自動でSTARTTLS への昇格は行ないません。あなたのトラフィックが誰かに読まれても構わない場合にのみこれを使ってください。

`nntp-open-tls-stream`
: 「安全な」チャンネルを使ってサーバーに接続します。これを使うためには、あなたの Emacs がGnuTLS (`https://www.gnu.org/software/gnutls/`) を使うようにコンパイルされていなければなりません。

  そしてサーバーを次のように定義します:

```
;; ポート 563 が "nntps" として /etc/services で定義済み
;; であっても、‘gnutls-cli -p’ でその名前は使えません。
;;
(nntp "snews.bar.com"
      (nntp-open-connection-function nntp-open-tls-stream)
      (nntp-port-number 563)
      (nntp-address "snews.bar.com"))
```

`nntp-open-ssl-stream`
: これは`nntp-open-tls-stream` の古い名前で、完全にそれと等価です。

`nntp-open-netcat-stream`
: `netcat` コマンドを使ってNNTP サーバーに接続します。ディフォルトの`nntp-open-network-stream` がそれをするのにもかかわらず、なぜこの関数があるのか不思議に思うかもしれません。その理由(の一つ) は、もしあなたが防壁の中にいたとしても`runsocks` のようなコマンドラッパーのおかげで外の世界を直接アクセスできるならば、それをこのように使うことができるのです:

```
(nntp "socksified"
      (nntp-pre-command "runsocks")
      (nntp-open-connection-function nntp-open-netcat-stream)
      (nntp-address "the.news.server"))
```

  ディフォルトのメソッドのままでそれを行なうには Emacs のセッション全体をラップする必要があるでしょうが、それは良い考えではありません。

`nntp-open-telnet-stream`
: `nntp-open-netcat-stream` に似ていますが、`netcat` ではなくて`telnet` を使います。行末コードを変更したりするので`telnet` はいささか堅実さに欠けるのですが、`netcat` が無い場合もあります。前の例はこのように書き換えられるでしょう:

```
(nntp "socksified"
      (nntp-pre-command "runsocks")
      (nntp-open-connection-function nntp-open-telnet-stream)
```

```
(nntp-address "the.news.server")
(nntp-end-of-line "\n"))
```

7.2.1.2 間接的に接続するための関数

これらの関数は、実際にNNTP サーバーに接続する前に中間のホストに接続するために間接的に呼ばれます。すべてのこれらの関数と関連する変数は“via”接続の仲間に属しているとも言えるので、それを明確にするためにすべて“via”という接頭語が付けられます。また、それらの動作はそれらが共通に参照する変数に影響されます(see Section 7.2.1.3 [Common Variables], p. 157)。

`nntp-open-via-rlogin-and-netcat`
: 遠隔システムに‘`rlogin`’ して、そこから本当のNNTP サーバーに接続するために`netcat` を使います。これは、例えばあなたが始めに防壁マシンに接続しなければならない場合に便利です。

  `nntp-open-via-rlogin-and-netcat`-用の変数:

  `nntp-via-rlogin-command`
  : 中間のホストにログインするために使われるコマンドです。ディフォルトは‘`rsh`’ ですが、‘`ssh`’ が人気のある代替手段です。

  `nntp-via-rlogin-command-switches`
  : `nntp-via-rlogin-command` のコマンドのスイッチとして使われる文字列のリストです。ディフォルトは`nil` です。もし‘`ssh`’を`nntp-via-rlogin-command` の値として使うならば、すべてのデータ接続を圧縮するために‘`("-C")`’ を使うことができます。

`nntp-open-via-rlogin-and-telnet`
: 本質的には同じことなのですが、中間のホストから本当のNNTP サーバーに接続するために、‘`netcat`’ の代わりに`telnet` を使います。行末コードを変更したりするので`telnet` はいささか堅実さに欠けるのですが、`netcat` が無い場合もあるでしょう。

  `nntp-open-via-rlogin-and-telnet`-用の変数:

  `nntp-telnet-command`
  : 中間のホストから本当のNNTP サーバーに接続するために使われるコマンドです。ディフォルトは‘`telnet`’ です。

  `nntp-telnet-switches`
  : `nntp-telnet-command` のコマンドのスイッチとして使われる文字列のリストです。ディフォルトは`("-8")` です。

  `nntp-via-rlogin-command`
  : 中間のホストにログインするために使われるコマンドです。ディフォルトは‘`rsh`’ ですが、‘`ssh`’ が人気のある代替手段です。

  `nntp-via-rlogin-command-switches`
  : `nntp-via-rlogin-command` のコマンドのスイッチとして使われる文字列のリストです。‘`ssh`’ を使う場合に、もし中間のホストで telnet コマンドが疑似端末を必要とするならば、これを‘`("-t" "-e" "none")`’ または‘`("-C" "-t" "-e" "none")`’ にする必要があるでしょう。ディフォルトは`nil` です。

nntp-end-of-line の値を‘\n’ に変更する必要があるであろうことに注意してください(see Section 7.2.1.3 [Common Variables], p. 157)。

nntp-open-via-telnet-and-telnet
: これもまた本質的には同じことなのですが、中間のホストに接続するために‘rlogin’ の代わりに‘telnet’ を使います。

  nntp-open-via-telnet-and-telnet-用の変数:

  nntp-via-telnet-command
  : 中間のホストにtelnet するために使われるコマンドです。ディフォルトは‘telnet’ です。

  nntp-via-telnet-switches
  : nntp-via-telnet-command のコマンドのスイッチとして使われる文字列のリストです。ディフォルトは‘("-8")’ です。

  nntp-via-user-password
  : 中間のホストにログインするときに使われるパスワードです。

  nntp-via-envuser
  : もし非-nil なら、中間のtelnet のセッション(クライアントとサーバーの両方) でENVIRON オプションをサポートし、ログイン名の入力を要求しません。これは例えば Solaris のtelnet で動作します。

  nntp-via-shell-prompt
  : 中間のホストでのシェルのプロンプトに合致する正規表現です。ディフォルトは‘bash\\|\$ *\r?$\\|> *\r?’ です。

  nntp-end-of-line の値を‘\n’ に変更する必要があるであろうことに注意してください(see Section 7.2.1.3 [Common Variables], p. 157)。

これらは上記のすべての関数が参照する付加的な変数です:

nntp-via-user-name
: 中間のホストに接続するときに使う利用者名です。

nntp-via-address
: 接続する中間のホストのアドレスです。

7.2.1.3 共通の変数

以下の変数は、すべての、またはいくつかのあらかじめ用意されている関数の動作に影響を及ぼします。設定されていなければ、すべての関数が影響されます(それぞれの仮想サーバーにおいて、サーバー変数として個々に値が設定されていない場合に、以下の値がディフォルトで使われます)。

nntp-pre-command
: 素の接続用の関数ではないもの(nntp-open-network-stream、nntp-open-tls-stream またはnntp-open-ssl-stream 以外のすべて) を通して接続するときに使うコマンドラッパーです。例えばあなたは‘SOCKS’ ラッパーを割り当てるでしょう。(訳注: telnet などの外部コマンドに被せて使われます。)

nntp-address
: NNTP サーバーのアドレスです。

`nntp-port-number`
: 接続するNNTP サーバーのポート番号です。ディフォルトは‘`nntp`’ です。TLS/SSL を介したNNTP を使うには、ポートの名前ではなくて整数(つまり‘`snews`’ や‘`nntps`’ ではなくて‘`563`’) を指定する必要があります。外部のTLS/SSL ツールはポートの名前では動作しないからです。

`nntp-end-of-line`
: NNTP サーバーとお話をしているときに行の終わりの印として使われる文字列です。これはディフォルトで‘`\r\n`’ ですが、素ではない接続用の telnet 同等の関数を使っているときは‘`\n`’ であるべきです。

`nntp-netcat-command`
: ‘`netcat`’ を通してNNTP サーバーと接続するときに使うコマンドです。これは中間のホストと接続するためのものではありません。これはまさに本当のNNTP サーバーと接続するためのものです。ディフォルトは‘`nc`’ です。

`nntp-netcat-switches`
: `nntp-netcat-command` に渡すスイッチのリストです。ディフォルトは‘`()`’ です。

7.2.2 ニューススプール

ローカルスプールから外部グループを購読することは極めて簡単だし便利かもしれません。非常に大きな記事があるグループ---例えば‘`alt.binaries.pictures.furniture`’ を読む速度が速くなります。

とにかく、`nnspool` を選択方法として、かつ`""` (もしくは何でも) をアドレスとして指定するだけです。

もしローカルスプールにつなぐことが可能なら、おそらくそれを基本選択方法として使うべきでしょう(see Section 2.1 [Finding the News], p. 3)。それは普通は`nntp` 選択方法を使うより速いですが、そうでないかもしれません。それは場合によります。何があなたのサイトで一番良いかを見つけるために、いろいろと試してみなければなりません。

`nnspool-inews-program`
: 記事を投稿するために使われるプログラムです。

`nnspool-inews-switches`
: 記事を投稿するときに inews プログラムに与えられるパラメーターです。

`nnspool-spool-directory`
: `nnspool` が記事を探すところです。これは普通は`/usr/spool/news/` です。

`nnspool-nov-directory`
: `nnspool` がNOV ファイルを探すところです。これは普通は`/usr/spool/news/over.view/` です。

`nnspool-lib-dir`
: ニュースのライブラリーが置かれているディレクトリーの場所です(ディフォルトで`/usr/lib/news/` です)。

`nnspool-active-file`
: アクティブファイルの絶対パス名です。

`nnspool-newsgroups-file`
: `newsgroups` ファイルの絶対パス名です。

`nnspool-history-file`
: `history` ファイルの絶対パス名です。

`nnspool-active-times-file`
: `active.times` ファイルの絶対パス名です。

`nnspool-nov-is-evil`
: `nil` でないと、`nnspool` はそれが見つけたどんなNOV ファイルも使おうとはしません。

`nnspool-sift-nov-with-sed`
: `nil` でないと、これがディフォルトですが、概観ファイル(overview) から関連する部分を得るために`sed` を使います。もし`nil` だと、`nnspool` はファイル全体をバッファーに読み込んで、そこで実行します。

7.3 IMAP を使う

おそらく最も人気があるメール・バックエンドは`nnimap` です。それはIMAP サーバーに接続します。IMAP サーバーはメールをサーバーに格納するので、クライアントは何もローカルに格納しません。もし、いろいろな場所から、またはいろいろなユーザー・エージェントでメールを読んでいるのであれば、それは便利な選択だと言えます。

7.3.1 IMAP サーバーに接続する

IMAP サーバーへの接続はとても簡単なはずです。グループバッファーで*B* を叩くか、または(もしあなたの第一の関心事がメールを読むことであるなら) 以下のようなことをしてください:

```
(setq gnus-select-method
      '(nnimap "imap.gmail.com"))
```

ユーザー名とパスワードを尋ねられます。それに飽きたなら、以下のものを`~/.authinfo` ファイルに加えてください:

```
machine imap.gmail.com login <username> password <password> port imap
```

ほとんどのユーザーには、それだけで良いはずです。

7.3.2 IMAP 接続をカスタマイズする

もっと複雑な接続方法の例:

```
(nnimap "imap.gmail.com"
        (nnimap-inbox "INBOX")
        (nnimap-split-methods default)
        (nnimap-expunge t)
        (nnimap-stream tls))
```

`nnimap-address`
: サーバーのアドレスです。‘`imap.gmail.com`’ のようなもの。

`nnimap-user`
: IMAP サーバーへの認証に使うユーザー名です。これは`~/.authinfo` ファイルの‘`login`’ トークンの値に相当します。同じIMAP サーバーから複数のアカウントにアクセスする場合に、この変数を設定してください。

nnimap-server-port
: サーバーのポートが標準とは異なるなら、ここで設定できます。代表的なポートは"imap" あるいは"imaps" でしょう。

nnimap-stream
: nnimap がどうやってサーバーに接続するかを指定します。使える値は:

  undecided
  : これがディフォルトです。最初にtls 設定を、次にnetwork 設定を試します。

  tls
  : 標準のTLS/SSL 接続を使います。ssl は等価ではありますが、これを設定する方法としては推奨されません。

  network
  : 暗号化されないので安全ではない、ストレートなソケット接続です。しかし、もし Emacs とサーバーの両方がサポートしていれば、暗号化されたSTARTTLS 接続に昇格します。

  starttls
  : 正規のIMAP ポート上で暗号化されたSTARTTLS を使います。

  shell
  : もしサーバーに接続するために他のホストをトンネルする必要があるなら、このオプションを使ってnnimap-shell-program を必要に応じてカスタマイズすることができます。

  plain
  : 暗号化されておらず安全ではないストレートな接続。たとえSTARTTLS が利用可能でも使いません。

nnimap-authenticator
: いくつかのIMAP サーバーは匿名ログインを許容しています。その場合、これをanonymous に設定する必要があります。もしこの変数を設定しないと標準のログイン手順が使われます。特定のログイン手順を指定したいなら、この変数をlogin (伝統的なIMAP のログイン手順)、plain、cram-md5、またはxoauth2のどれかを設定してください。(最後のものはoauth2.el ライブラリーを使う必要があります。)

nnimap-expunge
: 消去した記事を、いつ永久に取り除くか。もしnever であれば、消去された記事にはIMAP \\Delete フラグが付きますが、自動的に永久に取り除かれることはありません。もしimmediately であると、消去された記事は直ちに永久に取り除かれます(これにはサーバーがUID EXPUNGE コマンドをサポートしている必要があります)。もしon-exit であると、記事は消去したときにフラグが付けられ、フラグが付けられたすべての記事はそのグループを閉じるときに永久に取り除かれます。

  この変数は後方互換性のためにt またはnil にすることもできます。もしサーバーがUID EXPUNGE をサポートしているならばt とnil の両方ともimmediatelyと等価です。もしサーバーがUID EXPUNGE をサポートしていないならばnilはnever と等価ですが、t は現在消去済みとしてフラグが付けられているすべて の記事(つまり消去されたばかりの記事だけでなく) を直ちに永久に取り除きます。

`nnimap-streaming`
: 事実上すべてのIMAP サーバーはデータの高速ストリーミングをサポートしています。もしサーバーへの接続に問題があるのなら、これを`nil` に設定してみてください。

`nnimap-fetch-partial-articles`
: もし`nil` 以外の値だったら、サーバーから記事の部分を取り込みます。もし文字列が設定されていたら、それは正規表現であると解釈され、そのタイプが合致する部分が取り込まれます。例えば‘`"text/"`’ 場合はすべてのテキスト型の部分を取り込みますが、残りはサーバーに置いたままになります。

`nnimap-record-commands`
: もし`nil` でなければ、すべてのIMAP コマンドを‘`"*imap log*"`’ バッファーに記録します。

`nnimap-use-namespaces`
: もし`nil` でなければ、nnimap グループ名において IMAP 名前空間の接頭辞を省略します。もしあなたのメールボックスが‘`INBOX`’ や‘`INBOX.Lists.emacs`’ のように名付けられているとしても、nnimap グループ名を‘`INBOX`’ や‘`Lists.emacs`’ のようにしたいのであれば、このオプションを有効にしてください。

`nnimap-keepalive-intervals`
: ディフォルトでは、nnimap は接続を維持するために時々‘`NOOP`’ (keepalive) コマンドをサーバーに送信します。このオプションは、それが発生する頻度を管理します。これは秒を表す 2 つの整数の cons で、1 つ目は keepalive チェックを実行する頻度、2 つ目はユーザーがどのくらいの期間にわたって非アクティブ状態だったら実際にコマンドを送信するかを指定する秒数です。ディフォルトの`(900 . 300)` は、15 分ごとにチェックを実行し、ユーザーが 5 分間非アクティブだったら keepalive コマンドを送信することを意味します。完全に keepalive コマンドを完全に無効にするには`nil` に設定してください。

7.3.3 クライアント側での IMAP 分割

多くの人々が、メールをIMAP サーバー上のそれぞれのメールボックスに並び換え/分割することを好みます。そうすれば、さほど関心が無いメールをダウンロードする必要がありません。

もしクライアント側でメール分割をする必要があるのなら、関連する変数は次の通りです:

`nnimap-inbox`
: 新着メールが調べられるIMAP メールボックスです。これはメールボックス名のリストでもかまいません。

`nnimap-split-methods`
: `nnmail-split-methods` と同じ構文を使います(see Section 7.4.3 [Splitting Mail], p. 164)。ただし`default` というシンボルである場合は例外で、`nnmail-split-methods` の値を使います。

`nnimap-split-fancy`
: `nnmail-split-fancy` と同じ構文を使います。

`nnimap-unsplittable-articles`
: 分割を行なっているときに無視すべきフラグのシンボルのリスト。すなわち、それらのフラグをもっている記事は、分割を行なう対象として考慮されないということです。ディフルトは‘`(%Deleted %Seen)`’ です。

ディフォルトでは nnimap バックエンドはメッセージヘッダーのみを取得します。オプション`nnimap-split-download-body` (これは通常のカスタマイズのオプションであって、サーバー変数ではありません) は、メッセージの本文も取得するように指示します。これはIMAP の速度を低下させるのでディフォルトでは設定されていませんし、ユーザーのための適切な判断ではありません。

以下は、クライアント側で“特級” メール分割を行なう`nnimap` バックエンドの完全な例です:

```
(nnimap "imap.example.com"
        (nnimap-inbox "INBOX")
        (nnimap-split-fancy
         (| ("MailScanner-SpamCheck" "spam" "spam.detected")
            (to "foo@bar.com" "foo")
            "undecided")))
```

7.3.4 IMAP 拡張のサポート

Google の Gmail を使っていると、メールを読んでいるときに Gmail ラベルを見たいときがあるでしょう。Gnus はその情報を提供することができます。そのためには変数`gnus-extra-headers` の中で‘`X-GM-LABELS`’ を要求してください。例えばこんなふうに:

```
(setq gnus-extra-headers
      '(To Newsgroups X-GM-LABELS))
```

これによって、後に参照できるように Gnus はメッセージヘッダーにラベルを追加します。その内容は常に括弧で囲まれたリストになります(中身が無い場合もあります。)。

7.4 メール取得

ニュースリーダーでメールを読むなんて実に奇妙ですよね? いや、もちろんできるのですが。

7.4.1 ニュースリーダーでメール

使い慣れた伝統的なメールリーダーから Gnus に乗り換えることを決断したならば、かなりのカルチャーショックを経験することになるでしょう。

Gnus は伝統的なメールリーダーのようなふるまいをしません。あなたが望むならそのようにもできますが、それは骨折り損のくたびれ儲けです。

Gnus はふつう同じ手法ですべてのグループを扱います。あるグループを選んで新しい、または未読のメッセージを読むと、それらには既読の印が付けられ、(意図的に要求しなければ) 以後はそれらを目にすることはありません。これってとてもニュースリーダー的でしょ。

メッセージを消すために、取り立てて何かを行なうことはありません。

このことは既読のメッセージはすべて消されてしまうことを意味するのかって? そりゃあんまりですよね!

しかし、そうではありません。古いメッセージは何らかの仕組みによって期限切れ消去(*expire*) されるのです。ニュースのメッセージはニュースの管理人(が管理しているサーバー) によって期限切れ消去の処理が制御され、メールの期限切れ消去の処理はあなたが制御します。メールの期限切れ消去については、Section 7.4.9 [Expiring Mail], p. 181, で徹底的に網羅されています。

多くの Gnus の利用者が、それをニュースとメールの両方でしばらく使ってみた後で気が付くのは、その配送の機構がメッセージの扱い方に関して行なうことが、ほんの少ししか無いことです。

多くの人たちが複数のメーリングリストを購読しています。それらはSMTP で配送されるもの、すなわちメールです。それらのメッセージに返答をしないまま、あるいはさらに、それらを非常に注意深くは読まないままに、私たちは何週間も過ごすかもしれません。でも、そういうメッセージを保存しておく必要はありません。なぜならば、もう一度読む必要が生じたとしても、それらはどこかに保存されているからです。

ある人たちは小人数に利用されているローカルニュースグループを購読しています。それらはNNTP で配送されるもの、すなわちニュースです。私たちは自分の仕事に役立てるために、それらの膨大なメッセージの断片を読んだり返事をしなければなりません。しかもそれらがどこかに保存されているとは限らないので、興味のあるメッセージを個人メールと同じように保存しなければなりません。

配送の仕組みの違いはどうでもよいことで、大事なのはいかに主題に興味を持っているかと、もう一度読みたいときにいかに簡単に呼び出せるかなのです。

Gnus はメールをニュースグループのように「グループ」に並べ変えて、各々のグループ(メールかニュース) を別個に扱うための豊富な機能を提供します。

ある人たちは Gnus (えっへん) のやりかたに満足できなくて、Gnus が男(male) になること、もとい、メールリーダーになることを欲します。Gnus をもっとメールリーダー的なものにするために鞭打つことは可能ではあるのですが、前にも言ったように簡単ではありません。いわゆるメールリーダーが好みならばVM を使いましょう。これは優秀な、厳密な意味でのメールリーダーです。

脅かすわけではないのですが、はっきりさせておきたいのは、あなたにメッセージについての新しいやり方を修得して欲しいということです。あなたが Gnus のやり方を受け入れてくれた暁には、きっとあなたは Gnus が好きになるでしょう。請け合いますよ。(少くとも、私が Gnus に入れた Emacs のサブリミナル脳味噌洗濯関数を売ってくれた人はそれを保証しています。あなたも同化します。あなたは Gnus を愛します。あなたは Gnus でのメールの方法を愛します。絶対に。)

7.4.2 メールを読むことを始める

Gnus を使って新しいメールを読むことはまったく簡単です。あなたが選んだメールバックエンドを`gnus-secondary-select-methods` に放り込むだけで、自動的に読むことができるようになります。

例えば`nnml` (これは「一メールにつき一ファイル」のバックエンドです) を使いたいなら、次のものを`~/.gnus.el` ファイルに入れれば良いでしょう:

```
(setq gnus-secondary-select-methods '((nnml "")))
```

そうすると、次に Gnus を起動したときにこのバックエンドは新しい記事を求め、すべてのメッセージをスプールファイルからそのディレクトリー(ディフォルトでは`~/Mail/`) に移します。新しいグループ(‘`mail.misc`’) が作られて、他のグループと同じように読むことができるようになります。

たぶんメールをいくつかのグループに分割したいでしょうけれど:

```
(setq nnmail-split-methods
      '(("junk" "^From:.*Lars Ingebrigtsen")
        ("crazy" "^Subject:.*die\\^Organization:.*flabby")
        ("other" "")))
```

これは三つの新しいnnml メールグループ‘nnml:junk’, ‘nnml:crazy’ および‘nnml:other’を作ることになります。初めの二つのグループにふさわしくないすべてのメールは、最後のグループに置かれます。

Gnus でメールを読むにはこれで十分なはずです。マニュアルのこの部分の他の章を熟読する必要があるかもしれませんが。特にSection 7.4.13 [Choosing a Mail Back End], p. 187, とSection 7.4.9 [Expiring Mail], p. 181, を。

7.4.3 メールの分割

> 訳注: このマニュアルの多方面で使われている「分割」という語のうち、受信したメールをいろいろなグループに「区分け」することを意味するものは“split”という語に対応します。ある一つのメールを「分解」するのではなくて、外からやって来た複数のメールをそれぞれの格納先に一通ずつ「振り分ける」意味で使っています。

変数nnmail-split-methods は入ってくるメールをどのようにグループ分けするかを指定します。

```
(setq nnmail-split-methods
  '(("mail.junk" "^From:.*Lars Ingebrigtsen")
    ("mail.crazy" "^Subject:.*die\\|^Organization:.*flabby")
    ("mail.other" "")))
```

この変数はリストのリストで、それぞれのリストの最初の要素はメールグループの名前、二つめの要素はそれぞれのメールがそのグループに属するかどうかをヘッダーで判定するために使う正規表現です(ところで、メールグループの名前が‘mail’ で始まる必要はありません)。最初の文字列は、replace-match が合致した文章から取り出した副表現を挿入するために使われるような、‘\\1’ の様式を含むかもしれません。例えば:

```
("list.\\1" "From:.* \\(.*\\)-list@majordomo.com")
```

この場合、挿入されるテキストを小文字にすべきかどうかをnnmail-split-lowercase-expanded が制御します。See Section 7.4.6 [Fancy Mail Splitting], p. 175.

二番目の要素は関数でも構いません。その場合、それは規則の最初の要素(メールグループの名前) を引数として、ヘッダーだけに範囲を狭められたバッファーで呼ばれます。メールがそのグループに属すると判断したら、その関数はnil でない値を返す必要があります。

これらのグループの最後は常に総合的なものであるべきで、その正規表現は他の正規表現に合致しなかったメールに合致するために、いつも ‘""’ でなければなりません。(これらの規則は連想リストの初めから終わりまで順番に処理されます。クロスポストを有効にしていなければ、合致する最初の規則が「勝ち」ます。) もし合致する規則がなかったら、メールは最後に‘bogus’ グループに入ります。メール分割によって新しいグループが作られた場合は、それらを見るためにgnus-group-find-new-groups を実行する必要があるでしょう。これは‘bogus’ グループにも当てはまります。

あなた自身でこれをいじくりまわしたいときは、あなたの選んだ関数をこの変数に設定することができます。この関数は入って来たメールメッセージのヘッダーに範囲を狭めら

れたバッファーで、引数無しで呼ばれます。この関数は、そのメールメッセージをが行くべきだと判断するグループ名のリストを返さなければなりません。

この変数は特級メール分割であっても良いです。構文についてはSee Section 7.4.6 [Fancy Mail Splitting], p. 175.

すべてのメールバックエンドは、入って来た気の毒な無実のヘッダーを乱暴に扱っても良いことに注意してください。それらはすべてLines ヘッダーを追加します。いくつかはX-Gnus-Group ヘッダーを加えます。たいていのものは Unix の mbox のFromSPC 行を何か別のものに変えます。

すべてのメールバックエンドはクロスポストをサポートします。複数の正規表現が合致すると、メールはそれらすべてのグループに「クロスポスト」されます。nnmail-crosspost はこの機能を使うかどうかを指定します。どの記事も総合の(‘""’) グループにクロスポストされないことに注意してください。

nnmh とnnml はクロスポストされた記事にハードリンク(hardlink) を作ることによってクロスポストを行ないます。しかし、すべてのファイルシステムがハードリンクの機能を提供しているわけではありません。もしあなたがその場合に当てはまるのであれば、nnmail-crosspost-link-function をcopy-file に設定してください。(この変数はディフォルトでadd-name-to-file です。)

以前に行なわれたメール分割がメッセージをどこに入れたかを見たい場合は、*M-x nnmail-split-history* 命令を使ってください。これからスプールし直そうとするメッセージがどこに入るかを見たい場合は、gnus-summary-respool-trace および関連する命令(see Section 4.26 [Mail Group Commands], p. 114) を使ってください。

nnmail-split-header-length-limit の制限より長いヘッダー行は、分割関数の処理対象から除外されます。

ディフォルトでは分割の処理においてヘッダーをMIME デコードしないので、非-ASCII 文字列に合致させることができません。しかし、生のヘッダーのデータを元に記事の合致を判定したい場合には役立つでしょう。それを可能にするにはnnmail-mail-splitting-decodes 変数をnil ではない値に設定してください。加えてnnmail-mail-splitting-decodes がnil ではない場合に、nnmail-mail-splitting-charset 変数の値がMIME ではないエンコードされた文字列(訳注: iso-2022-jp でエンコードされた生のデータなど) をデコードするために使われます。ディフォルトはnil で、MIME ではないエンコードされた文字列をデコードしません。あなたにとって好都合な値はおそらくundecided か、またはあなたが興味があるメールで通常使われている文字セット(訳注: 実際は coding system) になるでしょう。

ディフォルトでは入ってくるすべてのメッセージに対して分割の処理が行なわれます。しかし、もしmail-sources 変数(see Section 7.4.4.1 [Mail Source Specifiers], p. 166) にdirectory の項目を設定すると、ディフォルトでは分割は行なわれません。変数nnmail-resplit-incoming をnil ではない値に設定すれば、この場合でも分割を起こさせることができます。(この変数は他の種類の項目に対しては効果がありません。)

Gnus はあなた自身に災いが及ぶ可能性あっても、あなたが望むすべての機会を提供します。例えば、あなたの上司からくるすべてのメールを入れるグループを作ったとしましょう。その後、偶発的にそのグループの購読をやめてしまうとどうなるでしょう。それでもGnus は上司からのすべてのメールを未購読のグループに入れるので、上司が「月曜日までにその報告書を準備しないと首だ!」というメールをあなたに送っても、あなたはそれを見ることはなく、火曜日になって本当は翌月の家賃を払うために空のボトルを集めるべきであっても、まだ有給で雇われていると信じているかもしれません。

7.4.4 メールソース

メールはたくさんの別のソース(source) から取得することができます---例えばメールスプールから、POP メールサーバーから、procmail ディレクトリーから、maildir から。

7.4.4.1 メールソース指示子

「メールソース指示子」に**メールソース** (see Section 7.4.4.4 [Fetching Mail], p. 173) を設定して、Gnus にメールを取得する方法を指示しましょう。

例です:

```
(pop :server "pop3.mailserver.com" :user "myname")
```

ご覧のように、メールソース指示子はリストで、最初の要素は「メールソースの型」、それに任意の数の「キーワード」が続きます。明示的に指定されていないキーワードはディフォルト値になります。

`mail-sources` はすべてのグループに対して共通です。しかし特定のグループのために、`mail-sources` に`group` メールソース指示子を持たせて、かつ単一のメールソースを指定する`mail-source` グループパラメーター(see Section 3.10 [Group Parameters], p. 23) を設定することによって、メールソースを追加することができます。これを使う場合の`mail-sources` は、一般には単なる`((group))` です。そしてグループのための`mail-source` パラメーターはこのようなものになるでしょう:

```
(mail-source . (file :path "home/user/spools/foo.spool"))
```

これは、そのグループは(このグループだけは) メッセージをスプールファイル‘`/user/spools/foo.spool`’ から取得することを意味します。

以下のメールソースの型が使用可能です:

`file` 単一のファイルからメールを取得します。普通はメールスプールです。

キーワード:

`:path` ファイルの名前です。ディフォルトは`MAIL` 環境変数の値か`rmail-spool-directory` の値(普通は`usr-mail/spool/user-name` のようなもの) です。

`:prescript`
`:postscript`
それぞれのメールを取得する前と後で実行するスクリプトです。

ファイルメールソースの例:

```
(file :path "/usr/spool/mail/user-name")
```

もしくは、ディフォルトのファイル名を使うと:

```
(file)
```

メールスプールファイルがローカルマシンに無い場合は、POP やIMAP などでメールを取得するのが最善です。ここでは ange-ftp のファイル名は使用できません---メールを移動しているときにメールスプールをロックする方法がありません。

適当なサーバーを設置することが不可能なら、変わりに ssh を使うことができます。

```
(setq mail-sources
```

```
          '((file :prescript "ssh host bin/getmail >%t")))
```

‘getmail’ スクリプトは以下のようなものになるでしょう:

```
#!/bin/sh
#  getmail - move mail from spool to stdout
#  flu@iki.fi

MOVEMAIL=/usr/lib/emacs/20.3/i386-redhat-linux/movemail
TMP=$HOME/Mail/tmp
rm -f $TMP; $MOVEMAIL $MAIL $TMP >/dev/null && cat $TMP
```

あなたが使いたい‘movemail’ と一時ファイルに合わせて、スクリプトを書き換えてください。

`directory`

特定のディレクトリーにある複数のファイルからメールを取得します。これは普通は procmail に新しいメールをいくつかのファイルに分割させているときに使われます。すなわち、そのディレクトリーにあるファイルとグループは一対一で対応しているので、`foo.bar.spool` ファイルにあるメールは`foo.bar` グループに置かれます(接尾語の`.spool` は変更可能です)。`nnmail-scan-directory-mail-source-once` を`nil` 以外の値にすると、Gnus に新しいメールソースを一回だけ調べさせることができます。これは特に、指定したレベルのメールグループだけを調べたいときに便利です。

`nnmail-resplit-incoming` という変数もあり、これを非-`nil` にすると通常の分割処理がそのディレクトリーにあるすべてのファイルに対して行なわれます(see Section 7.4.3 [Splitting Mail], p. 164)。

キーワード:

`:path` ファイルがあるディレクトリーの名前です。これにはディフォルト値はありません。

`:suffix` この接尾語で終わる名前ファイルだけが使われます。ディフォルトは‘`.spool`’ です。

`:predicate`
この述語が`nil` でない値を返すファイルだけが使われます。ディフォルトは`identity` です。これは追加の選別器として使用されます---正しい接尾語で、かつ この述語を満足するファイルだけが対象になります。

訳注: この場合の述語は関数で、正しい接尾語を持つファイルの名前が引数として渡されます。

`:prescript`
`:postscript`
それぞれのメールを取得する前と後で実行するスクリプトです。

ディレクトリーメールソースの例です:

```
(directory :path "/home/user-name/procmail-dir/"
           :suffix ".prcml")
```

`pop` POP サーバーからメールを取得します。

キーワード:

`:server` POP サーバーの名前です。ディフォルトは`MAILHOST` 環境変数から取得されます。

`:port` POP サーバーのポート番号です。これは数値(例えば‘`:port 1234`’)か文字列(例えば‘`:port "pop3"`’) です。もし文字列なら Unix システムにおける`/etc/services` に載っているサービス名でなければなりません。ディフォルトは‘`"pop3"`’ です。システムによっては‘`"pop-3"`’ としなければならないかもしれません。

`:user` POP サーバーに与える利用者名です。ディフォルトはログイン名です。

`:password`
POP サーバーに与えるパスワードです。設定しないと利用者は入力を求められます。

`:program` POP サーバーからメールを取得するために使用されるプログラムです。これは`format` で使うような文字列でなければなりません。例です:

```
fetchmail %u@%s -P %p %t
```

有効な書法仕様指示子は:

‘`t`’ メールがそこに移動させられるファイルの名前です。これは常にこの文字列に含まれていなければなりません。

‘`s`’ サーバーの名前です。

‘`P`’ サーバーのポート番号です。

‘`u`’ 使用する利用者名です。

‘`p`’ 使用するパスワードです。

これらの仕様で使われる値は、対応するキーワードに与えた値から取られます。

`:prescript`
メールを取得する前に実行されるスクリプトです。構文は`:program` キーワードと同じです。これは実行される関数であることもできます。

これの良く知られている使い道は、POP サーバーに接続するための SSH トンネルをお膳立てすることです、ここに例があります:

```
(pop :server "127.0.0.1"
     :port 1234
     :user "foo"
     :password "secret"
     :prescript
     "nohup ssh -f -L 1234:pop.server:110 remote.host sleep 3600 &
```

(訳注: 原典は nntp サーバー news.gmane.io の gmane.emacs.gnus.general グループに保存されている 81249 番の記事にあります。)

`:postscript`
メールを取得した後で実行されるスクリプトです。構文は`:program`キーワードと同じです。これは実行される関数であることもできます。

`:function`
POP サーバーからメールを取得するために使う関数です。その関数は一つのパラメーター(メールがそこへ移動されるべきファイルの名前) とともに呼ばれます

`:authentication`
これは`password` かシンボル`apop` のどちらかで、何の認証方式を使うかを指示します。ディフォルトは`password` です。

`:leave` メールを取得後もサーバーに残すには`nil` 以外の値にします。組み込み関数`pop3-movemail` (ディフォルト) だけがこのキーワードをサポートします。

もしこれが数値だったら、最初に新着メールをチェックしたときからその日数だけメールをサーバーに残します。その場合はUIDL で制御されるので、一度取得したメールを再び取り込むことはありません。`nil` の場合(それがディフォルト)、サーバーにあるメールは取得後すぐに削除されます。これが`nil` でも数値でもない場合はすべてのメールをサーバーに残すので、あなたは同じメールを何度も受け取る羽目になるでしょう。

`pop3-uidl-file` 変数でUIDL データをローカルに格納するファイルを指定します。ディフォルト値は`~/.pop3-uidl` です。

POP サーバーはセッションとセッションの間の状態の情報を維持しないので、クライアントがそこにあると思っているものと実際にあるものが一致しないかもしれないことに注意してください。それらが一致しないと、メールをダブって受け取るか、またはすべてが崩壊して、あなたは壊れたメールボックスとともに置き去りにされる可能性があります。

`:program` と`:function` キーワードが指定されていない場合は`pop3-movemail` が使われます。

メールをPOP サーバーから取得するための、いくつかの例を挙げます。

ディフォルトの利用者名を使って、ディフォルトのPOP サーバーから取得し、ディフォルトの取得方法を使用します:

```
(pop)
```

指名したサーバーから、指名した利用者とパスワードで取得します:

```
(pop :server "my.pop.server"
     :user "user-name" :password "secret")
```

サーバーに 14 日間メールを残します:

```
(pop :server "my.pop.server"
     :user "user-name" :password "secret"
     :leave 14)
```

メールの移動に‘movemail’ を使います:

```
(pop :program "movemail po:%u %t %p")
```

maildir Maildir からメールを取得します。これは少なくとも qmail と postfix によってサポートされているメールボックスの形式で、特別のディレクトリーにあるそれぞれのファイルは、厳密に一通のメールを含んでいます。

キーワード:

:path メールが保存されるディレクトリーの名前です。ディフォルトは環境変数MAILDIR から取得した値か、または~/Maildir/ です。

:subdirs Maildir のサブディレクトリーです。ディフォルトは‘("new" "cur")’です。

リモートマシンからメールを取り寄せることも出来ます。(というのも、maildir はロックの問題を気にせずに済むからです。)

Maildir メールソースの例をふたつ:

```
(maildir :path "/home/user-name/Maildir/"
         :subdirs ("cur" "new"))
(maildir :path "/user@remotehost.org:~/Maildir/"
         :subdirs ("new"))
```

imap IMAP サーバーからメールを取得します。何らかの理由で、IMAP をそれが意図されたようなネットワーク上でメールを読むプロトコルとしては(すなわちnnimap で) 使いたくないときは、Gnus ではPOP サーバーと同様に扱って、指定されたIMAP メールボックスから記事を取得することができます。詳しくはSection 7.3 [Using IMAP], p. 159, を参照してください。

キーワード:

:server IMAP サーバーの名前。ディフォルトは環境変数MAILHOST から得ます。

:port IMAP サーバーのポート番号。普通はディフォルトは‘143’で、TLS/SSL 接続には‘993’ です。

:user IMAP サーバーに渡す利用者名です。ディフォルトはログイン名です。

:password
IMAP サーバーに渡すパスワードです。指定されていないときは、利用者は入力することを促されます。

:stream サーバーに接続するときに使うストリーム。imap-stream-alist にあるシンボルの中のひとつを設定します。現状では‘gssapi’, ‘kerberos4’, ‘starttls’, ‘tls’, ‘ssl’, ‘shell’ またはディフォルトの‘network’ になります。

:authentication
サーバーでの認証にどの認証法を使うか。これにはimap-authenticator-alist で定義されているシンボルの一つを設定します。現状では‘gssapi’, ‘kerberos4’, ‘digest-md5’, ‘cram-md5’, ‘plain’, ‘anonymous’ またはディフォルトの‘login’ になります。

`:program` :stream に‘`shell`’ が設定されているときは、この値が変数`imap-shell-program` に割り当てられます。これは`format` ふうの文字列(または文字列のリスト) でなければなりません。例を示しましょう:

```
ssh %s imapd
```

何物もそのプログラムの出力を邪魔しないようにしてください。例えばエラー出力は void に振り分けましょう。有効な書法仕様指示子は以下の通りです。

‘`s`’ サーバーの名前。

‘`l`’ `imap-default-user` で設定された利用者名。

‘`p`’ サーバーのポート番号。

これらの指定に使われる値は、対応するキーワードに与えた値から取ってきます。

`:mailbox` メールを取得するメールボックスの名前。ディフォルトは‘`INBOX`’で、これは普通は入ってくるメールを受け取るメールボックスです。単一のメールボックスの代わりに、メールを取得する複数のメールボックスのリストでもよいです。

`:predicate`
取得する記事を見つけるために使われる述語。ディフォルトの、‘`UNSEEN UNDELETED`’ はおそらくたいていの人には最良の選択でしょうが、ときどきIMAP クライアントでメールボックスを覗いて、いくつかの記事に既読(または SEEN) の印を付けるなら、これを‘`1:*`’ に設定する必要があるかもしれません。そうすれば、メールボックスのすべての記事は印の如何に関わらず取得されます。述語の完全な一覧は、RFC2060 の 6.4.4 節を読んでください。

`:fetchflag`
サーバーで、取得した記事にフラグを付ける方法。ディフォルトの‘`\Deleted`’ はそれらに消去のフラグを付けますが、単に既読のフラグを付けるための‘`\Seen`’ が代案になるでしょう。これらは最もありそうな二つの選択ですが、他のフラグも RFC2060 の 2.3.2 節で定義されています。

`:dontexpunge`
`nil` でなかったら、記事を取得した後で、それらに消去の印が付いていても削除しません。

IMAP メールソースの例:

```
(imap :server "mail.mycorp.com"
      :stream kerberos4
      :fetchflag "\\Seen")
```

`group` 実際のメールソースとして`mail-source` グループパラメーターで設定されているものを使います。See Section 3.10 [Group Parameters], p. 23.

Common Keywords
共通キーワードはどんな型のメールソースにも使うことができます。

キーワード:

`:plugged` `nil` でなかったら、Gnus が unplugged (ネットワークから切り離されている状態) であってもメールを取得します。ディレクトリーをメールソースに使っているのならば、この例のように指定することができます:

```
(setq mail-sources
      '((directory :path "/home/pavel/.Spool/"
                   :suffix ""
                   :plugged t)))
```

こうしておくことによって unplugged であっても Gnus はメールを取得します。これはローカルのメールとニュースを使う場合に便利です。

7.4.4.2 関数インターフェース

上記のいくつかのキーワードは、実行するための Lisp 関数を指定します。関数が実行されている間だけ、それぞれのキーワード`:foo` に対して Lisp 変数`foo` が、そのキーワードの値に束縛されます。例えば、以下のメールソースの設定例について考えてみてください。

```
(setq mail-sources '((pop :user "jrl"
                          :server "pophost" :function fetchfunc)))
```

関数`fetchfunc` が実行されているとき、`user` というシンボルの値は`"jrl"` に束縛され、`server` というシンボルの値は`"pophost"` に束縛されます。`port`, `password`, `program`, `prescript`, `postscript`, `function` および`authentication` の値もまた(それらのディフォルト値に) 束縛されます。

それぞれの型のメールソースのためのキーワードのリストについては、前述の説明を参照してください。

7.4.4.3 メールソースのカスタマイズ

以下はメールの取得方法に影響する変数の一覧です。普通はこれらのどれも設定または変更する必要は無いでしょう。

`mail-source-crash-box`
: メールが、それを処理している期間中に格納されている場所です。ディフォルトは`~/.emacs-mail-crash-box` です。

`mail-source-delete-incoming`
: `nil` でなければ、入って来たメールを一時的に格納したファイルを、それを処理した後で消去します。`t` ではファイルをただちに消去し、`nil` ではいかなるファイルも消しません。正の数だった場合は、その日数以上に古いファイルを消去します(消去は新着メールを受け取るときだけ行なわれます)。`mail-source-delete-incoming` を`nil` にしておいて、`mail-source-delete-old-incoming` をフックで呼ぶか、または対話的に呼ぶこともできます。

`mail-source-delete-old-incoming-confirm`
: 非-`nil` だったら、古い incoming (メールの到着時に使われた) ファイルを消去するときに確認を求めます。この変数は`mail-source-delete-incoming` が正の数である場合だけ使われます。

`mail-source-directory`
: 入ってきたメールソースのファイルが(もしあれば) 格納されるディレクトリーです。ディフォルトは`~/Mail/` です。現時点でこれが使われる唯一のものは、変数`mail-source-delete-incoming` が`nil` または数値であった場合に、入ってきたメールを格納するファイルの置き場所の指定です。

`mail-source-incoming-file-prefix`
: 入ってきたメールを一時的に格納するファイルの名前の接頭語です。ディフォルトは`Incoming` で、この場合ファイルは`Incoming30630D_` や`Incoming298602ZD` のようになります。これが本当に関係するのは`mail-source-delete-incoming` が`nil` または数値の場合だけですが。

`mail-source-default-file-modes`
: すべての新しいメールファイルはこのファイルモードになります。ディフォルトは`#o600` です。

`mail-source-movemail-program`
: `nil` でなかったら、新着メールを取り込むためのプログラムの名前であると解釈されます。`nil` だったら`exec-directory` にある`movemail` が使われます。

7.4.4.4 メールの取得

新しいメールをどこから取得するかを実際に Gnus に指示する手段は、`mail-sources` をメールソース指示子のリストに設定することです(see Section 7.4.4.1 [Mail Source Specifiers], p. 166)。

この変数が`nil` であれば、メールバックエンドは決して自分自身ではメールを取得しようとしません。

ローカルのスプールとPOP メールサーバーの両方からメールを取得したいなら、このようにすることができます:

```
(setq mail-sources
      '((file)
        (pop :server "pop3.mail.server"
             :password "secret")))
```

あるいは、これらのキーワードのどんなディフォルトも使いたくなければ、このようにしてください:

```
(setq mail-sources
      '((file :path "/var/spool/mail/user-name")
        (pop :server "pop3.mail.server"
             :user "user-name"
             :port "pop3"
             :password "secret")))
```

あなたがメールバックエンドを使うとき、Gnus はすべてのメールを inbox から吸い上げてホームディレクトリーに放り込みます。あなたがメールバックエンドを使っていないときは、Gnus は一通もメールを移動しません---そういう場合には、最初にたくさんの魔法を唱えなければなりません。まず五芳星を描き、蝋燭を灯し、山羊を生け贄として捧げ終えた後には、Gnus があなたのメールを移動したとしても、あなたは実際にはあまり驚かないはずです。

7.4.5 メールバックエンド変数

これらの変数(のほとんど) は、すべてのさまざまなメールバックエンドに関連します。

`nnmail-read-incoming-hook`
: すべてのメールバックエンドは、新しいメールを読み込んだ後にこのフックを呼びます。そうしたければ、何かのメール監視プログラムに知らせるためにこのフックを使うことができます。

`nnmail-split-hook`
: それぞれのメッセージがそのヘッダーに基づいて分割がなされる直前に、それが格納されているバッファーで実行されるフックです。このフックは、それがふさわしいと考えられるようにするために、どんなやり方ででも自由にバッファーの内容を編集することができます---バッファーは分割が終わった後で捨てられ、バッファーで行なわれた変更はどのファイルにも現れません。`gnus-article-decode-rfc1522` は、このフックに加えられることがありそうな関数の一つです。

`nnmail-pre-get-new-mail-hook`
`nnmail-post-get-new-mail-hook`
: これらは、入ってくるメールを処理するときに実行される、有用な二つのフックです—`nnmail-pre-get-new-mail-hook` は新しいメールを処理する直前に呼ばれ、`nnmail-post-get-new-mail-hook` はメールの扱う処理が終わったときに呼ばれます。以下はこれらの二つのフックを使って、新しいメールのファイルのファイルモードを変更する例です:

```
(add-hook 'nnmail-pre-get-new-mail-hook
          (lambda () (set-default-file-modes #o700)))

(add-hook 'nnmail-post-get-new-mail-hook
          (lambda () (set-default-file-modes #o775)))
```

`nnmail-use-long-file-names`
: `nil` でないなら、メールバックエンドは長いファイル名とディレクトリー名を使います。'`mail.misc`' のようなグループは`mail.misc` という長い名前のディレクトリーかファイルに収められます(`nnml` バックエンドの場合はディレクトリー、`nnfolder` バックエンドの場合はファイルです)。`nil` だったら、同じグループは`mail/misc` に収められます。

`nnmail-delete-file-function`
: ファイルを消去するために呼ばれる関数です。ディフォルトは`delete-file` です。

`nnmail-cache-accepted-message-ids`
: `nil` でないと、バックエンドに(例えば`Gcc` によって) 入って来た記事の`Message-ID` を、メールの重複を発見するためのキャッシュに入れます。ディフォルトは`nil` です。

`nnmail-cache-ignore-groups`
: 正規表現か正規表現のリストです。名前がどれかの正規表現に合致するグループの記事は、`Message-ID` キャッシュに記録されません。

  例えば特級分割(see Section 7.4.6 [Fancy Mail Splitting], p. 175) を関数`nnmail-split-fancy-with-parent` とともに使っている場合に役立つでしょう。

7.4.6 特級メール分割

訳注: *Fancy* という単語は、創造的、空想的、気まぐれな、好きな、派手な、上等な、極上の、変わり種の、等々のさまざまな意味で使われますが、ここでは助動詞無しで使える利便を考えて「特級」という訳語を割り当てました。

比較的単純な、標準のメール分割指定の方法では思い通りにならないならば、`nnmail-split-methods` を`nnmail-split-fancy` に設定すると良いでしょう。そうすると、変数`nnmail-split-fancy` で遊ぶことができるようになります。

まずこの変数の値の例を見てみましょう:

```
;; メイラーデーモンから送られたメッセージが、普通のグループにクロス
;; ポストされないようにします。警告は本当のエラーとは違ったグループ
;; に入れます。
(| ("from" mail (| ("subject" "warn.*" "mail.warning")
                   "mail.misc"))
   ;; エラーでないメッセージはすべての関連するグループにクロスポスト
   ;; しますが、(ding) メーリングリストと他の (ding) 関連のメールの
   ;; ためのグループにはクロスポストしません。
   (& (| (any "ding@ifi\\.uio\\.no" "ding.list")
         ("subject" "ding" "ding.misc"))
      ;; 他のメーリングリスト...
      (any "procmail@informatik\\.rwth-aachen\\.de" "procmail.list")
      (any "SmartList@informatik\\.rwth-aachen\\.de" "SmartList.list")
      ;; 以下のどちらのメーリングリストも同じ接尾語なので、bugs- だ
      ;; けに投稿されたものが mypkg.list にクロスポストされないよう
      ;; にしています。しかし本当にクロスポストされた記事をクロスポ
      ;; ストすることはできるようになっています。
      (any "bugs-mypackage@somewhere" "mypkg.bugs")
      (any "mypackage@somewhere" - "bugs-mypackage" "mypkg.list")
      ;; 人々...
      (any "larsi@ifi\\.uio\\.no" "people.Lars_Magne_Ingebrigtsen"))
   ;; 合致しなかったメールはすべてを捕まえるグループへ行きます。
   "misc.misc")
```

この変数は「分割」の様式になっています。分割は(ことによると) それぞれの分割が他の分割を含む再帰的構造です。以下は使うことができる分割の構文です:

`group` 分割が文字列だったら、それはグループ名であるとみなされます。通常の正規表現の展開が行なわれます。後述の例(訳注: '`\\&`', '`\\1`'〜'`\\9`') を見てください。

(*field value* [- *restrict* [...]] *split* [*invert-partial*])

この分割は少なくとも三つの要素を含んでいる必要があります。最初の要素*field* (正規表現) に合致するヘッダーが*value* (これも正規表現) に合致する文字列を含んでいたならば、*split* で指定されたグループにメッセージを格納します。

field の後にあって、しかも合致した*value* の最後尾より前にある何かの文字列に*restrict* (これもまた正規表現) が合致したら、*split* は無視されます。いくつかの*restrict* のどれもが合致しなければ*split* が実行されます。

最後の要素*invert-partial* は任意です。これが省略されていなくて、しかも値が`nil` でなければ、語(word) の境界をまたいで正規表現の合致を行なうかどう

かの振る舞い(nnmail-split-fancy-match-partial-words 変数によって制御されます; 下記参照) が反転します。(Gnus 5.10.7 の新機能)

(| *split* ...)
分割がリストで、最初の要素が| (垂直棒) だったら、それぞれの*split* をそのうちの一つが合致するまで実行します。ここで言う「合致」とは、ある*split* がメッセージを一つ以上のグループに格納しようとすることです。

(& *split* ...)
分割がリストで、最初の要素が& だったら、そのリストにあるすべての*split* を実行します。

junk
もし分割がシンボルjunk だったら、そのメッセージを保存しません(すなわち、消去してしまいます)。非常に注意して使ってください。

(: *function arg1 arg2* ...)
もし分割がリストで、最初の要素が: だったら、二番目の要素が*args* を引数として関数として呼ばれます。関数は*split* を返さなければなりません。

例えば以下の関数は、記事のボディーに基づいた分割に使えるでしょう:

```
(defun split-on-body ()
  (save-excursion
    (save-restriction
      (widen)
      (goto-char (point-min))
      (when (re-search-forward "Some.*string" nil t)
        "string.group"))))
```

function が実行されるとき、バッファーは対象となるメッセージのヘッダー部分に狭められています。それが上記の例でsave-excursion とsave-restriction の後で(widen) を呼ぶ必要がある理由です。さらにnnimap バックエンドの場合、ディフォルトでは記事のボディーがダウンロードされないことに注意してください。それをするためにはnnimap-split-download-body をt に設定する必要があります(see Section 7.3.3 [Client-Side IMAP Splitting], p. 161)。

(! *func split*)
分割がリストで最初の要素が! だったら、*split* が実行され、*func* は*split* の結果を引数として呼ばれます。*func* は分割を返さなければなりません。

nil
分割がnil だったら、それは無視されます。

これらの分割で*fileld* は完全なフィールド名(と言うかヘッダー名) に合致しなければなりません。

通常これらの分割における*value* は、基礎モード(fundamental mode) 構文テーブル(syntax table) に従って、完全に語 (word) に合致しなければなりません。言い換えれば、すべての*value* は暗に\<...\> 印(語の区切り記号) で囲まれます。したがって、例えば以下の分割を使うと、

```
(any "joe" "joemail")
```

'joedavis@foo.org' からやって来たメッセージは、通常'joemail' には格納されないでしょう。この振る舞いを変更したければ、以下の三つのやり方のどれでもを使うことができます:

1. nnmail-split-fancy-match-partial-words 変数をnil ではない値に設定することによって、語の境界を無視させることができます。すると、合致はより grep ふうになり

ます。この変数は、特級分割で語の境界をまたいだ合致を行なうかどうかを制御します。ディフォルト値は`nil` です。

分割の規則のすべての*value* に影響することに注意してください。

2. `.*` で始まる*value* は、語の前にある語の境界を無視させます。同様に`.*` で終わる*value* は、語の後ろにある語の境界を無視させます。例えば`"@example\\.com"` という*value* は‘`foo@example.com`’ に合致しませんが、`".*@example\\.com"` ならば合致します。
3. この章の最初の方で述べた*invert-partial* フラグを、‘`(`*`field value ...`*`)`’ 型の分割規則で使うことができます。このフラグが設定されていると、`nnmail-split-fancy-match-partial-words` が`nil` であっても、語の両側にある語の境界は無視されます。逆に、このフラグが設定されていると、`nnmail-split-fancy-match-partial-words` が`nil` ではない値であっても、語の境界は無視されません。(Gnus 5.10.7 の新機能)

field と*value* は Lisp シンボルであることもできます。その場合それらは`nnmail-split-abbrev-alist` で指定された内容に従って展開されます。これはセルのCAR がキーを含んでいて、CDR が関連付けられた値を持っているコンスセル(cons cell) の連想リストです。以下の項目が、あらかじめ`nnmail-split-abbrev-alist` に定義されています:

`from` ‘`From`’、‘`Sender`’ および‘`Resent-From`’ の各フィールドに合致します。

`to` ‘`To`’、‘`Cc`’、‘`Apparently-To`’、‘`Resent-To`’ および‘`Resent-Cc`’ の各フィールドに合致します。

`any` `from` と`to` を統合したものです。

`list` ‘`List-ID`’、‘`List-Post`’、‘`X-Mailing-List`’、‘`X-BeenThere`’ および‘`X-Loop`’ フィールドに合致します。

`nnmail-split-fancy-syntax-table` は、これらのすべての分割が実行されているときに有効な構文テーブルです。

ヘッダーのいくつかの情報に基づいて、Gnus に動的にグループを作らせたい(例えば、グループ名の一部を`replace-match` のようなやり方で置き換えさせたい) ならば、次のようなことができます。

```
(any "debian-\\b\\(\\w+\\)@lists.debian.org" "mail.debian.\\1")
```

この例では、‘`debian-foo@lists.debian.org`’ に送られたメールは‘`mail.debian.foo`’ というグループに入れられます。

文字列が要素‘`\\&`’ を含んでいる場合は、直前に合致した文字列で置き換えられます。同様に、要素‘`\\1`’ から‘`\\9`’ までは、合致した文字列の一部で置き換えられます(訳注: 正規表現の中に‘`\\(`’ と‘`\\)`’ を使ってグループにまとめられたものが一つ以上ある場合に、‘`\\`*`n`*’ はその正規表現の*n* 個目のグループに合致する文字列の一部で置き換えられます)。

その際、合致した文字列を小文字にしたもので置き換えるべきかどうかを`nnmail-split-lowercase-expanded` が決定します。これを非-`nil` にすることによって、アドレスで大文字と小文字が区別せずに使われている(例えば mailing-list@domain と Mailing-List@Domain) 場合でも、複数のグループが生成されてしまうことを避けることができます。ディフォルトは`t` です。

関数`nnmail-split-fancy-with-parent` は、フォローアップ記事を親記事と同じグループに振り分けるために使います。メールの振り分けを一生懸命設定してみても完璧にはできないことがありますね。例えば、上司から個人宛てのメールが届いたとします。自分が携っているプロジェクトとは別の話です。けれど「他のメールと区別できるようにこれこ

れこういう言葉を表題に書いてください」と上司に向かって指図するわけにはいきませんから、結局自分の手を煩わしてひとつひとつメールを正しいグループに振り分けるはめになります。そんなときにこの関数を使うと、この面倒な作業を一スレッドにつき一回きりで済ますことができます。

この機能を利用するためには、まず変数nnmail-treat-duplicates およびnnmail-cache-accepted-message-ids の値をnil ではない値に設定する必要があります。それができたらnnmail-split-fancy-with-parent を使ってみてください。コロンを使ってこんな風に書きます:

```
(setq nnmail-treat-duplicates 'warn     ; または delete
      nnmail-cache-accepted-message-ids t
      nnmail-split-fancy
      '(| (: nnmail-split-fancy-with-parent)
          ;; 残りの振り分け方はここに書く
        ))
```

この機能は実際、次の様に働いています: 変数nnmail-treat-duplicates の値が非-nil の場合、Gnus は見つけた全記事のメッセージ ID を変数nnmail-message-id-cache-file で指定されたファイルに記録します。このとき、それぞれの記事が格納されたグループの名前を併記します(ただしメールではないメッセージの場合は、グループ名は省略されます)。さて、いよいよメールの振り分けが始まると、関数nnmail-split-fancy-with-parent は、分割される各記事の References (と In-Reply-To) ヘッダーを調べ、nnmail-message-id-cache-file で指定されたファイルにそれらのメッセージ ID があるかどうか調べます。親記事が見つかると、そのグループ名が正規表現nnmail-split-fancy-with-parent-ignore-groups に合致しなければ、この関数は対応するグループ名を返すわけです。ここで、変数nnmail-message-id-cache-length の値をディフォルトよりも幾らか大きな値に設定することを勧めます。そうすると、今調べられたメッセージ ID たちは今しばらくキャッシュの中に存続できます(5000 に設定するとキャッシュファイルの大きさはだいたい 300 キロバイトぐらいになるみたいです)。さらに、変数nnmail-cache-accepted-message-ids の値を非-nil に設定すれば、Gnus は移動された記事のメッセージ ID をも記録するので、フォローアップ記事は親記事の移動先と同じグループに入るようになります。

特定のグループをキャッシュに記録したくない場合は、変数nnmail-cache-ignore-groups も参照してください。例えば、外に出すすべてのメッセージを“outgoing”グループに保存しているのならば、nnmail-cache-ignore-groups をそのグループ名に合致するように設定すれば良いでしょう。さもないとあなたのすべてのメッセージに対する返事が“outgoing”グループに入ってしまいます。

nnmail-debug-splitting をnil 以外の値にしておくと、実施したすべてのメール分割の判定結果を‘*nnmail split*’ バッファーに記録します。

7.4.7 グループメール分割

何ダースものメーリングリストを購読しているけれど、手でメール分割規則を維持したくないというときのために、グループメール分割というものがあります。あなたがしなければならないことは、グループパラメーターかグループカスタマイズでto-list, to-address の両方もしくはどちらかを設定してnnmail-split-methods をgnus-group-split に設定するだけです。分割関数はすべてのグループでこれらのパラメーターを走査し、それに従って分割します。すなわち、メールグループのパラメーターto-list かto-address で指定されたアドレスから投稿されたものか、そのアドレスへ投稿されたメッセージがそのグループに保存されます。

ときには、メーリングリストには複数のアドレスがあり、メール分割にそれらすべてを認識させる必要があるかもしれません: extra-aliases グループパラメーターを追加のアドレスのリストに設定するだけで終りです。あえて正規表現を使いたければ、split-regexp を設定してください。

これらのすべてのグループのパラメーターは、nnmail-split-fancy の分割を作成するために使用されます。その分割の仕様の中身は、*field* の値が'any' であり、*value* の値がto-list とto-address とextra-aliases のすべてとsplit-regexp に合致するもののすべてに合致する単一の正規表現、そして*split* がグループの名前になります。*restrict* も使うことができます: それにはsplit-exclude パラメーターを正規表現のリストに設定してください。

これらのすべてのパラメーターを使って正しい分割が生成されないときや、何かもっと凝ったものが必要なときは、split-spec パラメーターをnnmail-split-fancy の分割に設定することができます。この場合は、前もって書かれたすべてのパラメーターはgnus-group-split に無視されます。特に、split-spec はnil (訳注: これもnnmail-split-fancy の分割の一種です) に設定することができ、その場合そのグループはgnus-group-split に無視されます。

それぞれのグループのために、一つの分割を含む単一の& 特級分割を定義することによって、gnus-group-split は合致するすべてのグループにクロスポストをします。どの分割にも合致しないメッセージは、どれかのグループで*split-spec* がcatch-all に設定されていない場合はgnus-group-split-default-catch-all-group で指定された名前のグループに格納されます。その場合、そのグループはすべてを受け取る(catch-all) グループとして使われます。この変数はしばしばグループを指定するためだけに使われますが、任意の複雑な特級分割に設定することもできるので(結局のところグループ名は特級分割なのです)、個人のメールフォルダーにそれらのメールが格納されるどのメーリングリストにも当てはまらないメールを、分割するのに便利でしょう。なおこの特級分割は、| 分割リスト(それは、グループパラメーターから抽出された規則を持った& 分割をも含んでいます) の最後の要素として追加されることに注意してください。

そろそろ例を出すべきでしょう。以下のグループパラメーターが定義されていることを仮定します:

```
nnml:mail.bar:
((to-address . "bar@femail.com")
 (split-regexp . ".*@femail\\.com"))
nnml:mail.foo:
((to-list . "foo@nowhere.gov")
 (extra-aliases "foo@localhost" "foo-redist@home")
 (split-exclude "bugs-foo" "rambling-foo")
 (admin-address . "foo-request@nowhere.gov"))
nnml:mail.others:
((split-spec . catch-all))
```

nnmail-split-methods をgnus-group-split に設定すると、nnmail-split-fancy が選択されていて、かつ変数nnmail-split-fancy が以下のように設定されているかのように振舞います:

```
(| (& (any "\\(bar@femail\\.com\\|.*@femail\\.com\\)" "mail.bar")
      (any "\\(foo@nowhere\\.gov\\|foo@localhost\\|foo-redist@home\\)"
           - "bugs-foo" - "rambling-foo" "mail.foo"))
```

```
"mail.others")
```

グループ分割をすべてのメールグループで積極的には使いたくなければ、nnmail-split-fancy の分割を次のように使用することで、いくつかのグループだけで使うことができます。

```
(: gnus-group-split-fancy groups no-crosspost catch-all)
```

groups は、出力の分割を生成するためにパラメーターが走査されるグループ名のリストか、それらのグループ名に合致する正規表現です。*no-crosspost* はクロスポストを禁止するために使うことができ、その場合は、単一の| 分割が出力されます。*catch-all* はgnus-group-split-default-catch-all-group のように、最後の手段として使われる特級分割です。*catch-all* がnil に設定されているか、split-regexp がどれかの選択されたグループで空の文字列に合致すると、すべてを受け取る(catch-all) 分割は発行されません。そうでない場合、あるグループにcatch-all に設定されているsplit-spec があると、そのグループは*catch-all* 引数の値よりも優先されます。

不運なことに、すべてのグループとそれらのパラメーターを走査することは、特にすべてのメッセージに対して行なわなければならないことを考慮に入れると、非常に遅くなるでしょう。でも、絶望してはいけません。gnus-group-split-setup 関数を、はるかに効率的な方法でgnus-group-split を動作させるために使うことができます。それはnnmail-split-methods をnnmail-split-fancy に設定し、nnmail-split-fancy をgnus-group-split-fancy で生成される分割に設定します。そうすることによって、どんなに分割するメッセージがたくさんあっても、グループパラメーターは一度だけ走査されるようになります。

しかしながら、グループパラメーターを変更すると、nnmail-split-fancy を手で更新しなければならなくなるでしょう。gnus-group-split-update を実行することによって、それを行なうことができます。どちらかと言えば、それを自動的に更新したい場合には、gnus-group-split-setup にそれを実行するように指示してください。例えば、~/.gnus.el に以下のものを追加すれば良いでしょう:

```
(gnus-group-split-setup auto-update catch-all)
```

auto-update がnil でなければgnus-group-split-update がgnus-get-top-new-news-hook に追加されるので、二度とnnmail-split-fancy の更新について心配する必要はありません。*catch-all* を省略しない場合は(それはオプションでnil と等価です)、gnus-group-split-default-catch-all-group がその値に設定されます。

gnus-group-split-update によって設定されたnnmail-split-fancy を後で変更する必要があるときのために、この関数(gnus-group-split-update) は終了する直前にgnus-group-split-update-hook を実行します。

7.4.8 古いメールを取り込む

たいていの人は色々なファイルフォーマットで保存されたたくさんの古いメールを持っているでしょう。Gnus の粋なメールバックエンドの一つを使うように設定したのであれば、おそらく古いメールをメールグループに取り込みたいと思いますよね。

それをすることはとても簡単です。

例を挙げましょう: nnml (see Section 7.4.13.3 [Mail Spool], p. 188) を使ってメールを読んでいて、nnmail-split-methods を申し分の無い値に設定しているものとしましょう。重要な、しかし古いメールで、古い Unix mbox ファイルが満たされています。あなたはそれをnnml グループに移動したいと思っています。

方法です:

1. グループバッファーに行ってください。
2. `G f` をタイプしてください。`nndoc` グループを作成するための元になる mbox ファイルの名前を求められるので、それを入力してください(see Section 3.9 [Foreign Groups], p. 21)。
3. `SPC` をタイプして、新しく作られたグループに入ってください。
4. `M P b` をタイプして、グループバッファーのすべての記事に実行印を付けてください(see Section 4.7.6 [Setting Process Marks], p. 66)。
5. `B r` をタイプしてプロセス印の付いたすべての記事を再スプールしてください。その際に入力を求められるので、'`nnml`' と答えてください(see Section 4.26 [Mail Group Commands], p. 114)。

今や mbox ファイルのすべてのメールメッセージは、あなたの`nnml` グループ群にもばらまかれています。それらに入って、ものごとが変な故障も無く、うまくいったかどうかを調べてください。大丈夫なようであれば、mbox ファイルを消そうと思うかもしれませんが、私はすべてのメールがあるべきところに納まったことを完全に確認するまでは、そうはしません。

再スプールすることは、あるメールバックエンドを別のものに変更するときにも便利なものです。古いメールグループにあるメールは、新しいメールバックエンドを使ってただ再スプールすれば良いのです。

7.4.9 メールの期限切れ消去

伝統的なメールリーダーは、既読の印を付けるとメールの記事を何らかの方法で削除する傾向があります。Gnus はメールを読むことに対して、基本的に違う方法を取ります。

基本的に Gnus は、メールを少々変わった方法で受け取られたニュースであるとみなします。実際にメールを変更したり、メールメッセージを消す権限があるとは考えません。あなたがメールグループに入って記事に「既読」の印を付けたり、何らかの他のやり方で切ったりしても、メールの記事はまだシステムに存在しています。繰り返します: Gnus はあなたの古い既読のメールを消去しません。もちろん、あなたがそうしろと要求しない限りの話ですが。

要らないメールを Gnus に削除させるには、記事に「期限切れ消去可能」(expirable) の印を付けなければなりません。(ディフォルトのキー割り当てでは、`E` をタイプしなければならないということです。) しかしながら、これは記事が即座に消え去るということではありません。一般的にメール記事は、1) 期限切れ消去可能の印が付いていて、かつ 2) 一週間以上経っている、という場合に、システムによって削除されます。記事を期限切れ消去可能にしなければ、それは地獄が凍りつくまでシステムに残り続けます。このことは、もう一度強調付きで繰り返されるに足るものです:「もし」あなたが記事を「期限切れ消去可能」に「しない」なら、Gnus は「決して」それらの「記事」を消去しません。

手作業で記事に期限切れ消去可能の印を付けなければならないわけではありません。Gnus は“auto-expire”および“total-expire”と呼ばれる二つの機能を提供して、あなたの手助けをします。かいつまんで言えば“auto-expire”はあなたが記事を選択したときに Gnus が`E` を叩いてくれることを意味します。そして“total-expire”は、すべての既読の記事は期限切れ消去可能であると Gnus が解釈することを意味します。したがって'`E`' の印が付けられた記事に加えて、'`r`', '`R`', '`O`', '`K`', '`Y`' (およびその類) の印が付けられた記事も期限切れ消去可能であると解釈されます。これらの印の完全なリストは`gnus-auto-expirable-marks` にあります。

では auto-expire または total-expire は、いつ使用されるべきなのでしょうか? メーリングリストを購読しているほとんどの人々は、それぞれのリストがそれ用のグループに分割されるようにして、それらのグループに対して auto-expire または total-expire を有効にしています。(それぞれのリストをそれ用のグループへの分割する件についてのさらなる情報はSection 7.4.3 [Splitting Mail], p. 164, を参照してください。)

auto-expire と total-expire のどちらが良いのでしょうか? それに答えるのは簡単ではありません。概して言えば、たぶん auto-expire が速いでしょう。auto-expire の別の利点は、より多くの印を後で読み返すつもりの記事に使うことができる、つまり今までどおりに可視(tick)、保留(dormant) または既読(read) の中から選ぶことができるということです。しかし total-expire では、dormant と ticked からしか選べません。total-expire の利点は、適応スコア付け(see Section 8.6 [Adaptive Scoring], p. 244) で良好に働くことです。auto-expire は通常のスコア付けでは動作しますが、適応スコア付けではだめです。

正規表現`gnus-auto-expirable-newsgroups` に合致するグループでは、読んだすべての記事に自動的に期限切れ消去可能の印が付けられます。期限切れ消去可能の印の付いたすべての記事は、概略バッファーの最初の桁に'`E`' が表示されます。

自動期限切れ消去を有効にすると、ディフォルトではあなたが読んだすべての記事に、以前に読まれたかどうかに関わらず、Gnus は期限切れ消去可能の印を付けます。既読の印が付いている記事に、自動的に期限切れ消去可能の印が付けられるのを避けるには、以下のようなものを`~/.gnus.el` ファイルに置いておけば良いでしょう:

```
(remove-hook 'gnus-mark-article-hook
             'gnus-summary-mark-read-and-unread-as-read)
(add-hook 'gnus-mark-article-hook 'gnus-summary-mark-unread-as-read)
```

グループを自動期限切れ消去可能にしても、すべての既読の記事が期限切れ消去されるわけではなく、期限切れ消去可能の印が付いている記事だけが期限切れ消去されることに気を付けてください。また、*d* 命令が自動的に記事を期限切れ消去可能にするのでは無いことにも気を付けてください---自動期限切れ消去可能にしたグループでは、記事に既読の印が半自動で付けられることによってのみ、記事が期限切れ消去可能になるということです。

2〜3 のメーリングリストを購読していて、読み終わってしばらく経ったら記事が消えてしまうようにしたいなら、例えばこんな風に設定しましょう:

```
(setq gnus-auto-expirable-newsgroups
      "mail.nosense-list\\|mail.nice-list")
```

自動期限切れ消去を行なわせるもう一つの方法は、そのグループのグループパラメーターに`auto-expire` という要素を持たせることです。

もし適応スコア付け(see Section 8.6 [Adaptive Scoring], p. 244) と自動期限切れ消去を使用していると、問題が起こるでしょう。自動期限切れ消去と適応スコア付けはあまり良く調和しません。

変数`nnmail-expiry-wait` で、期限切れ消去可能な記事をどれくらいの期間残しておくかのディフォルトの時間を設定します。この変数の値は日数(整数である必要はない) か、もしくは期限切れ消去しないことを意味する`never` か即時消去を意味する`immediate` のうち、どちらか一つのシンボルの、いずれかを指定できます。Gnus はメッセージが送り出されたときではなく、それが到着 してからの日数を計算します。ディフォルトは 7 日間です。

`nnmail-expiry-wait-function` 変数を使うと、記事が存在するグループに基づいて記事の存続期間を微調整することできます。関数に設定すると、その戻り値が`nil` でなければ`nnmail-expiry-wait` の値を上書きします。そうでなければ代わりに`nnmail-expiry-wait` の値が使われます。

例えば‘mail.private’ グループは一ヶ月、‘mail.junk’ グループは一日、その他全部は六日間に期限を設定するのならば、以下のやり方で可能です:

```
(setq nnmail-expiry-wait-function
      (lambda (group)
        (cond ((string= group "mail.private")
               31)
              ((string= group "mail.junk")
               1)
              ((string= group "important")
               'never)
              (t
               6))))
```

この関数に与えられるグループ名には「装飾」すなわち‘nnml:’ のようなものは付きません。

変数nnmail-expiry-wait またはnnmail-expiry-wait-function の代わりに、expiry-wait グループパラメータを使って有効期間を選択的に変更することもできます(see Section 3.10 [Group Parameters], p. 23)。

記事を期限切れ消去するときに取られる通常の動作は、それらを消去することです。しかし、場合によってはそれらを消去するよりも別のグループに移動した方が有意義かもしれません。変数nnmail-expiry-target (とグループパラメーターexpiry-target) はこれを制御します。この変数の値はすべてのグループに対するディフォルトになりますが、特定のグループごとにグループパラメーターを使って指定すれば、そちらを優先させることができます。ディフォルトの値はdelete ですが、文字列(記事を移動する先のグループ名) または移動先のグループ名かdelete を返す関数にすることができます(関数の場合は、記事に範囲を狭めたバッファーで、その記事が存在しているグループ名が引数として与えられます)。

グループ名を指定する場合の例:

```
(setq nnmail-expiry-target "nnml:expired")
```

Gnus には期限切れのメールをグループに移動させるための関数があります。それは変数nnmail-fancy-expiry-targets に従って動作します。例です:

```
(setq nnmail-expiry-target 'nnmail-fancy-expiry-target
      nnmail-fancy-expiry-targets
      '((to-from "boss" "nnfolder:Work")
        ("subject" "IMPORTANT" "nnfolder:IMPORTANT.%Y.%b")
        ("from" ".*" "nnfolder:Archive-%Y")))
```

この設定を行なうことにより、表題ヘッダーにIMPORTANT を持っていて、YYYY 年MMM 月に発信されたいかなるメールも、期限になるとnnfolder:IMPORTANT.YYYY.MMM グループに移動させられます。また、From または To ヘッダーが文字列boss を含んでいるメールはnnfolder:Work に、それ以外のすべてのメールはnnfolder:Archive-YYYY に、それぞれ期限になると移動させられます。

nnmail-keep-last-article がnil でないと、Gnus はメールグループの最後の記事を決して期限切れ消去しません。これは procmail の利用者の人生をより楽にするためのものです。

補足: 上記の、Gnus が決して期限切れ消去可能でない記事を期限切れ消去することはない、というのは嘘です。total-expire をグループパラメーターに入れても、記事に期限切れ消去

の印が付くことはありませんが、読んだすべての記事は期限切れ消去の処理に通されます。非常に注意して使ってください。さらに危険なのは変数gnus-total-expirable-newsgroupsです。この正規表現に合致するすべてのグループでは、読んだすべての記事が期限切れ消去の処理に通されます。これは、当のグループのすべて の古いメールの記事は、しばらく後で削除されるということです。非常に注意して使ってください。そして、あなたが使った正規表現が間違ったグループに合致してしまい、すべての重要なメールが消えてしまったと言って、私に泣き付いて来ないでください。しっかしりなさい! (直訳: 男になりなさい、あるいは女になりなさい、さもなければもっと気持ちいい何にでもなりなさい!) ほうら、言わんこっちゃない!

たいていの人はほとんどのメールグループを total-expirable (全体期限切れ消去可能) にしますが。

gnus-inhibit-user-auto-expireがnil でなければ、グループで自動期限切れ消去が有効になっていても、利用者が印を付ける命令が記事に期限切れ消去可能の印を付けることはありません。

記事の期限切れ消去可能の印は、自動期限切れ消去が有効になっていないグループにコピーするか移動するとき削除されます。これは記事が不意に期限切れ消去されてしまうことを防ぐためです。一方、自動期限切れ消去が有効になっているグループにコピーまたは移動される記事の期限切れ消去可能の印は、ディフォルトでは変化しません。つまり、そのようなグループにコピーまたは移動されるとき、期限切れ消去可能だった記事は期限切れ消去可能のままにされ、期限切れ消去可能ではなかった記事に期限切れ消去可能の印が付くことはありません。したがって、たとえ自動期限切れ消去のグループであっても、いくつかの記事は期限切れ消去されないでしょう(それらを再び読まない限りは)。自動期限切れ消去のグループに期限切れ消去しない記事が紛れ込んでしまうかもしれないその動作が気に入らないなら、gnus-mark-copied-or-moved-articles-as-expirable をnil ではない値に設定することができます。その場合、読み終わった記事は自動期限切れ消去が有効になっているグループにコピーまたは移動するとき、期限切れ消去可能の印が自動的に付けられます。ディフォルト値はnil です。

7.4.10 メール洗濯

メイラーやメーリングリストのサーバーは、メールに対して本当に本当に馬鹿げたことをすることで悪名高いです。「わぁ、RFC 822 はサーバーを通っていくメッセージのすべての行の最後にwE aRe ElItE!!!!!1!! を加えることを明示的に禁止はしていないぞ。さぁ、やってみよう!!!!1!」えぇ、そのとおりですが、RFC 822 とその後継はおろか者が読むようには書かれていません。当たり前なこと(訳注: 良識から逸脱すること) はそこでは議論されていません。ですから、この章が必要なのです。

適例: ドイツ語版の Microsoft Exchange は返答の表題に‘Re: ’の代わりに‘AW: ’を付け加えます。私はこれに動揺して狼狽しているふりをすることもできましたが、そうする気力がありませんでした。それは笑うべきことです。

Gnus は表示する記事を洗濯するために多すぎるほどの関数を提供していますが、メールをディスクに保存する前にふるいにかけることができた方が良いかもしれません。その目的のために、三つのフックとそれらのフックに入れることができる色々な関数を用意しています。

`nnmail-prepare-incoming-hook`
: このフックはメールに何かをする前に呼ばれ、総括的に掃除してきれいにする所作のためにあります。それは新しいすべての入ってきたメールを含んでいるバッファーで呼ばれます。使うことのできる関数は:

  `nnheader-ms-strip-cr`
  : それぞれの行から、最後にあるキャリッジリターン(carriage return)を取り除きます。これは MS のマシン上で動作している Emacs のディフォルトです。

`nnmail-prepare-incoming-header-hook`
: このフックはそれぞれのメールのヘッダーに範囲を狭められて呼ばれます。ヘッダーをきれいにするときに使うことができます。使うことのできる関数は:

  `nnmail-remove-leading-whitespace`
  : 「役に立つ」メーリングリストのサーバーが、見栄えを良くするためだと称して、ヘッダーの前の方に付け加えた空白を無くします(訳注: 例えば‘`Subject:`’ などの直後に二つ以上の空白文字があったら、一つを残して消します)。まったくもう。

    (この関数はすべてのメッセージのヘッダーとボディー(の中にあるにあるヘッダー行のようなもの) に対しても動作するので、使用に際しては潜在的な危険を孕んでいます。したがってバグを修正するよりは、そういう特徴があることを文書で説明するのが、もちろん正しい解決の道です。)

  `nnmail-remove-list-identifiers`
  : いくつかのメーリングリストのサーバーは、そのリストが配信したメールであることを同定するための識別子---例えば‘`(idm)`’---をすべての`Subject` ヘッダーの先頭に付け加えます。石器時代のメールリーダーを使っている人たちには、それは確かに良いことです。この関数は正規表現`nnmail-list-identifiers` に合致する文字列を取り除きます。それは正規表現のリストでも構いません。ただし正規表現に`\\(..\\)` を含めてはいけません。

    例えば‘`(idm)`’ と‘`nagnagnag`’ という識別子を取り除きたいのなら:

    ```
    (setq nnmail-list-identifiers
          '("(idm)" "nagnagnag"))
    ```

    これは`gnus-list-identifiers` で非破壊的に行なうこともできます。See Section 4.18.3 [Article Hiding], p. 92.

  `nnmail-remove-tabs`
  : すべての‘`TAB`’ 文字を‘`SPC`’ 文字に変換します。

  `nnmail-ignore-broken-references`
  : いくつかの MUA (例えば Eudora と Pegasus) は壊れた`References` ヘッダーを作成しますが、`In-Reply-To` ヘッダーにはちゃんとしたものを入れます。この関数は、ヘッダー部に正規表現`nnmail-broken-references-mailers` に合致する行があったら、`References` ヘッダーを取り除きます。

`nnmail-prepare-incoming-message-hook`
このフックはそれぞれのメッセージに範囲を狭められて呼ばれます(訳注: 一度に複数のメールを受信した場合でも、一通ずつ呼ばれるということです)。使うことのできる関数は:

`article-de-quoted-unreadable`
Quoted Readable エンコードをデコードします(訳注: 実際に行なうのは quoted printable のデコードです)。

7.4.11 重複

いくつかのメーリングリストのメンバーなら、時々同じメールを二つ受け取ることがあるでしょう。これはとても煩わしいので、`nnmail` はそれが見つけたどんな重複をも、調べて処理します。これをするために、`nnmail` は古い`Message-ID` を`nnmail-message-id-cache-file` (ディフォルトでは`~/.nnmail-cache`) に保存します。それに保存される`Message-ID` のおおよその最大数は変数`nnmail-message-id-cache-length` で制御され、ディフォルトは 1000 です。(ですから千個の`Message-ID` が溜められます。) これで怖気をふるったなら、`nnmail-treat-duplicates` を`warn` (ディフォルトではそのようになっていますが) に設定しても良いでしょう。そうすると、`nnmail` は重複したメールを消去しない代わりに、それが別のメッセージの重複であるという警告をメールのヘッダーに挿入します。

この変数は関数であることもできます。その場合、関数は当のメッセージに範囲を狭められたバッファーから`Message-ID` を引数として呼ばれます。この関数は`nil`, `warn`, `delete` のどれかを返さなければなりません。

変数を`nil` に設定することによって、この機能を完全に使わないようにすることができます。

もしすべての重複したメールを特別な*duplicates* グループに入れたいのであれば、普通のメール分割方法を使ってそれをすることができます:

```
(setq nnmail-split-fancy
      '(| ;; 重複したメッセージは分かれたグループへ。
        ("gnus-warning" "duplicat\\(e\\|ion\\) of message" "duplicate")
        ;; デーモンやポストマスターなどからのメッセージは他へ。
        (any mail "mail.misc")
        ;; 他の規則。
        [ ... ] ))
```

もしくは次のようなもの:

```
(setq nnmail-split-methods
      '(("duplicates" "^Gnus-Warning:.*duplicate")
        ;; 他の規則。
        [...]))
```

すてきな使い方があるよ: 受け手である彼女がメールを Gnus で読んでいることと、彼女が`nnmail-treat-duplicates` を`delete` に設定してあることを知っていれば、彼女がすでに受け取ったことがわかっているメールの`Message-ID` そのものを使って、考えられる限りたくさんの侮辱を送ることができるんだぜ。その面白さを考えてみてよ! 彼女はそれらを決して見ることはないんだ! わぉ!

7.4.12 メールを読むのではない

あなたが使い始めたどんなメールバックエンドでも、あなたがそれらでメールを読みたいと思っていると仮定するという、悩ましい癖を持っていることに気が付くでしょう。これは決して不合理ではないかもしれませんが、あなたの望むことではないかもしれません。

`mail-sources` を`nil` に設定すれば、どのバックエンドも入ってくるメールを読もうとしなくなるので、役に立つはずです。

でも、それは行き過ぎでしょう。あなたが、例えば`nnml` でメールを読むことと、しまいこんである古い(Emacs 23 より前の) Rmail ファイルを`nnbabyl` を使ってざっと覗くことだけで、まったく満足していているのならば。すべてのバックエンドにはバックエンド`-get-new-mail` という変数があります。もし`nnbabyl` がメールを読み込みをやめさせたいのであれば、そのグループの仮想サーバー編集して、`nnbabyl-get-new-mail` を`nil` に設定しましょう。

すべてのメールバックエンドは、入ってくるメールを読み込むときに、保存されるべき記事に範囲を狭めて`nn*-prepare-save-mail-hook` を呼びます。

7.4.13 メールバックエンドを選ぶ

メールグループを動作するようにすると Gnus はメールスプールを読み込みます。メールのファイルはまずあなたのホームディレクトリーに複写されます。その後で何が起こるかは、メールをどの様式で格納したいかによります。

標準の Gnus では六つの違ったメールバックエンドがあり、さらに多くのバックエンドを個別に手に入れることができます。ほとんどの人が使うメールバックエンドは(それがたぶん最速なので) `nnml` です(see Section 7.4.13.3 [Mail Spool], p. 188)。

7.4.13.1 Unix メールボックス

nnmbox バックエンドはメールを格納するために標準の Un*x mbox ファイルを用います。`nnmbox` はそれぞれのメール記事にそれがどのグループに属しているかを示す追加のヘッダーを加えます。

仮想サーバーの設定:

`nnmbox-mbox-file`
: 利用者のホームディレクトリーのメールボックスの名前。ディフォルトは`~/mbox` です。

`nnmbox-activate-file`
: メールボックスのアクティブファイルの名前。ディフォルトは`~/.mbox-active` です。

`nnmbox-get-new-mail`
: `nil` でなければ、`nnmbox` は入って来たメールを読み込んでグループに分割します。ディフォルトは`t` です。

7.4.13.2 Babyl

nnbabyl バックエンドはメールを格納するために Babyl メールボックスを使います。`nnbabyl` はそれぞれの記事にそれがどのグループに属しているかを示す追加のヘッダーを加えます。

仮想サーバーの設定:

`nnbabyl-mbox-file`
: Babyl ファイルの名前。ディフォルトは`~/RMAIL` です。

`nnbabyl-active-file`
: Babyl ファイルのためのアクティブファイルの名前。ディフォルトは`~/.rmail-active` です。

`nnbabyl-get-new-mail`
: `nil` でなければ、`nnbabyl` は入ってくるメールを読み込みます。ディフォルトは`t` です。

7.4.13.3 メールスプール

nnml スプールメール様式は他の知られている様式とは互換性がありません。それは少し注意して使われるべきです。

このバックエンドを使うと、Gnus は入ってくるメールを、それぞれのメールを 1 ファイルとしてファイルに分割し、記事を変数`nnml-directory` で指定されたディレクトリーの下の対応するディレクトリーに入れます。ディフォルトの値は`~/Mail/` です。

前もってディレクトリーを作っておく必要はありません。その面倒は Gnus がすべて見てくれます。

あなたのアカウントに保存できるファイルの数に厳密な制限があるなら、このバックエンドを使うべきではありません。それぞれのメールはそれ自身のファイルを伴うので、数週間で数千の i ノードを占有する可能性は十分にあります。あなたにとってこれが問題でなく、親切なシステム管理者が気が狂ったように「誰が僕の i ノードを食いつぶしているんだ? 誰だ? 誰!?!」と叫びながら歩き回ることも問題でないなら、これがおそらく使うことのできる一番速い様式であるということは知っておくべきでしょう。新しいメールを読むためだけに大きな mbox ファイルを重い足取りで探す必要はありません。

`nnml` は記事分割に関してはおそらく一番遅いバックエンドでしょう。多くのファイルを作らなければならず、入ってくるメールのためのNOV データベースも作成しなければなりません。これのために、メールを読むことに関してはたぶん最速のバックエンドになるのです。

仮想サーバーの設定:

`nnml-directory`
: すべての`nnml` ディレクトリーはこのディレクトリーの下に置かれます。ディフォルトは`message-directory` の値(そのディフォルト値は`~/Mail`) です。

`nnml-active-file`
: `nnml` サーバーのためのアクティブファイル。ディフォルトは`~/Mail/active` です。

`nnml-newsgroups-file`
: `nnml` グループ記述ファイル。See Section 12.7.8.2 [Newsgroups File Format], p. 383. ディフォルトは`~/Mail/newsgroups` です。

`nnml-get-new-mail`
: `nil` でなければ、`nnml` は入って来たメール読み込みます。ディフォルトは`t` です。

`nnml-nov-is-evil`
: `nil` でなければ、このバックエンドはどのNOV ファイルも無視します。ディフォルトは`nil` です。

`nnml-nov-file-name`
: NOV ファイルの名前。ディフォルトは`.overview` です。

`nnml-prepare-save-mail-hook`
: 保存する前に一つの記事に範囲を狭めて実行するフックです。

`nnml-use-compressed-files`
: 非-`nil` だったら、`nnml` は圧縮されたメッセージファイルを扱うことができるようになります。ただし`auto-compression-mode` が有効になっていなければなりません(see Section "Compressed Files" in *The Emacs Editor*)。`nnml-use-compressed-files` の値が文字列だった場合、それは圧縮プログラムを指定するファイル拡張子として使われます。Emacs がそれをサポートしていれば、それを'`.bz2`' に設定することができます。値`t` は'`.gz`' と等価です。

`nnml-compressed-files-size-threshold`
: メッセージファイルを圧縮するかどうかを判断するための、サイズの閾値です。`nnml-use-compressed-files` が非-`nil` に設定されていて、本文の文字数がこの変数の値より大きかったら、メッセージファイルは圧縮されます。

`nnml` グループとNOV ファイルの調子が完全に狂ってしまったら、*M-x nnml-generate-nov-databases* とタイプすることによって、完全に更新することができます。この命令は、それぞれすべてのファイルを見ることによって`nnml` 階層全体をトロール魚網でさらうので、それが終わるまでには時間がかかるかもしれません。この機能へのより良いインターフェースはサーバーバッファーで見つかるでしょう(see Section 7.1.2 [Server Commands], p. 147)。

訳注: 単一の`nnml` グループのNOV データベースを再生成させるための`nnml-generate-nov-databases-1` という命令もあります。

7.4.13.4 MH スプール

`nnmh` は、NOV データベースを作らないこととアクティブファイルや印ファイルを保持しないことを除いて、`nnml` と似ています。このことは`nnmh` を`nnml` よりかなり 遅いバックエンドにしていますが、procmail のスクリプトを書くことはずっとやりやすくなってもいます。

仮想サーバーの設定:

`nnmh-directory`
: すべての`nnmh` ディレクトリーはこのディレクトリーの下に置かれます。ディフォルトは`message-directory` の値(そのディフォルトは`~/Mail`) です。

`nnmh-get-new-mail`
: `nil` でなければ、`nnmh` は入ってくるメールを読み込みます。ディフォルトは`t` です。

`nnmh-be-safe`
: `nil` でなければ、`nnmh` はフォルダーにある記事が実際に Gnus が考えているものと同じであるかを調べるという馬鹿げたことをやります。それは日付と目に入るすべての情報を調べるので、これを`t` に設定すると深刻な速度低下が起こります。`nnmh` の記事を読むのに Gnus 以外のものを使っていないのであれば、この変数を`t` に設定する必要はありません。ディフォルトは`nil` です。

7.4.13.5 Maildir

nnmaildir は各々の Gnus のグループに対応する maildir に、maildir フォーマットでメールを格納します。このフォーマットはhttps://cr.yp.to/proto/maildir.html で文書化されています。またnnmaildir は maildir の中の.nnmaildir/ ディレクトリーに追加の情報を格納します。

Maildir フォーマットは、配送と購読を、ロックを必要とせずに同時に行なうことができるようにするために設計されました。他のバックエンドでは、メールを何らかのスプールに渡した後で、そのスプールからグループに分割するために、Gnus を設定しなければならないでしょう。それは今まで通りnnmaildir で行なうことができますが、もっと普通のやり方は、Gnus のグループとして現われる maildir に配送されたメールを、直接手にすることです。

nnmaildir は完全に信頼できることを目指しています: *C-g* はメモリー中のデータを壊さないし、SIGKILL がファイルの中のデータを壊すことはありません。

nnmaildir は記事の印とNOV データを、それぞれの maildir に格納します。それによって、ある Gnus の環境から別の場所に maildir 全体をコピーすることができ、印は保持されます。

仮想サーバーの設定:

directory
: それぞれのnnmaildir サーバー(一つを越えるサーバーが必要だとはとても思えませんが) に対してディレクトリーを作り、それを maildir または maildir へのシンボリックリンクとして実装する必要があります(maildir のためだけにです。他の目的のためにすでに使われているディレクトリーを選んではいけません)。それぞれの maildir は、そのサーバーのニュースグループとして Gnus に現れ、シンボリックリンクのファイル名がそのグループの名前になります。ディレクトリーにある‘.’ で始まるどんなファイル名も無視されます。ディレクトリーは最初に Gnus を起動したときとグループバッファーで*g* をタイプしたときはいつでも走査され、どれかの maildir が削除または追加されていると、nnmaildir はそのときにそれを知ります。

  directory パラメーターの値は Lisp 式でなければなりません。それはこのサーバーのためのディレクトリーのパスを得るためにeval とexpand-file-name で処理されます。その式はサーバーが開かれたときだけeval され、その結果得られた文字列が、サーバーが閉じられるまで使われます(もし、式やeval を知らなくても心配ご無用; 単なる文字列で動作します)。このパラメーターは任意ではなく、必ず設定しなければなりません。"~/Mail" やそれのサブディレクトリーを使うことは推奨しません。なぜかと言うと、Gnus の他の複数の部分がそれをディフォルトでいろんなものに使うので、nnmaildir でもそれを使うと混乱するかもしれないからです。"~/.nnmaildir" が一般的な値です。

target-prefix
: これは Lisp 式でなければなりません。それはeval とexpand-file-name で処理されます。その式がeval されるのはサーバーが開かれたときだけで、その結果得られた文字列がサーバーが閉じられるまで使われます。

  nnmaildir サーバーにグループを作ると、その名前の頭にtarget-prefix が付加された maildir と、その maildir を指し示すシンボリックリンクが素のグループ名の名前で作成されます。したがって、directory が"~/.nnmaildir"

で、target-prefix が"../maildirs/" だった場合にfoo というグループを作ると、nnmaildir は maildir として~/.nnmaildir/../maildirs/foo を、../maildirs/foo へのシンボリックリンクとして~/.nnmaildir/foo を作成します。

同じdirectory に maildirs とシンボリックリンクの両方を作成するために、スラッシュを含まない文字列をtarget-prefix に設定することができます。この場合は、directory で見つかる名前がtarget-prefix で始まるどの maildir も、グループとは見なされません(が、それらを指し示すシンボリックリンクがグループになります)。

特別な場合としてtarget-prefix が"" (それがディフォルトです) だったら、グループを作るときに、対応するシンボリックリンクを持たない maildir がdirectory において作成されます。そのようなグループに対しては、force 引数を与えないとgnus-group-delete-group が使えないことに気をつけてください。

directory-files
: これはdirectory-files と同じインターフェースを持っている関数(またはdirectory-files そのもの) でなければなりません。これは maildir 用のサーバーのdirectory を走査するために使われます。このパラメーターは任意です。ディフォルトは、nnheader-directory-files-is-safe がnil だったらnnheader-directory-files-safe で、それ以外の場合はdirectory-files です(nnheader-directory-files-is-safe はサーバーが開いたときに一回だけ検査されますが、ディレクトリーが走査されるときに毎回チェックさせたいのならば、それを行なう関数をあなたが自前で用意する必要があります)。

get-new-mail
: 非-nil にしておくと、いつもの通りにグループの maildir 自体において新着メールを走査した後で、このサーバーはさらにmail-sources から、nnmail-split-methods かnnmail-split-fancy の設定に従って、従来の Gnus の方法でメールを取り込みます。ディフォルト値はnil です。

  mail-sources とnnmaildir グループの両方で同じ maildir を使ってはいけません。その結果は運良く有益になるかもしれませんが、そんな意図では設計されていませんし、将来は違う結果をもたらす可能性があります。

7.4.13.6 グループパラメーター

nnmaildir は複数のグループパラメーターを使います。これらのすべてを無視しても安全です。ディフォルトのnnmaildir の動作は、他のメールバックエンドのディフォルト(記事が一週間後に消去される、など) と同じです。期限切れ消去のパラメーターを除いて、この機能はすべてnnmaildir だけにあるものです。したがって、別のバックエンドですでに行なっている動作を単に踏襲させようというのであれば、これを無視することができます。

これらのパラメーターのうちのどれでも、その値がベクトルである場合は、オリジナルの値に代わって、第一の要素が Lisp 式として評価された結果が使われます。値がベクトルでない場合は、その値そのものが Lisp 式として評価されます。(それが、これらのパラメーターが他とは違う名前、すなわち他のバックエンドでサポートされているものとは違うけれども似た意味を持っている同様のパラメーターを使っている理由です。) (数値、文字列、nil、およびt についてもeval の関与を無視することができます。他の値について、そ

うすることがふさわしい場合には、追加のクオートを使い、かつベクトルで値を包むことを忘れないでください。)

`expire-age`
: 記事が消去されるまでの寿命の秒数を指定する整数、あるいは記事が期限切れ消去されてはならないことを指定する`never` というシンボルです。このパラメーターが設定されていないと、いつもの`nnmail-expiry-wait` 変数または`nnmail-expiry-wait-function` 変数を最後のよりどころにします(`expiry-wait` グループパラメーターが設定されていると、その値が`nnmail-expiry-wait` より優先して使われ、`nnmail-expiry-wait-function` は無効にされます)。3 日の値が必要ならば、`[(* 3 24 60 60)]` のようなものを使ってください。`nnmaildir` は式を評価して、その結果を使います。記事の寿命は記事ファイルの変更時刻を基点に計測されます。通常これは記事が配送された時刻と同じですが、記事の編集はそれを若くします。(期限切れ消去以外の) 記事の移動もまた、記事を若くしてしまうでしょう。

`expire-group`
: これが以下のような完全な Gnus のグループ名の文字列で、

```
"backend+server.address.string:group.name"
```

かつこのパラメーターが設定されているグループの名前と同じではなかったら、期限切れ消去が行なわれる際に、記事は消去される代わりに、これで指定されたグループに移動させられます。これが`nnmaildir` グループに設定されていると、移動先のグループにおいて、記事は元のグループにあったときとちょうど同じ古さになります。 したがって、移動先のグループにおける`expire-age` には注意してください。これがパラメーターが設定されているのと同じグループの名前に設定されると、記事はまったく期限切れ消去されません。ベクトルの式を使うと、最初の要素が一回、それぞれの記事について評価されます。したがって記事をどこに置くかを決めるために、その式は`nnmaildir-article-file-name` などに照会することができます。たとえこのパラメーターが設定されていなくても、`nnmaildir` は`expiry-target` グループパラメーターや`nnmail-expiry-target` 変数を顧みません。

`read-only`
: これが`t` に設定されていると、`nnmaildir` はその記事をこの maildir では読み出し専用として扱います。この意味は、記事は`new/` から`cur/` に改名されない、記事は`cur/` ではなく`new/` でのみ見つかる、記事は消去されない、記事は編集できない、ということです。`new/` は他の maildir の`new/` ディレクトリーへのシンボリックリンクであると想定されます(そのディレクトリーには、例えばみんなが興味があるメーリングリストを含んでいる、システムで共通のメールボックスがあります)。`new/` 以外の maildir にあるすべてのものは、読み出し専用として扱われません。したがって、みんなで共有するメールボックスに対しては、あなた自身の maildir を設置する(または共有のメールボックスに書き込み権限を持つ) 必要が依然としてあります。そうすれば、あなたの maildir は記事の余分なコピーをまったく含まなくて済むでしょう。

`directory-files`
: `directory-files` と同じインターフェースの関数です。記事を見つけるために、このグループに対応する maildir のディレクトリーを走査するために使わ

れます。ディフォルトはそのサーバーのdirectory-files パラメーターで設定されている関数です。

distrust-Lines:
非-nil にしておくと、nnmaildir はLines: ヘッダーフィールドを使う代わりにいつも記事の行数を数えます。nil だった場合は、あればそのヘッダーフィールドが使われます。

always-marks
['(read expire)] のような印シンボルのリストです。Gnus が記事の印をnnmaildir に尋ねるときはいつでも、ファイルシステムに格納されている印が何であるかとは無関係に、nnmaildir はすべての記事がこれらの印を持っていると答えます。これは機能を検証するためのもので、おそらく結局は削除されるでしょう。それは Gnus 本体で行なわれるか、あるいは有益でなければ放棄されるべきです。

never-marks
['(tick expire)] のような印シンボルのリストです。Gnus が記事の印をnnmaildir に尋ねるときはいつでも、ファイルシステムに格納されている印が何であるかとは無関係に、nnmaildir はこれらの印を持っている記事は無いと答えます。never-marks はalways-marks よりも優先されます。これは機能を検証するためのもので、おそらく結局は削除されるでしょう。それは Gnus 本体で行なわれるか、あるいは有益でなければ放棄されるべきです。

nov-cache-size
NOV メモリーキャッシュのサイズを指定する整数です。スピードアップのために、nnmaildir はそれぞれのグループの限定された数の記事に対して、メモリー上にNOV データを保持します。(これはたぶん有用ではなく、将来はおそらく削除されるでしょう)。このパラメーターの値は、サーバーが開かれた後で最初にグループが見られたとき、すなわち一般には最初に Gnus を起動したときだけ注目されます。サーバーが閉じられて再び開かれるまでは、NOV キャッシュのサイズは変更されません。ディフォルトは概略バッファーに表示される記事の数の見積り(tick 印がある記事の数かread が無い記事の数に、少々の余分を加えたもの) です。

7.4.13.7 記事の識別

記事はそれぞれの maildir のcur/ ディレクトリーに格納されます。各々の記事にはuniq:info のような名前が付けられます。ここでuniq はコロンを含みません。nnmaildir は:info の部分を保持しますが無視します。(他の maildir リーダーは一般に印を格納するためにこの部分を使います。) uniq の部分は記事をユニークに識別し、maildir の.nnmaildir/ サブディレクトリーの色々な場所に、対応する記事の情報を格納するために使われます。記事の完全なパス名は、概略バッファーで記事を要求した後でnnmaildir-article-file-name 変数から得られます。

7.4.13.8 NOV データ

uniq によって識別される記事は、そのNOV データ(概略バッファーの行を生成するために使われる) を.nnmaildir/nov/uniq に格納します。nnmaildir-generate-nov-databases 関数はありません。(その必要はあまりありません。記事のNOV データは記事かnnmail-extra-headers が変化したときに自動的に更新されます。) 対応するNOV ファイルを消

すことによって、単一の記事だけのNOV データの生成をnnmaildir に強制することはできます。しかしご用心。これはnnmaildir にこの記事に新しい記事番号を割り振らせるので、seen 印、エージェント、およびキャッシュにとって面倒なことになります。

7.4.13.9 記事の印

.nnmaildir/marks/flag/uniq ファイルがある場合に、uniq によって識別される記事は、flag 印を持つものと考えられます。Gnus がnnmaildir にグループの印を尋ねると、nnmaildir はそのようなファイルを探して、見つけた印のセットを報告します。Gnus がnnmaildir に印のセットを格納することを要求すると、nnmaildir は必要に応じて対応するファイルを生成し、または消去します。(実際は、それぞれの印のために新しいファイルを作るのではなく、i ノードを節約するために単に.nnmaildir/markfile へのハードリンクを張ります。)

.nnmaildir/marks/ に新しいディレクトリーを作ることによって、新しい印を創造することができます。印を保持しつつ maildir を tar でまとめてサーバーからそれを削除し、後で tar をほどくと、印は保持されています。印ファイルを作成または消去することによって、あなた自身が印を追加または削除することができます。Gnus が動作していてnnmaildir サーバーが開いているときにこれを行なう場合は、最初にすべてのnnmaildir グループの概略バッファーから退出してグループバッファーで*s* をタイプし、その後グループバッファーで*g* か*M-g* をタイプするのが最良です。そうしないと Gnus は変更を捉えてくれずに、それらを元に戻してしまうかもしれません。

7.4.13.10 メールフォルダー

nnfolder はそれぞれのメールグループを別々のファイルに格納するバックエンドです。それぞれのファイルは標準の Un*x mbox 様式です。nnfolder は記事番号と到着時刻を見失わないようにするための追加のヘッダーを加えます。

仮想サーバーの設定:

nnfolder-directory
: すべてのnnfolder メールボックスはこのディレクトリーの下に置かれます。ディフォルトはmessage-directory の値(そのディフォルトは~/Mail) です。

nnfolder-active-file
: アクティブファイルの名前。ディフォルトは~/Mail/active です。

nnfolder-newgroups-file
: グループ記述ファイルの名前。See Section 12.7.8.2 [Newsgroups File Format], p. 383. ディフォルトは~/Mail/newsgroups" です。

nnfolder-get-new-mail
: nil でなければ、nnfolder は入ってくるメールを読み込みます。ディフォルトはt です。

nnfolder-save-buffer-hook
: フォルダーを保存する前に実行されるフックです。nnfolder バッファーに対してさえも、Emacs は通常とおりファイル名を変更してバックアップを行なうことに注意してください。この機能を無効にしたいのであれば、~/.gnus.el ファイルで次のようなことをすれば良いでしょう:

```
(defun turn-off-backup ()
  (set (make-local-variable 'backup-inhibited) t))
```

```
(add-hook 'nnfolder-save-buffer-hook 'turn-off-backup)
```

`nnfolder-delete-mail-hook`
: これから消去されるメッセージに範囲を狭められて実行されるフックです。この関数は別の場所にメッセージをコピーしたり、消去する前に何らかの情報を取り出すために使うことができます。

`nnfolder-nov-is-evil`
: もし非-`nil` なら、このバックエンドはどんなNOV ファイルをも無視します。ディフォルトは`nil` です。

`nnfolder-nov-file-suffix`
: NOV ファイルの拡張子です。ディフォルトは`.nov` です。

`nnfolder-nov-directory`
: NOV ファイルが格納されるディレクトリーです。`nil` だったら`nnfolder-directory` が使われます。

`nnfolder` で読みたいたくさんの`nnfolder` に似たファイルを持っているのなら、そのようなすべてのファイルが`nnfolder-directory` にあることを`nnfolder` に気付かせるために、*M-x nnfolder-generate-active-file* 命令を使ってください。もっとも、これは長いファイル名を使っているときだけ動作しますが。

7.4.13.11 メールバックエンドの比較

まず用語としての「バックエンド」(back end) は、それによってなにものかが取得される、低次のアクセス手段、あるいはそう言いたければ輸送手段です。それが意図するのはどこからかメールを取ってくることなので、Gnus がすぐに手が届く距離の範囲内でメールを受け取るための、適当なバックエンドを選択する必要があります。

同じ概念が Usenet 自身にも存在します。近ごろでは記事へのアクセスは一般的にNNTP で行なわれますが、凄涼たる暗黒の昔には、世界中の誰もが、記事を置いてあるマシン(今日ではNNTP サーバーと呼ぶもの) でリーダーを動作させることによって Usenet に接続したものでした。また、アクセスは記事のディレクトリーのスプールの領域に直接に踏み込むリーダーによって行なわれました。たまたまそういうサーバーにいるのなら(あるいは NFS を介して、とにかくそれのスプールのディレクトリーを見ることができるのなら)、今でも`nntp` か`nnspool` バックエンドのどちらかを選ぶことができます。

> (訳注:「凄涼たる暗黒の昔には」はポーの詩「大鴉」の冒頭部分“Once upon a midnight dreary”。)

メールバックエンドを選択することの行き着く先は、元の形式を処理し、かつ将来便利に使える形式でメールを残すことを、同時に実現するのに適した方法を選び出すことです。それぞれいくつかの良い点と悪い点があります:

`nnmbox`
: 歴史的に UNIX システムは、とても一般的で行き届いた定義のたった一つの形式を持っています。すべてのメッセージは単一の「スプールファイル」に到着し、それらは正規表現‘`^From_`’ に合致する行で区切られています。(‘`_`’ という記号はスペースを意味し、この例ではこれが RFC で規定されている‘`From:`’ ヘッダーではないことをはっきりさせるために使っています。) Emacs それに Gnus も歴史的に Unix 環境から始まっているので、元の mailbox 形式をあまりいじくり回さずに済めば、それが最も単純です。したがってこのバックエン

ドを選んだ場合に、本当のスプールからメールを取得して Gnus にとって都合が好いディレクトリーにメールを移動するために Gnus が主に行なうのは、処理の過程で何も(目立つような) 変更をせずに、単にそれを複製することです。それは Gnus が処理を行なうことができる環境にメールを移動するための「最も気が利かない」方法です。これは移動させることを速くしますが、Gnus がどこに何があるかを調べるときは、解析が遅くなります。

`nnbabyl` むかしむかしあるところに DEC-10 と DEC-20 がありました。それらは TOPS というオペレーティングシステムや似たようなものを実行していて、メールを読むための普通の(もしかしたら唯一の?) 環境は Babyl というものでした。そのシステムに届いたメールでどんな形式が使われていたかはわかりませんが、Babyl にはメールを変換するための、それ用の内部形式がありました。その変換とは、Babyl 特有のヘッダーと状態ビットを、ファイルにあるそれぞれのメッセージの先頭に挿入するための仕組みによって、スプールファイル風の実体を作ることでした。Rmail は Emacs の最初のメールリーダーで、Richard Stallman によって書かれました。Stallman はその TOPS/Babyl の環境の出身だったので、すでに存在していたメールファイルの一族を理解するように Rmail を書きました。Gnus は(この件に関しては VM も) この形式をサポートし続けています。それは、そのメーラー特有のヘッダー/ステータス・ビットというものが、かなり良質だと認められているからです。Rmail 自身ももちろんまだ存在していて、今でも Emacs の中で維持されています。Emacs 23 から、Babyl に代わって標準の mbox 様式が使われるようになりました。

上記の両方の形式は、メールをファイルシステムにおける単一のファイルに置いたままにするので、メールを見るたびにファイル全体を解析しなければなりません。

`nnml` `nnml` は、あたかも`nnspool` でアクセスされる Usenet システムで実際に操作しているかのような感じのするバックエンドです。(実際のところ、`nnml` はすごく以前に`nnspool` から枝分かれしたものだと思います。) メールは元のスプールファイルから取り出された後で、個々のファイルに 1:1 で切り分けられます。Usenet 様式のアクティブファイル(INN や CNews に基づいたニュースシステムの`/var/lib/news/active` ファイル(例えば) や、‘`NNTP LIST`’ 命令で返されるものに類似したもの) を維持し、今ではかなりの年数にわたってNNTP サーバーのために定義されている*overview* ファイルも、グループへ入るときの効率を良くするために作成します。たくさんのファイルを作成し、`nnml` アクティブファイルを更新し、さらにメッセージ毎に overview への追加を行なうので、メール分割では遅くなりますが、アクセスするときには、アクティブファイルと overview によって提供される索引機能に支援されて、とてつもなく速くなります。

`nnml` は*inode* を非常にたくさん消費します。すなわち、新しいファイルを置くことができる場所をファイルシステム上に定めるための資源を、たくさん占有します。ぎっしりつまった共有ファイルシステムで大量の inode を占有することを、システム管理者は快く思いません。もっとも、そのファイルシステムが自分自身のもので、容量が希少ではない個人のマシンにいるのならば、`nnml` には非常に大きな利点があるのですが。

FAT16 の Windows の世界にいる場合にも、たくさんの小さなファイルで多くの場所を取られてしまう点で問題があります。

`nnmh` Rand MH メール閲覧システムは UNIX システムにかなり長い間存在しています。それはメッセージのスプールファイルを個々のファイルに分割することによって動作しますが、索引機能は少ししか、あるいはまったくありません—`nnmh` は、意味的には「アクティブファイルまたは overview の無い`nnml`」と等価です。これはおそらく最悪の選択でしょう。なぜならば、個々のファイルを作ることの遅さが、何がグループで新しいかを知るときに解析するために行なうアクセスの遅さに結び付くからです。

`nnfolder` 基本的に`nnfolder` が実現することは、グループ毎の`nnmbox` (上で説明されている最初の方法) です。すなわち`nnmbox` 自体はすべて のメールを一つのファイルに入れます。でも`nnfolder` はメールグループのそれぞれが Unix mail box ファイルを持つように、ほんの少し最適化をします。それぞれのグループは別々に解析されるので`nnmobx` よりも速く、しかもなお、メールを移動させるのに最小限の労力しか要求しない、単純な Unix mail box 形式を提供します。加えて「アクティブ」ファイルを維持し、Gnus がそれぞれの別のグループにどのくらいのメッセージがあるかを調べることをとても速くします。

もしたくさんの量のメッセージを受け取ることが予想されるグループがあるなら、`nnfolder` は最善の選択ではありませんが、ほどほどの量のメールしか受け取らないなら、おそらく`nnfolder` はすべての中で最も都合の良いバックエンドでしょう。

`nnmaildir`

期限切れ消去その他もろもろを設定するのに、`nnmaildir` は他のメールバックエンドとは少々異なった、互換性の無いグループパラメーターを使います。

`nnmaildir` は大方`nnml` と似たものですが、いくらか顕著な違いがあります。それぞれのメッセージは別々のファイルに格納されますが、ファイル名は Gnus の記事番号と関係がありません。また`nnmaildir` は`nnml` の overview に相当するファイルを記事ごとに一つ格納するので、`nnml` の約二倍の量の i ノードを使います。(`df -i` を使って i ノードの割り当てがどれほどたくさんあるかを調べてください。) そのために遅くなったり多くの場所を取ってしまうようならば、非ブロック構造のファイルシステムを使ってください。

maildir は受信配送のためのロックを必要としないので、あなたがグループとして使っている maildir は、配送されてきたメールを直接受け取るための maildir にすることもできます。これは、メールが配送されてくる過程で異なるメールボックスに仕分されるようになっているのならば、Gnus のメール分割を省略できることを意味します。`mail-sources` における`directory` の項には(訳注: maildir を使わなくても) 似た効果がありますが、配送されてくるメールをスプールするためのメールボックスの一揃い(mbox 形式ではそのためにメッセージの本文が壊れる) と、他の(何であれあなたの好みの形式の) グループとして使われる組が必要です。一方 maildir は、`new/` サブディレクトリーに置かれる組み込みスプールを持ちます。メール分割による代わりに`new/` から`cur/` に移動されたメールは、ダブっているかどうかをチェックするような処理を今のところは受けないことに注意してください。

`nnmaildir` はグループの記事の印を、それに対応する maildir に格納します。Gnus の外からそれらを簡単に操作できるようにするために、そのように作られているのです。maildir を tar でまとめてから別のどこかで展開しても、印はそのままです。

nnmaildir は速度を上げるためにかなりの量のメモリを使います。(nnml の場合はファイルに格納し、nnmh では何度もメッセージファイルを解析して得るものごとを、それはメモリ上に保持します。) これがあなたにとって問題ならば、nov-cache-size グループパラメーターを何か小さな値(0 はおそらくだめですが 1 だったらたぶん働きます) に設定することによって、少ないメモリで済むようにすることができます。このキャッシュ機構は、おそらく将来は削除されるでしょう。

起動は他のバックエンドよりもnnmaildir の方が遅いでしょう。ファイルシステムに依存していないすべての部分では速いでしょう。

nnmaildir はnnoo を使わないので、nnmaildir から派生したバックエンドを書くのにnnoo は使えません。

7.5 ウェブの閲覧

ウェブに基づいた議論の場はどんどん広まっています。多くの分野で、ウェブの掲示板は最も重要な場になり、メーリングリストやニュースグループの重要性を翳らせています。理由は簡単です---新しい利用者が使い易いからです。ただ場所をクリックするだけで、議論の場があります。メーリングリストでは、面倒な購読手続きをしなければならず、ほとんどの人はニュースグループというものがが何であるかすら知りません。

この筋書きから浮かび上がる問題は、ウェブブラウザーはニュースリーダーとしてはあまり良くないということです。どんな記事を読んだかを記録しません。興味のある表題にスコアを付けることができません。オフラインで読むことができません。何度もクリックすることを要求し、最後にはあなたを怒らせます。

ならば---ウェブブラウザーが掲示板を読むのに適していないのなら、代わりに Gnus を使いませんか?

Gnus はこれらのソースへのインターフェースを提供するバックエンド群を少し備えつつあります。

これらのウェブソースの一番の問題は、長期間は動作しない可能性が高いことです。HTML のデータから情報を拾い集めるのはせいぜい推測で、その構造が変化したときには、Gnus バックエンドは動作しません。でも、ある程度新しいバージョンのバックエンドを使っていれば大丈夫のはずです。

これらのウェブの手段に共通することは、ウェブソースはしばしば落ちていたり、使用可能でなかったり、はっきり言って楽しむには遅すぎる、ということです。そういう場合に、Gnus Agent (see Section 7.9 [Gnus Unplugged], p. 216) に記事のダウンロードを任せて、ローカルディスクから好きなときに読むようにすることは、大いに意義があります。これで World Wide Wait とはおさらばです。

7.5.1 ウェブ検索

ううむ、まあ、調べたい文字が書いてある記事を、その、Usenet で探すということは、ですね、もちろん素晴らしいことこの上ないのではありますが、しかし、何と申しましょうか、ウェブブブラウザーといいますか、そういうものを使ってことを行なうのは、何ともその、はばかりながら、ぶざまと言いますか、そうすると、コマーシャルを見ないわけにはいかないのでありまして、しかるに、Gnus を使えばブラウザー無しで検索することができます。

nnweb バックエンドは、強力な検索エンジンへの簡単なインターフェースを提供します。nnweb グループを作成し、検索パターンを入力してから、そのグループに入って他の普

通のグループのように記事を読んでください。グループバッファー(see Section 3.9 [Foreign Groups], p. 21) の`G w` 命令によって、手軽にこれができます。

`nnweb` グループは、固定グループになろうとはしません---このグループでは記事番号はごく一時的なものとして扱われます。実際、`nnweb` グループに入るたびに(たとえ検索パターンを変更していなくても)、記事の順序が違っているかもしれません。また、重複抑制(see Section 4.30 [Duplicate Suppression], p. 121) を使っても役に立たないでしょう。というのは、検索エンジン(例えば Google) を使って記事を読み込む前の段階では、`nnweb` はそれらの`Message-ID` を知らないからです。あなたが読んだ記事を憶えておくための唯一の方法は、`Date` ヘッダーを元にスコアを付けることだけです---つまり、そのグループを最後に読んだ日付より前に投稿された記事を、すべて既読にするということです。

もし検索エンジンの出力形式が実質的に変更されると、`nnweb` はそれをうまく解釈できなくて、処理に失敗するでしょう。ウェブの提供者たちがそんなことをしても、彼らを責めることはできないでしょう---それは広告で金を稼ぐのが彼らのレーゾン・デートル (存在理由) であり、社会にサービスを提供することではないからです。`nnweb` はすべての記事から広告を洗い流してしまうので、提供者たちがムカついていると思われるかもしれません。まあ見ててください。

以下は仮想サーバー変数です。

`nnweb-type`
: どの検索エンジンを使うかを指定します。現在サポートされている種類は、`google` と`dejanews` です。`dejanews` は`google` の別名になっていることに注意してください。

`nnweb-search`
: 検索エンジンに与える検索文字列です。

`nnweb-max-hits`
: 一つの検索で表示する最大のヒット数の希望値で、ディフォルトは 999 です。

`nnweb-type-definition`
: 種類と定義の連想リストです。この連想リストは、さまざまな検索エンジンの種類に対して、`nnweb` がどうすべきかを指定します。以下に示す要素を与えなくてはなりません。

  `article` 記事をデコードし、Gnus が理解できる何かを提供する関数です。

  `map` メッセージヘッダーと URL を、記事番号を元にして得るための連想リストを作成する関数です。

  `search` 検索エンジンに検索文字列を送るための関数です。

  `address` 前述の関数が検索文字列を送るべきアドレスです。

  `id` `Message-ID` を元にして記事を取得するための、URL フォーマットの文字列です。

7.5.2 RSS

いくつかのウェブサイトは RDF site summary (RSS) を持っています。RSS は、ニュース関連のサイト(BBC や CNN のような) の見出しを要約するためのフォーマットです。しかし、基本的にリストのようなものなら何でも、RSS feed として提供することができます:

weblogs, changelogs あるいは wiki (例えば`https://cliki.net/site/recent-changes`) の最新の変更などが対象になります。

RSS はとても規則的で良質なインターフェースを持っているので、Gnus がグループを最新の状態に保っておくために必要な情報を得ることが可能です。

Feed を購読するには、グループバッファーから*G R* を使ってください---feed の所在、タイトルおよび説明の入力を求められるでしょう。タイトルはどんな文字でもよく、それはグループ名とグループのデータ・ファイルの名前に使われます。説明は省略できます。

簡単に`nnrss` を始める方法は、グループバッファーで*B nnrss* `RET RET` *y* のようなことを唱え、そしてグループを購読することです。

`nnrss` バックエンドは、それぞれの`nnrss` グループのためのデータ・ファイルを`nnrss-directory` (下記参照) に保存します。非-ASCII 文字を含んでいるファイル名は、`nnmail-pathname-coding-system` 変数または他のもので指定された coding system でエンコードされます。詳細はここ(see Section 3.17 [Non-ASCII Group Names], p. 41) を見てください。

`nnrss` バックエンドは、それぞれが‘`text/plain`’ パートと‘`text/html`’ パートを含んでいる‘`multipart/alternative`’ 型のMIME 記事を作ります。

あなたの購読目録をOPML フォーマット(Outline Processor Markup Language) でロード/セーブするために、以下のコマンドを使うこともできます。

`nnrss-opml-import` *file* [関数]
OPML ファイルの入力を促し、そのファイルにあるそれぞれの feed を購読します。

`nnrss-opml-export` [関数]
現在のRSS 購読目録をOPML フォーマットでバッファーに書き出します。

以下の`nnrss` 変数が変更可能です:

`nnrss-directory`
`nnrss` がファイルを書き込むディレクトリーで、ディフォルトは`~/News/rss/` です。

`nnrss-file-coding-system`
`nnrss` グループのデータ・ファイルを読み書きするときに使われる coding system です。ディフォルトは`mm-universal-coding-system` の値(そのディフォルトは`utf-8-emacs`) です。

`nnrss-ignore-article-fields`
いくつかのフィードは、記事フィールドの例えばコメント数を、その存続期間を通じて絶えず更新します。しかしそれはローカルに保存したものとの差異を生むので、サーバーに新しい記事があるように解釈されてしまいます。いくつかのフィールドを無視してこれを防ぐためには、この変数に無視するべきフィールドのリストを設定してください。ディフォルトは`'(slash:comments)` です。

`nnrss-use-local`
`nnrss-use-local` を`t` に設定すると、`nnrss` は`nnrss-directory` にあるローカルファイルから feed を読みます。`nnrss-generate-download-script` コマンドを使うことによって、`wget` を使ったダウンロード・スクリプトを作ることができます。

概略バッファーに説明を表示させたいならば、以下のコードが役に立つでしょう。

```
(add-to-list 'nnmail-extra-headers nnrss-description-field)
(setq gnus-summary-line-format "%U%R%z%I%(%[%4L: %-15,15f%]%) %s%uX\n")

(defun gnus-user-format-function-X (header)
  (let ((descr
         (assq nnrss-description-field (mail-header-extra header))))
    (if descr (concat "\n\t" (cdr descr)) "")))
```

以下のコードは、概略バッファーから直接 nnrss の url をオープンするのに便利でしょう。

```
(require 'browse-url)

(defun browse-nnrss-url (arg)
  (interactive "p")
  (let ((url (assq nnrss-url-field
                   (mail-header-extra
                    (gnus-data-header
                     (assq (gnus-summary-article-number)
                           gnus-newsgroup-data))))))
    (if url
        (progn
          (browse-url (cdr url))
          (gnus-summary-mark-as-read-forward 1))
      (gnus-summary-scroll-up arg))))

(with-eval-after-load "gnus"
  (keymap-set gnus-summary-mode-map "RET" 'browse-nnrss-url))
(add-to-list 'nnmail-extra-headers nnrss-url-field)
```

あなたがHTML パートを見たくないために‘text/html’ をmm-discouraged-alternatives 変数(see Section “表示のカスタマイズ” in *The Emacs MIME Manual*) に加えていたとしても、特にnnrss グループでは‘text/html’ を表示する方が便利かもしれません。以下はnnrss グループでだけは‘text/html’ パートを表示するために、グループパラメーターとしてmm-discouraged-alternatives を設定する例です:

```
;; mm-discouraged-alternatives のディフォルト値を設定。
(with-eval-after-load "gnus-sum"
  (add-to-list
   'gnus-newsgroup-variables
   '(mm-discouraged-alternatives
     . '("text/html" "image/.*"))))

;; nnrss グループでは ‘text/html’ パートを表示。
(add-to-list
 'gnus-parameters
 '("\\`nnrss:" (mm-discouraged-alternatives nil)))
```

7.5.3 Atom

いくつかの web サイトでは Atom 配信フォーマット(Atom Syndication Format) の web フィードを提供しています。Atom は機能が RDF Site Summary (see Section 7.5.2 [RSS], p. 199) に似ている web フィードのフォーマットです。

nnatom バックエンドを使うと、HTTP フィードやローカル Atom フィードを、例えばgnus-secondary-select-methods に Gnus サーバーとして追加したり、*Group* バッファーで"B" を押すことによって外部サーバーとして追加することができます(see Section 2.1 [Finding the News], p. 3)。それぞれのサーバーのアドレスはフィードの場所です(もっとも、そのアドレスは<http://> や<https://> では始まりませんが)。各サーバーにはフィードのエントリーが置かれている単一のグループがあります。

nnatom の機能は次の通りです:

- サーバーのデータはサーバーごとにgnus-directory のatom サブディレクトリに保存されます。
- フィード内各エントリのサマリーとコンテンツの両方について、記事のパートが生成されます。すべての MIME タイプのコンテンツは、(それらの MIME タイプがサポートされていて、フィードが MIME タイプを明示している限り) Gnus によって正しく表示されるはずです。
- 記事の変更と公開された日付は追跡され、もし変化があったら記事が更新されます。

7.6 その他のグループ源

Gnus はただ単にニュースやメールを読む以上のことができます。以下に示す方法によって、Gnus でディレクトリーやファイルを、あたかもニュースグループであるかのように閲覧することができるようになります。

7.6.1 ディレクトリーグループ

たくさんの記事が個別のファイルとして入っているディレクトリーがあれば、それをニュースグループとして扱うことができます。もちろん、ファイルは数字のファイル名をもっていなければなりません。

素晴らしい Emacs のパッケージの中でも最も素晴らしいange-ftp について触れるのに、ここは良い機会でしょう。私がnndir を書いたときは、これ(ディレクトリーを読むバックエンド) についてはあまり考えていませんでした。とんでもないことだね。

ange-ftp はこの情況を劇的に変化させました。例えばディレクトリー名としてange-ftp の様式で/ftp.hpc.uh.edu:/pub/emacs/ding-list/ というファイル名をディレクトリー名として入力したとすると、ange-ftp は実に「シナ」の向こうのディレクトリーをニュースグループとして読めるようになるのです。おーい、分散ニュースだぞーっ!

(訳注:「シナ」(原典‘sina’) は China のことか?)

nndir はNOV ファイル群が存在すればそれらを利用します。

nndir は「読み出し専用」のバックエンドです---この選択方法では、記事の削除や期限切れ消去を行なうことはできません。nndir が使えるものなら何でも、nnmh あるいはnnml でも使うことができるので、もし読み出し専用ではないnndir が必要だと思ったら、これらのどちらかの方法に切り替えることもできます。

7.6.2 なんでもグループ

nneething はnndir バックエンド(単一のスプール風ディレクトリーを読むバックエンド)のほんの少し先にあるもので、それはどんなディレクトリーでもニュースグループに見せかけてしまいます。不思議ですが真実です。

nneething にディレクトリーを与えると、そのディレクトリーを走査して各ファイルに記事番号を割り当てます。そのようなグループに入ったら、nneething は Gnus が使える「ヘッダー」を作らなくてはなりません。つまるところ Gnus はニュースリーダーなんです。忘れているかもしれないので念のため。nneething はこれを二段階で処理します。最初に、対象となるそれぞれのファイルを覗いてまわります。もしそのファイルが記事のように見えたなら(すなわち最初の数行がヘッダーのように見えたら) それをヘッダーとして使います。もしそれがヘッダーの無いただの適当なファイル(例えば C のソースファイル) だったら、nneething はヘッダーを虚空からでっち上げます。これはファイルの所有者、名前および日付を使い、それらの要素を元にできることを何でもやります。

これはあなたにとってはすべて自動的に起こることで、あなたはニュースグループにとても良く似た何かを見せられることになるでしょう。本当に寸分違わない、ニュースグループのようなものを。記事を選択すると、それはいつものように記事バッファーに表示されるでしょう。

ディレクトリーを表わしている行を選択すると、Gnus はいきなりあなたをこのnneethingグループのための新しい概略バッファーに連れて行くでしょう。以下同様に、あなたがそうしたければ、この方法で全ディスクを駆け巡ることができます。ですが、Gnus は本当は dired ではないし、そのように意図されたものでもないことは覚えておいてください。

ここでの動作には二つの全体的なモードがあります— 一時モードと固定モードです。一時的な操作を行なっているときは(すなわちグループバッファーで*G D*)、Gnus はどのファイルを読んだか、どのファイルが新しいか、などの情報を憶えておきません。普通に*G m* で固定nneething グループを作れば、Gnus は記事番号とファイル名の対応表を憶えておくので、このグループを他のグループと同様に扱うことができるようになります。固定nneething グループを活かすと、それが未読記事をいくつ含んでいるかを知らせてもらえる、等々の利便があります。

いくつかの変数があります:

`nneething-map-file-directory`
: すべての固定nneething グループの対応表が、このディレクトリーに格納されます。このディフォルトは`~/.nneething/` です。

`nneething-exclude-files`
: この正規表現に合致するファイルはすべて無視されます。自動保存ファイルなどを除外するのに便利に使えます。そしてそれがまさにディフォルトで行なわれる動作です。

`nneething-include-files`
: どのファイルをグループに含めるかを示す正規表現です。この変数がnil でなければ、この正規表現に合致するファイルだけが含まれます。

`nneething-map-file`
: 対応表ファイルの名前です。

7.6.3 文書グループ

nndoc は単一のファイルをニュースグループとして読むことをできるようにする、ちょっと気の利いたやつです。複数のファイルの種別がサポートされています:

`babyl` Babyl 様式。

`mbox` 標準 Unix mbox ファイル。

`mmdf` MMDF 形式のメールボックス。

`news` 一つのファイルにまとめられた複数のニュース記事。

`rnews` rnews のバッチ転送形式。

`nsmail` Netscape のメールボックス。

`mime-parts`
MIME のマルチパートのメッセージ。

`standard-digest`
標準(RFC1153) のまとめ送り形式。

`mime-digest`
MIME のまとめ送りメッセージ。

`lanl-gov-announce`
ロスアラモス国立研究所(LANL) Gov Announce からの発表メッセージ。

`git` git で commit したことを伝えるメッセージ。

`rfc822-forward`
RFC 822 またはその後継に従って転送されたメッセージ。

`outlook` Outlook のメールボックス。

`oe-dbx` Outlook Express の dbx メールボックス。

`exim-bounce`
Exim MTA から跳ね返されたメッセージ。

`forward` 非公式の規則で転送されたメッセージ。

`rfc934` RFC934 形式で転送されたメッセージ。

`mailman` mailman のまとめ送り。

`clari-briefs`
Clarinet のニュース項目を要約したまとめ送り。

`slack-digest`
非標準まとめ送り形式---だいたいのものを扱えるが、下手。

`mail-in-mail`
最後の手段。

特別な「ファイル種別」である`guess` を使うこともできます。これを使うと、見ているファイルの種別が何かを`nndoc` が推測しようとします。また、`digest` というファイル種別は、そのファイルがどのまとめ送り形式かを`nndoc` に推測させます。

nndoc はファイルを書き換えようとしたり、余分なヘッダーを挿入しようとしたりはしません---単に、ファイルをそのグループを作る元として使えるようにする、というようなことです。それだけのことです。

保存された古い記事を持っていて、それを新しくてかっこいい Gnus のメールバックエンドに追加したいなら、おそらくnndoc が助けになるはずです。例えば新しいnnml グループに振り分けたいメールが、今は古いRMAIL ファイルにメールがあるとしましょう。そういう場合は、そのファイルをnndoc を使って開き(グループバッファーで*G f* 命令(see Section 3.9 [Foreign Groups], p. 21) を使いましょう)、バッファー内の全記事にプロセス印を(例えば*M P b* で) 付けてから、それらがnnml グループ群に振り分けられるように(*B r* 命令を使って) 再スプールしてください。すべてがうまくいけば、RMAIL ファイル内のすべてのメールは、たくさんのnnml ディレクトリーの中にも格納されます。そうしたら、あの厄介なRMAIL を削除してしまっても良いでしょう。あなたにガッツがあれば!

仮想サーバー変数:

nndoc-article-type
: これはmbox, babyl, digest, news, rnews, mmdf, forward, rfc934, rfc822-forward, mime-parts, standard-digest, slack-digest, clari-briefs, nsmail, outlook, oe-dbx, mailman およびmail-in-mail またはguess のいずれかでなくてはなりません。

nndoc-post-type
: この変数は、そのグループをニュースグループとみなすかメールグループとみなすかを Gnus に伝えます。二つの有効な値はmail (ディフォルト) およびnews です。

7.6.3.1 文書サーバーの内部

nndoc で認識される新しい文書の種別を追加することは難しくありません。その文書がどのように見えるかの定義を仕上げ、その文書種別を認識するための述語関数を書いて、nndoc を手なずけるだけで良いのです。

まず、これが文書の種別の定義の例です:

```
(mmdf
 (article-begin .  "^\^A\^A\^A\^A\n")
 (body-end .  "^\^A\^A\^A\^A\n"))
```

この定義は種別を示すためのユニークな名前(*name*) と、それに続く仮想的な変数名およびその設定値の単純な連なりからなります。以下が使うことができる変数です---変数の数に圧倒されないでください。ほとんどの文書の種別は、ごくわずかな設定で定義することができます:

first-article
: これが設定されていると、nndoc はこの正規表現に合致する何かが見つかるまで、すべてのテキストを読み飛ばします。それより前のすべてのテキストは完全に無視されます。

article-begin
: この設定は、すべての文書の種別の定義に必ず存在しなければなりません。それぞれの記事の始まりがどのように見えるかを指定する正規表現です。単純な正規表現では対処できないもっと複雑なことをしたい場合は、これの代わりにarticle-begin-function を使うことができます。

`article-begin-function`
: これを設定する場合は、それぞれの記事の開始位置にポイントを移動させる関数を指定してください。これは`article-begin` より優先されます。

`head-begin`
: これを設定する場合は、記事のヘッダーの始まりに合致する正規表現を指定してください。単純な正規表現では対処できないもっと複雑なことをしたい場合は、これの代わりに`head-begin-function` を使うことができます。

`head-begin-function`
: これを設定する場合は、記事のヘッダーの開始位置にポイントを移動させる関数を指定してください。これは`head-begin` より優先されます。

`head-end`
: これを設定する場合は、記事のヘッダーの最後に合致する正規表現を指定してください。ディフォルトは‘`^$`’、つまり空行です。

`body-begin`
: これを設定する場合は、記事のボディーの始まりに合致する正規表現を指定してください。ディフォルトは‘`^\n`’ です。単純な正規表現では対処できないもっと複雑なことをしたい場合は、これの代わりに`body-begin-function` を使うことができます。

`body-begin-function`
: これを設定する場合は、記事のボディーの開始位置にポイントを移動させる関数を指定してください。これは`body-begin` より優先されます。

`body-end`
: これを設定する場合は、記事のボディーの最後に合致する正規表現を指定してください。単純な正規表現では対処できないもっと複雑なことをしたい場合は、これの代わりに`body-end-function` を使うことができます。

`body-end-function`
: これを設定する場合は、記事のボディーの最後の位置にポイントを移動させる関数を指定してください。これは`body-end` より優先されます。

`file-begin`
: これを設定する場合は、ファイルの始まりに合致する正規表現を指定してください。それより前のすべてのテキストは完全に無視されます。

`file-end`
: これを設定する場合は、ファイルの最後に合致する正規表現を指定してください。それより後ろのすべてのテキストは完全に無視されます。

このように`nndoc` はこれらの変数を使って、文書ファイルをそれぞれヘッダーとボディーを持った記事の連なりとして切り分けることができます。しかし、すべての文書の種別がこのようなニュース風になっているわけではないので、さらにヘッダーやボディーを Gnus の趣味に合うように変形させる変数が、いくらか必要になります。

`prepare-body-function`
: これに関数を設定しておくと、記事が要求されたときに呼び出されます。これはボディーの開始位置のポイントを引数として呼び出され、文書にいくつかのエンコードされた内容物のパートがある場合に有用です。

`article-transform-function`
: これに関数を設定しておくと、記事が要求されたときに呼び出されます。これは記事のヘッダーとボディーの両方に、より広範囲な変形を行なうために使われるものです。

`generate-head-function`
: これに関数を設定しておくと、Gnus が理解できるヘッダーを生成するために呼び出されます。これは記事番号をパラメーターとして呼び出され、その記事のための良質なヘッダーを生成することを求められます。すべての記事のヘッダーが要求されるときに呼び出されます。

`generate-article-function`
: これに関数を設定しておくと、Gnus が理解できる完全な記事を生成するために呼び出されます。これはすべての記事のヘッダーが要求されるときに、記事番号をパラメーターとして呼び出されます。

`dissection-function`
: これに関数を設定しておくと、それだけを使って文書ファイルを記事に切り分けるために呼び出されます。これは`first-article`, `article-begin`, `article-begin-function`, `head-begin`, `head-begin-function`, `head-end`, `body-begin`, `body-begin-function`, `body-end`, `body-end-function`, `file-begin` および`file-end` より優先されます。

私が出会った中で最も複雑な例を見てください。標準まとめ送り形式のためのものです:

```
(standard-digest
 (first-article . ,(concat "^" (make-string 70 ?-) "\n\n+"))
 (article-begin . ,(concat "\n\n" (make-string 30 ?-) "\n\n+"))
 (prepare-body-function . nndoc-unquote-dashes)
 (body-end-function . nndoc-digest-body-end)
 (head-end . "^ ?$")
 (body-begin . "^ ?\n")
 (file-end . "^End of .*digest.*[0-9].*\n\\*\\*\\|^End of.*Digest *$")
 (subtype digest guess))
```

70 文字のダッシュ('-') の行より前はすべて無視されるというのが分かりますね。また‘`^End of`’ で始まる行より後ろもすべて無視されます。各記事は 30 文字のダッシュの行で始まり、ヘッダーとボディーの区切りの行は一個のスペースを含むことがあり、そしてボディーはそれが渡される前に`nndoc-unquote-dashes` を通されます。

あなた独自の文書のための定義を`nndoc` で使えるようにするには、`nndoc-add-type` 関数を使ってください。これは二つのパラメーターをとります— 一つ目は定義そのもので、二つ目の(省略可能な) パラメーターは、この定義を文書の種別を定義する連想リストのどこに置くかを指定します。この連想リストは順番に走査され、与えられた種別*type* に対して`nndoc-`*type*`-type-p` が呼び出されます。したがって、例えば`mmdf` という種別であるかどうかを調べるためには`nndoc-mmdf-type-p` が呼び出され、他の種別の場合も同様です。これらの種別述語関数は、文書がその種別でない場合は`nil` を返し、その種別である場合は`t` を返し、その種別かもしれないときは数値を返さなくてはなりません。高い数値は高い可能性を意味し、低い数値は低い可能性を意味します。‘0’ は正しい値の中でもっとも低い数値です。

7.6.4 メールからニュースへのゲートウェイ

あなたのローカルの`nntp` サーバーが何らかの理由で投稿を許可していなくても、数ある mail-to-news ゲートウェイを使って投稿することができます。`nngateway` バックエンドはこのインターフェースを提供します。

このバックエンドからは何も読み出せないことに注意してください---これは投稿するためだけに使われます。

以下はサーバー変数です。

`nngateway-address`
: これが mail-to-news ゲートウェイのアドレスです。

`nngateway-header-transformation`
: ニュースヘッダーは、mail-to-news ゲートウェイが受け付けられるように、何か奇妙なやり方で変形しておかなければならないことがしばしばです。この変数はどんな変形処理が呼び出されるべきかを指示するもので、ディフォルトでは`nngateway-simple-header-transformation` になります。その関数は変形しようとするヘッダーの領域だけに狭められたバッファーで、ゲートウェイのアドレスを一つの引数として呼び出されます。

  ディフォルトの関数は、単に`Newsgroups` ヘッダーとゲートウェイのアドレスに基づいた新しい`To` ヘッダーを挿入します。例えば、以下のような`Newsgroups` ヘッダーを持つ記事には、

  ```
  Newsgroups: alt.religion.emacs
  ```

  次のような`To` ヘッダーが挿入されます。

  ```
  To: alt-religion-emacs@GATEWAY
  ```

  以下の関数が用意されています:

  `nngateway-simple-header-transformation`
  : *newsgroup*@`nngateway-address` のような`To` ヘッダーを作ります。

  `nngateway-mail2news-header-transformation`
  : `nngateway-address` のような`To` ヘッダーを作ります。

例です:

```
(setq gnus-post-method
      '(nngateway
        "mail2news@replay.com"
        (nngateway-header-transformation
         nngateway-mail2news-header-transformation)))
```

したがってこれを使うには、単にこんな風にすれば良いでしょう:

```
(setq gnus-post-method '(nngateway "GATEWAY.ADDRESS"))
```

7.6.5 空っぽのバックエンド

実際は必要が無いのにどこかにバックエンドを設定しなければならない場合に、`nnnil` は代用として使うことができるバックエンドです。典型的な例は、第一の選択方法を必要としないが第二の(secondary) 選択方法だけを使いたい場合です:

```
(setq gnus-select-method '(nnnil ""))
(setq gnus-secondary-select-methods
      '((nnimap "foo")
        (nnml "")))
```

7.7 仮想グループ

Gnus は、多くのソースからの記事を組み合わせたり、複数のグループ全体を仮想グループに合併させることができます。

7.7.1 選択グループ

Gnus は収集したメッセージで構成される仮想グループを作成するための*nnselect* メソッドを提供します。それらのメッセージが複数のサーバーとバックエンドにまたがるグループからのものである場合でさえ動作するものです。ほとんどの場合、それらの仮想グループは他のグループと同じように動作し、メッセージはスレッド化、印の付加、移動、削除、コピーなどが可能です。グループは一時的または永続的です。グループは`gnus-group-make-group` によって作られるか、または`gnus-group-browse-foreign-server` によって外部グループとして閲覧されるものになります。

nnselect グループを使う上での要点は、そこに含めるメッセージを指定することです。それぞれの nnselect グループはグループパラメーター`nnselect-specs` を持ちます。それは関数`nnselect-function` と(もしあれば) 引数`nnselect-args` の 2 つの要素を持つ連想リスト(alist) です。

関数`nnselect-function` はベクトルを返さなければなりません。このベクトルのそれぞれの要素は、1 つのメッセージに対応する 3 つの要素のベクトルです。3 つの要素は完全なグループ名、メッセージ番号、および付加的なソートに使うことができるスコアです。スコアの値は任意で、直接 nnselect メソッドで使われるものではないので、たとえばすべて 100 に設定することができます。

例です:

```
(nnselect-specs
 (nnselect-function . identity)
 (nnselect-args
  . [["nnimap+work:mail" 595 100]
     ["nnimap+home:sent" 223 100]
     ["nntp+news.gmane.io:gmane.emacs.gnus.general" 23666 100]]))
```

関数は`identity` で、引数は仮想グループに含めるメッセージそのもののリストです。

あるいは、検索結果からグループを作成することもできます:

```
(nnselect-specs
 (nnselect-function . gnus-search-run-query)
 (nnselect-args
  (search-query-spec
   (query . "mark:flag"))
  (search-group-spec
   ("nnimap:home")
   ("nnimap:work"))))
```

これは、"home" と"work" という 2 つの IMAP サーバー上のすべてのグループからの、すべてのフラグ付きメッセージを含むグループを作ります。

そして最後の例です。これは、検索を実行し、特定のグループから最近受信したすべてのメッセージを見つける関数です:

```
(defun my-recent-email (args)
  (let ((query-spec
```

```
            (list
             (cons 'query
                   (format-time-string "SENTSINCE %d-%b-%Y"
                                       (time-subtract nil
                                                      (days-to-time (car args)))))
             (cons 'criteria "")))
           (group-spec (cadr args)))
      (gnus-search-run-query (list (cons 'search-query-spec query-spec)
                                   (cons 'search-group-spec group-spec)))))
```

すると以下のnnselect-specs は:

```
(nnselect-specs
 (nnselect-function . my-recent-email)
 (nnselect-args . (7 (("nnimap:home") ("nnimap:work")))))
```

過去 7 日間に受信した home サーバーと work サーバー上のすべてのメッセージで構成されるグループを提供します。

nnselect-function を実行して nnselect グループが選択するものを更新しようとすると、完了するまでに時間がかかるかもしれません。そのためgnus-group-get-new-news を実行しても、ディフォルトでは nnselect グループは更新されない場合があります。関数の実行に時間がかかりすぎない場合は、nnselect-rescan のnil ではないグループパラメータは自動更新を可能にします。gnus-group-get-new-news-this-group を通じて手動で更新することは、常に可能です。

ディフォルトでは、選択されたものを圧縮したバージョンが他のグループの情報とともに newsrc に(永続グループ用に) 保存されます。これが望ましくない場合(たとえば選択範囲が非常に長くて圧縮が十分になされない場合) は、nnselect-always-regenerate をグループパラメーターにしてnil ではない値を使うと、リストは保存されなくなる代わりに、必要になるたびにリストが再生成されるようになります。もっと柔軟性が必要な場合はnnselect-get-artlist-override-function およびnnselect-store-artlist-override-function を、記事のリストを取得して保存する関数に設定することができます。

Gnus には、さまざまなバックエンドを検索するためのエンジンが含まれています。それぞれの検索エンジンの細部は異なりますが、検索の結果は常に nnselect メソッドで使われる種類のベクトルであり、問い合わせの結果は通常 nnselect グループを使って表示されます。実際のところ、標準の検索関数であるgnus-group-read-ephemeral-search-group は、適切な検索の問い合わせとしてのnnselect-specs を持つ、一時的な nnselect グループを作成するだけです。

7.7.2 合併グループ

nnvirtual グループは、実は複数のグループを寄せ集めたものに過ぎません。

例えば、小さなグループをたくさん読むのが嫌になってきたら、それらを一つの大きなグループに入れて、嫌になるくらい巨大で手に負えないグループを読むことができます。これはコンピューティングの醍醐味だね!

選択方法としてnnvirtual を指定してください。アドレスは、それを構成するグループに合致する正規表現です。

仮想グループで付けられたすべての印は、その構成要素のグループの記事にくっつけられます。つまり、仮想グループで記事に可視記事の印を付けると、その記事はもともとの構成要素のグループでも可視記事になります。(そして逆も成り立ちます---構成

要素のグループで付けた印は、仮想グループでも表示されます。) 空の仮想グループを作るには、グループバッファーで*G V* (gnus-group-make-empty-virtual) を実行し、*M-e* (gnus-group-edit-group-method) で選択方法の正規表現を編集してください。

これが、Andrea Dworkin に関するすべてのニュースグループを、一つの巨大でシアワセなニュースグループにまとめるnnvirtual 選択方法の例です:

```
(nnvirtual "^alt\\.fan\\.andrea-dworkin$\\|^rec\\.dworkin.*")
```

構成要素のグループは基本グループでも外部グループでも構いません。すべて問題無く動くはずですが、もしあなたのコンピューターが爆発でもしてしまったら、それはたぶん私が悪いんでしょうね。

利用者が(訳注: 記事を投稿する人たちが) Distribution ヘッダーを使って配布範囲を制限している場合に、同じグループを複数のサーバーから寄せ集めることは、本当にうまい考えかもしれません。'soc.motss' を日本のサーバーとノルウェーのサーバーの両方から読みたければ、グループの正規表現として以下のものを使うことができるでしょう:

```
"^nntp\\+server\\.jp:soc\\.motss$\\|^nntp\\+server\\.no:soc\\.motss$"
```

(でもちょっと注意。*G m* でグループを作成するときは、バックスラッシュを二重に付けてはいけません。そして文字列の最初と最後の引用記号('"..."') も取り払ってください。)

これはまあ、すらすらと動作するはずです---両方のグループのすべての記事は一つのグループに入り、重複も無いはずです。スレッド表示(とその他) も通常通り動作するでしょうけれど、記事の並ぶ順序には問題があるかもしれません。日付による並べ替えが、ここでは一つの選択肢になるかもしれません(see Section 3.3 [Selecting a Group], p. 16)。

なお、ここで一つだけ制限があります---仮想グループに含まれるグループはすべて生きている(すなわち購読または非購読の) 状態でなくてはなりません。削除された(killed) グループあるいはゾンビのグループはnnvirtual グループを構成するグループになることはできません。

nnvirtual-always-rescan 変数がnil でなければ(それ、つまり非-nil がディフォルト)、nnvirtual は仮想グループに入ったときに常に未読記事を走査します。この変数がnil になっていて、仮想グループを作った後に構成要素のグループで記事を読んだ場合は、その構成要素のグループで読まれた記事は、仮想グループに現れてしまうでしょう。共通な構成要素のグループを持つ二つの仮想グループがある場合にも、この影響があります。そういう場合には、この変数をt にするべきです。さもなければ、仮想グループに入る度に、毎回その仮想グループの上でM-g を叩いても良いでしょう---これにはほぼ同様の効果があります。

nnvirtual はメールとニュースの両方のグループを構成要素のグループにすることができます。nnvirtual グループの記事に返答するときは、nnvirtual は記事の出所の構成要素のグループのバックエンドに、それがニュースのバックエンドであるかメールのバックエンドであるかを尋ねなければなりません。しかし^ をしたときには、普通は構成要素のバックエンドがこれを知るための確実な方法が無いので、その場合nnvirtual は、Gnus に記事はニュースではないバックエンドからやって来たと告げます。(単にそれが安全な側なので。)

これらの場合にメッセージバッファーで*C-c C-n* を行なうと、応答しようとしている記事からNewsgroups 行を抜き出して挿入します。

nnvirtual グループは、記事と印以外は構成要素のグループから継承しません---例えばグループパラメーターもそうなのですが、それらは継承されません。

7.8 電子メールによる日程管理

この章ではnndiary という特別なメールバックエンドと、その仲間のgnus-diary ライブラリーについて説明します。それが「特別」なのは、Gnus でメールを読むための標準の選択肢の一つであるつもりは無いからです。(メールバックエンドの選択についてはSee Section 7.4.13 [Choosing a Mail Back End], p. 187.) 代わりに、特別な方法であなたのメールのいくつか を扱う、すなわちこれはリマインダー(予定を思い出させるもの) として使われます。

典型的な筋書きは、こうです。

- あなたはアンディ・マクドウェルかブルース・ウィリス(あなたの好みに合わせて、どちらかを選んでください) と、一ヶ月後にデートの約束をしました。それを忘れるわけにはいきません。
- そこで、自分宛てにリマインダーのメッセージを(本当に毎日一通) 送ることにしました。
- あなたはそのことをすっかり忘れて、いつもどおりに新しいメールを取り込んで読み続けます。
- デートの日が近付いてくると、グループバッファーで*g* をタイプしたときに、ときどきあなたの予定を思い出させるために、あたかも新着で未読のように、メッセージが再びポップアップするでしょう。
- これが含まれている「新しい」メッセージたちを読んでください、そして、再びあなたが過ごす夜を夢見てください。
- いったんデートが終わると(実際にはディナーのすぐ後で寝入ってしまったとしても)、期限切れ消去の印が付いていれば、メッセージは自動的に消去されます。

Gnus Diary バックエンドは、(常に取り消されることが無い) 定期的な予定を、几帳面な人たちと同じように扱う能力を持っていて、本当のメールバックエンドのように動作し、いろんなやり方で設定することができます。このすべてが、以下の各章で説明されています。

7.8.1 NNDiary バックエンド

nndiary はnnml (see Section 7.4.13.3 [Mail Spool], p. 188) にとてもよく似ているバックエンドです。現にそれはnnml とnndraft を合わせたものに見えるでしょう。nnml をご存知ならば、あなたはすでにnndiary がメッセージを格納する仕組み(一通あたり一つのファイル、一グループあたり一つのディレクトリー) に精通しています。

何はさておき、nndiary をちゃんと動作させるには、一つの要件があります: Gnus のグループの日付の機能を使わなければなりません。それがどういうふうに行なわれるかはSection 3.18.3 [Group Timestamp], p. 45, を見てください。

7.8.1.1 日程メッセージ

七つの特別なヘッダーが必須であること以外、nndiary のメッセージはまったく普通のものです。それらのヘッダーはX-Diary-<something> の様式で表され、<something> の部分はMinute, Hour, Dom, Month, Year, Time-Zone およびDow のうちの一つです。Dom は「日(Day of Month)」を、Dow は「曜日(Day ofWeek)」を意味します。これらのヘッダーはcrontab の設定のように働いて、予定日を定義します。

- Time-Zone のもの以外のすべてのヘッダーについて、ヘッダーの値は星印(可能なすべての値を意味します) かコンマで区切られたフィールドのリストです。
- フィールドは整数か範囲のどちらかです。

- 範囲とは、ダッシュ(-) で区切られた二つの整数です。
- 可能な値は、それぞれMinute には 0--59、Hour には 0--23、Dom には 1--31、Month には 1--12、Year には 1971 より大きい値、そしてDow には 0--6 (0 が日曜日) です。
- 特別な場合として、Dom またはDow のどちらか一方における星印は「可能なすべての値」ではなく、「もう一方のフィールドだけを使う」意味になります。両方とも星印にした場合は、どちらを使っても同じ結果になることに注意してください。
- Time-Zone ヘッダーは、値を一つしか持てない(例えばGMT) 点で特別です。星印は「可能なすべての値」ではなく(それは意味をなさないので)、「現在のローカルなタイムゾーン」を意味します。ここではたいてい星印を使うでしょう。しかし、利用できるタイムゾーンの値については、変数nndiary-headers を見てください。

1999 年から 2010 年までの毎週月曜日と毎月の一日の 12:00, 20:00, 21:00, 22:00, 23:00 および 24:00 を設定するために、メッセージに加える日程ヘッダーの具体例です(その時何をしたら良いかは、自分で考えてください):

```
X-Diary-Minute: 0
X-Diary-Hour: 12, 20-24
X-Diary-Dom: 1
X-Diary-Month: *
X-Diary-Year: 1999-2010
X-Diary-Dow: 1
X-Diary-Time-Zone: *
```

7.8.1.2 NNDiary を動かす

nndiary には二つの動作モードがあります。一つはディフォルトの「伝統型(traditional)」、もう一つは「自律型(autonomous)」です。伝統型のモードでは、nndiary はそれ自身が新着メールを取得することはありません。日程メッセージとして扱うために、あなたはメールを基本のメールバックエンドから nndiary グループに、移動(*B m*) またはコピー(*B c*) しなければなりません。自律型のモードでは、nndiary はそれ自身のメールを取ってきて、基本のメールバックエンドとは独立してそれを扱います。

本質的に Gnus は、同時に複数の「マスター」メールバックエンドを許容するようには設計されていなことに注意すべきです。しかしnndiary では、これは意味をなします。あなたは本当に、日程メッセージを日程グループに直接送って、それらを受け取りたいのです。そこでnndiary は、まさに「二番目の第一メールバックエンド」をサポートします(私が知っている限り、それはこの機能を提供する唯一のバックエンドです)。しかしながら制約があって(いつの日にか解消することを願いますが)、自律型のモードでは再スプールができません。

自律型のモードでnndiary を使うためには、いくつかのことをやってもらわなければなりません:

- 新着メールをnndiary が自分で取り込めるようにします。以下の行を~/.gnus.el ファイルに記入してください:

```
(setq nndiary-get-new-mail t)
```

- 日程メッセージ(X-Diary-* ヘッダーを含んでいる) が、Gnus がそれらを処理する前に専用のフォルダーに分配されるように、準備を行なわなければなりません。繰り返しますが、Gnus が複数の第一メールバックエンドを適切に扱うことが(まだ?) できないので、これが必要です。別々のソースからそれらのメッセージを取り込むことによって、この欠点はある程度補われます。

日程ファイルを~/.nndiary (これがディフォルトのnndiary のメールソース・ファイルです) に格納するための procmailrc の項目の例です:

```
:0 HD :
* ^X-Diary
.nndiary
```

いったんこれを実施したら、日程メールの取り込みと分割の処理に影響する、以下の二つのオプションをカスタマイズする必要があるでしょう:

nndiary-mail-sources [変数]
: 標準のmail-sources 変数の、日程用に特化した代替品です。同じ構文(syntax) を使い、ディフォルトは(file :path "~/.nndiary") です。

nndiary-split-methods [変数]
: 標準のnnmail-split-methods 変数の、日程用に特化した代替品です。同じ構文(syntax) を使います。

最終的にはgnus-secondary-select-methods に、恒久的なnndiary 仮想サーバー((nndiary "diary") が行なうべきであるようなもの) を追加しても良いでしょう。

うまくいけば、Gnus を再起動すると、ほとんどすべて(nndiary.el の TODO の項を参照) が期待通りに動作するでしょう。自律型のモードでは、*g* や*M-g* をグループバッファーでタイプすれば新しい日程メールをも取り込んで、日程用に特化した規則に従ってそれらを分割するし、*F* は新しい日程グループを見つけてくれる、など。

7.8.1.3 NNDiary のカスタマイズ

さあnndiary が立ち上がって動作しています。それをカスタマイズするときが来ました。カスタマイズするためのグループはnndiary です(へえー)。どのオプションをカスタマイズし倒したいかを見つけるために、それに目を通してください。あなたが変更したいのは、おそらく以下のたった二つの変数でしょう:

nndiary-reminders [変数]
: 予定を思い出させてもらいたい時刻のリスト(例えば三週間前、それから二日前、それから一時間前、そしてそのとき) です。「思い出させてもらう」の意味は、新着メールを取り込んだときに、日程メッセージが真新しく未読になって、ポップアップすることであることを思い出してください。

nndiary-week-starts-on-monday [変数]
: 読んで字の如し。さもなくば日曜日が仮定されます(それがディフォルトです)。

7.8.2 Gnus Diary ライブラリー

nndiary を手作業で使うこと(ヘッダーを手で書くことなど) は、いささかうんざりします。幸いnndiary の上位階層に書かれたgnus-diary というライブラリーがあって、たくさんの便利なことをやってくれます。

それを使うためには、以下の行を~/.gnus.el ファイルに加えてください:

```
(require 'gnus-diary)
```

さらに、どんなgnus-user-format-function-[d|D] (see Section 4.1.1 [Summary Buffer Lines], p. 47) も、使ってはいけません。gnus-diary はそれらの両方を提供します(あなたがそれらを使っていたら、すみません)。

7.8.2.1 日程の概略行仕様

標準の概略行仕様(通常'From Joe: Subject' のようなもの) で日程メッセージを表示するのは、まったく役に立ちません。たいていはあなたがメッセージを書いた人で、おおかた予定の日付を見たいと思っているでしょう。

gnus-diary は、概略行仕様で使う二つの追加の利用者定義の書法仕様を提供します。D は次の予定が生じるときのための整形された時刻表示(例えば"Sat, Sep 22 01, 12:00") を表すのに対して、d は次の予定が生じるまでのおおよその残り時間(例えば"in 6 months, 1 week") を表します。

ジョーの誕生日が、概略行にどう表示されるかの例です(定期的な予定を指定すると消されないことを除いて、メッセージが期限切れ消去可能であることに気を付けてください):

```
E  Sat, Sep 22 01, 12:00: Joe's birthday (in 6 months, 1 week)
```

上記のようなものを得るために、普段だったら、あなたは以下の行を日程グループのパラメーターに加えようとするでしょう:

```
(gnus-summary-line-format "%U%R%z %uD: %(%s%) (%ud)\n")
```

しかしgnus-diary はそれを自動で行ないます(see Section 7.8.2.4 [Diary Group Parameters], p. 216)。それでもあなたは、以下のユーザー・オプション群で提供される概略行仕様を、カスタマイズすることができます:

gnus-diary-summary-line-format [変数]
: 日程グループのために使われる概略行仕様を定義します(see Section 4.1.1 [Summary Buffer Lines], p. 47)。gnus-diary はそれを、日程グループのパラメーターを自動で更新するために使います。

gnus-diary-time-format [変数]
: 日程の概略バッファーに日付を表示するための書法仕様を定義します。これは利用者定義の書法仕様D で使われます。詳細は変数の説明文を見てください。

gnus-diary-delay-format-function [変数]
: 日程の概略バッファーに遅延(残り時間) を表示するための整形関数を定義します。これは利用者定義の書法仕様d で使われます。現在は英語とフランス語のための組み込み関数があり、自分で定義することもできます。詳細は変数の説明文を見てください。

7.8.2.2 日程記事の並べ替え

gnus-diary は並べ替え(see Section 4.10 [Sorting the Summary Buffer], p. 76) のためにgnus-summary-sort-by-schedule、gnus-thread-sort-by-schedule およびgnus-article-sort-by-schedule という新しい関数を提供します。これらの関数によって、最も近い予定から最も遠い方まで、日程の概略バッファーを整理することができます。

gnus-diary は自動的に概略バッファーの「並べ替え(sort)」メニューにgnus-summary-sort-by-schedule を組み込み、他の二つを第一次の(ゆえにディフォルトの) 並べ替え関数として、グループパラメーター(see Section 7.8.2.4 [Diary Group Parameters], p. 216) に登録します。

7.8.2.3 日程ヘッダーの生成

gnus-diary は、X-Diary-* ヘッダーの取り扱いを補佐するために、gnus-diary-check-message という関数を提供します。この関数は、現在のメッセージがすべての必要な日程ヘッダーを確実に含むようにして、必要ならば値を入力するか修正することを要求します。

記事を日程グループに移動またはコピーすることによって自動的にそれが発動されるようにするために、この関数はnndiary バックエンドのフックとして組み入れられています。それはさらに、通常のメールを日程用のものに変換する操作を簡単にするために、message-mode とarticle-edit-mode において*C-c C-f d* キーとして設定もされています。

接頭引数を伴ってこの関数を呼ぶと、それらがあるか、正しいかどうかとは無関係に、日程ヘッダーの入力を強制します。そうやって、例えばすでに正しく設定されたメッセージの日程を、とても簡単に変更することができます。

7.8.2.4 日程グループのパラメーター

新しい日程グループを作るか、またはそれを開くと、gnus-diary は自動的にグループパラメーターを検査し、必要なら概略行仕様を日程用に特化した値に設定し、日程用の並べ替え関数を組み込み、さらにそのグループの投稿様式(posting-style) に種々のX-Diary-* ヘッダーを加えます。そして、日程メッセージを送るのは、もっと簡単です。メッセージを用意するために、日程グループで*C-u a* か*C-u m* を使うことによって、これらのヘッダーが自動的に挿入されるので(まだ適切な値で満たされていませんが)。

nndiary は本当の メールバックエンドです。本当にあなたは本当の日程メッセージを本当に送ります。これは、日程メッセージを送ることによって、誰にでも(彼らが Gnus とnndiary を使っているのならば) 予定を伝えることができることをも意味します。

7.9 Gnus の切り離し

いにしえの時代(およそ 1988 年 2 月頃)、人々はニュースリーダーをネットワークに常時接続した大きなマシンで走らせていました。ニュースの配送はニュースサーバーによって取り扱われ、すべてのニュースリーダーがすべきことはニュースを読むことであったのです。信じられないかもしれませんが。

今日では多くの人々は自宅でニュースやメールを読み、ネットワークに接続するためにモデムの類を使います。電話代の請求書が莫大なものに上らないように、すべてのニュースとメールをすすり込んで電話を切り、数時間かけて読んでから送りたい返信をすべて送信する、という手段を持つことは良いことでしょう。あとはこの手順を繰り返すのです。(訳注: この章の前身は 1997 年頃に書かれました。)

もちろん、これを行なうためにニュースサーバーを使うこともできます。私はinn をslurp, pop, sendmail と一緒にここ数年使ってきましたが、しかしこれは退屈な仕事です。もしあるマシン上でニュースを読む人があなたしかいなければ、ニュースサーバーの機能をニュースリーダーに任せるようにすることは理にかなっています。

Gnus を「オフライン」のニュースリーダーとして仕立てるのは極めて簡単です。実際、エージェントは今やディフォルトで有効になっている(see Section 7.9.11 [Agent Variables], p. 228) ので、あなたは何も設定する必要が無いのです。

もちろん、これをそんなふうに使うには、いくつか新しい命令を覚えなくてはなりません。

7.9.1 エージェントの基礎

まず、いくつかの用語を片付けておきましょう。

ネットワークとの接続を切っているとき(かつエージェントにそのことを知らせてあるとき)、Gnus エージェントは*unplugged* です、と言います。ネットワークとの接続が復活したら(かつ Gnus がそのことを知っていれば)、エージェントは*plugged* です。

「ローカル」マシンとは、あなたがそこで作業しているもので、継続的にネットワークに接続されているわけではありません。

「ダウンロード」とは、あなたのローカルマシンに、何かをネットワークから取ってくることを意味します。「アップロード」はその逆をすることです。

ご存知のように Gnus はあなたがドジを踏むすべての機会を提供します。それを柔軟性と言う人もいます。さらに Gnus は大いにカスタマイズ可能で、それは利用者が、Gnus がどのように動作するかについて発言権を持っていることを意味します。他のニュースリーダーは有無を言わずあなたにドジを踏ませるかもしれませんが、Gnus ではあなたに選択権があります!

Gnus は実際には plugged または unplugged のどちらの状態にもありません。もっと正確に言えば、サーバーごとにそれぞれの状態を持ちます。これは、いくつかのサーバーが unplugged でも、他のサーバーは plugged になることができるということです。さらに、エージェントがいくつかのサーバーをまとめて無視する(それらを常に plugged になっているように見せかける) ようにもできます。

さて、エージェントを unplugged にしたのに Gnus がネットに接続しているのを疑問に思ったら、行なうべき次のステップはサーバーがすべてエージェント化されているかどうかを確かめることです。エージェント化されていないサーバーがあったら、あなたは犯人を見つけたのです。

もう一つは「オフライン」という状態です。サーバーはときどき接続できなくなります。Gnus がこのことに気付くと、そのサーバーをオフラインの状態に切り換えても良いかどうかを尋ねます。Yes と答えたならば(オンラインに戻して良いかと Gnus が尋ねた場合以外は)、サーバーはいくらか unplugged だったときのように振る舞います。

エージェントを使った典型的な Gnus の対話操作を見てみましょう:

- Gnus を`gnus-unplugged` で起動します。これは unplugged で Gnus エージェントを立ち上げます。このモードでは、すでに取得しているニュース記事はすべて読むことができます。
- 次に、新しいニュースが到着しているかどうかを調べることにします。マシンをネットワークに(PPP か何かを使って) 接続してから Gnus を*plugged* にするために`J j` を叩き、いつものように新着メールを検査するために`g` を使います。Gnus エージェントが unplugged になっているときに新着メールを検査するには、Section 7.4.4.1 [Mail Source Specifiers], p. 166, を参照してください。
- そうすれば、直ちに新しいニュースを読むこともできるし、ニュースをローカルマシンにダウンロードすることもできます。後者を実行したいときは、`g` を押して新しいニュースがあるかどうかを検査し、次に`J s` ですべてのグループのすべての適格な(訳注: あなたが指定した条件に合致する) 記事を取得します。(どの記事をダウンロードしたいかを Gnus に指示するにはSection 7.9.2 [Agent Categories], p. 218, を参照してください。)
- 記事を取得した後で`J j` を押し、Gnus を再び unplugged にして、PPP の接続(か何か)を閉じます。その後でニュースをオフラインで読みます。
- そして第二ステップに戻ります。

エージェントを初めて使うときは(またはそのくらいの時期に)、以下のいくつかの作業をしなければなりません。

- どのサーバーをエージェントで面倒を見るかを決めます。メールのバックエンドをエージェントに面倒を見させるのはおそらく無意味でしょう。サーバーバッファーに移

動し(グループバッファーで^)、エージェントに扱って欲しいサーバー(複数可) で`J a`を押す(see Section 7.9.3.3 [Server Agent Commands], p. 225) か、またはエージェントに扱って欲しくないのに自動的に追加されたサーバーで`J r` を押します。ディフォルトではどのタイプのサーバーもエージェント化されません。

- ダウンロード方針を決定します。あなたの方針を実装するために、エージェント分類、トピックパラメーター、グループパラメーターのどれを使うかをいったん決めてしまえば、これはかなり簡単です。あなたが Gnus の初心者ならば、たぶん分類で始めるのが最良でしょう、See Section 7.9.2 [Agent Categories], p. 218.

  トピックパラメーター(see Section 3.16.5 [Topic Parameters], p. 40) とエージェント分類(see Section 7.9.2 [Agent Categories], p. 218) の両方とも、多数のグループに適用する方針の設定を用意しています。どれを使うかは完全にあなたの責任です。両方を混ぜて使う場合は、トピックパラメーターは分類を無効にすることを考慮に入れなければならないでしょう。あなたの方針にそぐわない少数のグループがあるのならば、それらの設定を変更するためにグループパラメーター(see Section 3.10 [Group Parameters], p. 23) を使うことができます。

- ええと...、以上です。

7.9.2 エージェント分類

ニュースを配送する機構をニュースリーダーに統合する主要な理由の一つは、どの記事をダウンロードするかについて、もっと強力に制御できるようにすることです。莫大な量の記事をダウンロードすることにあまり意味はなく、それらを読んでもあまり面白くないことが分かるだけです。何をダウンロードするかの選択ではもう少し保守的になって、その記事がやっぱり面白そうだとわかったら、主動でダウンロードするための印を付ける方がすぐれています。

何をダウンロードするかを制御するためのより有効な方法の一つは、分類(*category*) を作成して、その分類にいくつか(または全部) のグループを割り当てることです。どんな分類にも属さないグループは「ディフォルト」の分類に属します。Gnus は分類の作成と管理のための独自のバッファーを持っています。

もしそうしたければ、グループパラメーター(see Section 3.10 [Group Parameters], p. 23) とトピックパラメーター(see Section 3.16.5 [Topic Parameters], p. 40) を、エージェントを制御する代替手段に使うことができます。実際に違うのは、グループとトピックパラメーターが何でもかんでも(kitchen sink) 含むのに対して、分類はエージェントに特化している(したがってあまり学ばなくても良い) ということだけです。

エージェントパラメーターは複数の違う場所で設定することができるので、どのソースが信用できるかを決めるための規則を設けました。この規則は、パラメーターのソースが次の順序で調べられることを定めます: グループパラメーター、トピックパラメーター、エージェント分類、そして最後はカスタマイズできる変数群です。したがって、広い範囲で動作を起こさせるためにこれらのソースをすべて混合することができます。どこに設定を置いたのかを忘れてしまったからといって、私を責めないでくださいよ。

7.9.2.1 分類の文法

分類は、名前、その分類に属するグループのリスト、およびカスタマイズ可能な変数よりも優先される多くの任意なパラメーターから成ります。エージェントパラメーターの完全なリストを以下に示します。

`agent-groups`
: この分類にあるグループのリスト。

`agent-predicate`
: (通常) どの記事をダウンロードするのが適当かという大まかな輪郭を与える述語。そして

`agent-score`
: (通常) どの記事をダウンロードするかを決めるときのよりきめの細かいスコア規則。(このダウンロードスコア(*download score*) は通常のスコアとは必ずしも関係が無いことに注意してください。)

`agent-enable-expiration`
: このグループの古い記事をエージェントが期限切れ消去すべきかどうかを示すブール変数。大抵のグループはディスク空間を浪費しないために期限切れ消去されるべきです。いや、実際には gnus.* 階層は期限切れ消去されるべきではないグループだけを含んでいると言っても、たぶん差し支えありません。

`agent-days-until-old`
: 既読の記事を期限切れ消去しても差し支えないことを判断する前に、エージェントが待っているべき日数を示す整数。

`agent-low-score`
: `gnus-agent-low-score` よりも優先される整数。

`agent-high-score`
: `gnus-agent-high-score` よりも優先される整数。

`agent-short-article`
: `gnus-agent-short-article` よりも優先される整数。

`agent-long-article`
: `gnus-agent-long-article` よりも優先される整数。

`agent-enable-undownloaded-faces`
: ダウンロードされていない記事を`gnus-summary-*-undownloaded-face` のフェース群を使って概略バッファーに表示すべきかどうかを示すシンボル。`nil` 以外ならどんなシンボルでも、ダウンロードされていない記事用のフェースを使うようになります。

いったん分類が作られたら、分類の名前を変えることはできません。

それぞれの分類は、その分類の排他的な(他の分類には無い) メンバーであるグループのリストを維持します。排他規則は自動的に執行され、新しい分類にグループを追加すると、それは古い分類から自動的に取り除かれます。

述語の一番単純な形式は`true` や`false` のような単独の述語からなります。これらの二つはそれぞれ、すべての可能な記事をダウンロードするか、まったく何もしないか、です。これらの二つの特別な述語の場合は、追加のスコア規則は不要です。

`high` や`low` という述語は下で説明されているように、`gnus-agent-high-score` と`gnus-agent-low-score` との記事のスコアとの関係により記事をダウンロードします。

何をもってダウンロードすることが適格だと見なされるかについて、さらに細かい制御を得るために、述語は論理演算子が間に散りばめられた述語の組み合わせからなることができます。

おそらくいくつかの例が必要でしょう。

以下は簡単な述語です。(ディフォルトの述語は`false` で、他のどの分類にも属さないすべてのグループに対して使用されます。)

```
short
```

とっても簡単でしょ? この述語は、記事が短い(「短い」ことを意味する何らかの価値がある) 場合に限り真になります。

これはもっと複雑な述語です:

```
(or high
    (and
     (not low)
     (not long)))
```

この意味は、高いスコアを持っているか、あるいはスコアが低くなくてかつ長くない、という記事をダウンロードする、ということです。様子はわかりましたね。

使ってもよい論理演算子は`or`, `and` および`not` です。(もし使いたければ、より“C”風の演算子‘`|`’, `&`, `!` を代りに使うことができます。)

以下の述語があらかじめ定義されていますが、これらのどれもあなたのやりたいことに適さなければ、自分で独自のものを書くこともできます。

それぞれのこれらの述語を評価するとき、名前が付けられた定数は、適切なパラメーターを与えて`gnus-agent-find-parameter` を呼ぶことによって決定される値に束縛されます。例えば gnus-agent-short-article は`(gnus-agent-find-parameter group 'agent-short-article)` に束縛されます。これは、あなたの分類で述語を指定してから、その述語を個々のグループについて調整できることを意味します。

`short` 記事が`gnus-agent-short-article` の行数より短かければ真です。ディフォルトは 100 です。

`long` 記事が`gnus-agent-long-article` の行数より長ければ真です。ディフォルトは 200 です。

`low` 記事のダウンロードスコアが`gnus-agent-low-score` の値より小さければ真です。ディフォルトは 0 です。

`high` 記事のダウンロードスコアが`gnus-agent-high-score` の値より大きければ真です。ディフォルトは 0 です。

`spam` Gnus エージェントがその記事を spam だと推測した場合に真です。この検出法は今後変更されるかもしれませんが。現時点では、これはチェックサムを計算し、記事が一致するかどうかを調べているだけです。

`true` 常に真です。

`false` 常に偽です。

独自の述語関数を作成したければ、このことを知っておかなければなりません: 関数は引数無しで呼び出されますが、`gnus-headers` と`gnus-score` という動的な変数が有意な値に束縛されるということを。

例えば、一定の日数以上前に投稿された記事(例えば`gnus-agent-expire-days` の日数以上前に投稿されたもの) をダウンロードしないと決断することもできます。その場合、以下のような関数を書いて、

```
(defun my-article-old-p ()
```

```
"Say whether an article is old."
(< (time-to-days (date-to-time (mail-header-date gnus-headers)))
   (- (time-to-days nil) gnus-agent-expire-days)))
```

そして述語はこのように定義すれば良いでしょう:

```
(not my-article-old-p)
```

もしくは~/.gnus.el か何かで、あらかじめ定義されているgnus-category-predicate-list の値に、自分の述語を追加することもできます。

```
(require 'gnus-agent)
(setq gnus-category-predicate-alist
      (append gnus-category-predicate-alist
              '((old . my-article-old-p))))
```

この場合は、次のように述語を指定するだけです:

```
(not old)
```

上のようなものを使うときは、世の中には正しく設定されていないシステム/メーラーがあり、記事の日付はいつ投稿されたかを常に確実に示すわけではないことを知っていてください。困ったことに、それを少しも気にかけない人もいるんです。

上記の述語はその分類に属するすべて のグループに適用されます。しかし、分類中の個々のグループのための特定の述語を設定したかったり、単に不精を決め込んで新しい分類を設定したくないのならば、グループの個々の述語を次のようにグループパラメーターに入れることができます:

```
(agent-predicate . short)
```

これは agent 分類のディフォルトと等価なグループ/トピックパラメーターです。このように単一の語で述語を指定するときは、agent-predicate の設定値はドット対で表記しなければならないことに注意してください。

上のものと等価な長い方の例はこうなるでしょう:

```
(agent-predicate or high (and (not low) (not long)))
```

述語の値がドット対で表記されていなくて、その値はリストだと仮定されるので、分類の設定で要求される外側の括弧が、ここでは入れられません。

さて、ダウンロードスコアの文法は通常のスコアファイルの文法と同じですが、例外があります。記事そのものを実際に調べる必要がある要素は厳禁です。つまり、以下のヘッダーだけがスコア付けできるということです: Subject, From, Date, Message-ID, References, Chars, Lines およびXref。

述語の場合のように、**ダウンロードスコア規則** の設定は、それをグループに関して使う限りは、そこのすべてのグループに適用できるものならば分類の定義、グループに特有ならばグループパラメーター、のどちらかにできます。

これら両方の場所で、**ダウンロードスコア規則** は以下の三つの形式の一つを取ることができます:

1. スコア規則

   上で書かれているように、スコア付けキーワードの一部分しか使えないことを除けば、これは普通の Gnus スコアファイルの構文と同じです。

   例:

   - 分類指定

     ```
     (("from"
     ```

```
        ("Lars Ingebrigtsen" 1000000 nil s))
  ("lines"
        (500 -100 nil <)))
```

- グループ/トピックパラメーター指定

```
(agent-score ("from"
                    ("Lars Ingebrigtsen" 1000000 nil s))
             ("lines"
                    (500 -100 nil <)))
```

ここでも一番外側の括弧が省略されていることに注意してください。

2. エージェントスコアファイル

これらのスコアファイルは、上で述べられている使用可能なスコア付けキーワードだけ を含んでいなければなりません。

例:

- 分類指定

```
("~/News/agent.SCORE")
```

または、もしかすると

```
("~/News/agent.SCORE" "~/News/agent.group.SCORE")
```

- グループパラメーター指定

```
(agent-score "~/News/agent.SCORE")
```

ここでも前述のように、追加のスコアファイルを指定することができます。括弧について言わなければいけませんか?

3. **普通** のスコアファイルの使用

あるグループのためにあなたが望んだ「ダウンロード」の基準が、「読む」基準と同じならば、一つのグループのために二つのスコア規則を維持管理したいとは思わないでしょう。そういう場合は、何をダウンロードするかを決める際に、エージェントに**普通**のスコアファイルを参照させることができます。

分類の定義やグループパラメーターでこれらの指示を行なうと、エージェントはあるグループに適用することができるすべてのスコアファイルを読み込んで、使うことが許されているスコア付けキーワードの副セットではない項目を選別して取り除きます。

- 分類指定

```
file
```

- グループパラメーター指定

```
(agent-score . file)
```

7.9.2.2 分類バッファー

通常すべての分類は分類バッファーから管理します。これに(グループバッファーで*J c* 命令を使って) 初めて入ると、ディフォルトの分類だけが表示されます。

このバッファーでは以下の命令を使うことができます:

q グループバッファーに戻ります(gnus-category-exit)。

e 選択された分類のパラメーターを一括して設定するために、カスタマイズバッファーを使います(gnus-category-customize-category)。

k 現在の分類を消去します(gnus-category-kill)。

c 現在の分類を複製します(gnus-category-copy)。

a 新しい分類を追加します(gnus-category-add)。

p 現在の分類の述語を編集します(gnus-category-edit-predicate)。

g 現在の分類に属するグループのリストを編集します(gnus-category-edit-groups)。

s 現在の分類のダウンロードスコア規則を編集します(gnus-category-edit-score)。

l すべての分類を表示します(gnus-category-list)。

7.9.2.3 分類変数

gnus-category-mode-hook
: 分類バッファーで実行するフックです。

gnus-category-line-format
: 分類バッファーの行様式です(see Section 10.4 [Formatting Variables], p. 272)。有効な要素は:

  'c' 分類の名前です。

  'g' その分類に属するグループの数です。

gnus-category-mode-line-format
: 分類モード行の様式です(see Section 10.4.2 [Mode Line Formatting], p. 273)。

gnus-agent-short-article
: この値より少ない行数の記事は短いと見なします。ディフォルトは 100 です。

gnus-agent-long-article
: この値より多い行数の記事は長いと見なします。ディフォルトは 200 です。

gnus-agent-low-score
: この値より小さいスコアを持つ記事は低スコアだと見なします。ディフォルトは 0 です。

gnus-agent-high-score
: この値より大きいスコアを持つ記事は高スコアだと見なします。ディフォルトは 0 です。

gnus-agent-expire-days
: 期限切れ消去する前に、既読記事をエージェントのローカルディスクに留めておかなければならない日数(「期限切れ消去」という名前は同じですが、サーバーで期限切れ消去することではありません。単に記事のローカルな複製を消すことを意味します)。さらに理解すべき大事なことは、記事が読まれた時ではなくローカルディスクに記事が書かれた時から計数が始まるということです。ディフォルトは 7 日です。

gnus-agent-enable-expiration
: グループの記事が、ディフォルトで期限切れ消去されるか、無期限に保持されるかを決定します。ディフォルトはENABLE で、あなたが望むならば期限切

れ消去をさせないようにしなければならないことを意味します。一方、これをDISABLE に設定することができます。その場合、選択されたグループでの期限切れ消去を有効にしなければなりません。

7.9.3 エージェント命令

すべての Gnus エージェント命令は*J* サブマップにあります。*J j* (gnus-agent-toggle-plugged) 命令はすべてのモードで動作し、Gnus エージェントの plugged/unplugged 状態を切り替えます。

7.9.3.1 グループエージェント命令

J u 現在のグループの適格な(訳注: あなたが指定した条件に合致する) 記事をすべて取得します(gnus-agent-fetch-groups)。

J c エージェント分類バッファーに入ります(gnus-enter-category-buffer)。

J s 全グループの適格な(訳注: あなたが指定した条件に合致する) 記事をすべて取得します(gnus-agent-fetch-session)。

J S 順番待ち(queue) グループにある送信可能なメッセージをすべて送信します(gnus-group-send-queue)。See Section 6.7 [Drafts], p. 143.

J a 現在のグループをエージェント分類に追加します(gnus-agent-add-group)。この命令はプロセス/接頭引数の習慣を理解します(see Section 10.1 [Process/Prefix], p. 271)。

J r 現在のグループを、もし存在していれば、その分類から消去します(gnus-agent-remove-group)。この命令はプロセス/接頭引数の習慣を理解します。(see Section 10.1 [Process/Prefix], p. 271)。

J Y リモートサーバーが unplugged のときに変更されたフラグがあれば同期させます。

7.9.3.2 概略エージェント命令

J # 記事にダウンロード印を付けます(gnus-agent-mark-article)。

J M-# 記事からダウンロード印を消去します(gnus-agent-unmark-article)。

@ 記事をダウンロードするかどうかを切り替えます(gnus-agent-toggle-mark)。ディフォルトではダウンロードの印は‘%’ です。

J c キャッシュされていない、ダウンロードされていない、またはダウンロードできないすべての記事を既読にします(gnus-agent-catchup)。

J S このグループのすべての望ましい(訳注: あなたが指定した条件に合致する) 記事(see Section 7.9.2 [Agent Categories], p. 218) をダウンロードします。(gnus-agent-fetch-group)。

J s このグループのすべてのプロセス印が付いた記事をダウンロードします。(gnus-agent-summary-fetch-series)。

J u 現在のグループのダウンロード可能な記事を、すべてダウンロードします(gnus-agent-summary-fetch-group)。

7.9.3.3 サーバーエージェント命令

`J a` 現在のサーバーを Gnus エージェントで扱われるサーバーのリストに追加します(`gnus-agent-add-server`)。

`J r` 現在のサーバーを Gnus エージェントで扱われるサーバーのリストから削除します(`gnus-agent-remove-server`)。

7.9.4 エージェントの視覚効果

Unplugged のときに概略を開くと、現在エージェントに格納されているヘッダーよりも多くの記事があることを Gnus がそのグループの active (訳注: 何番から何番までの記事があるかを示す管理情報) の範囲から知っている場合には、表題が‘`[Undownloaded article #####]`’ のようになっているいくつかの記事を見るかもしれません。それらは見当たらないヘッダーのための穴埋め(placeholders) です。印を設定することは別として、それらの穴埋めの一つでできることは多くはありません。最終的に Gnus がグループのヘッダーを取って来る機会を得たときに、それらの穴埋めは実際のヘッダーで自動的に置き換えられるでしょう。気になるならば、それらの穴埋めを読み飛ばすために、概略バッファーの動作を操作することができます(`gnus-auto-goto-ignores` 参照)。

すべての人にとって明白かもしれませんが、オフラインのときに利用できるのは、plugged だった期間にエージェントに取り込まれたヘッダーと記事だけです。言い換えると「plugged だった期間に取り込むことを忘れると、オフラインのセッションを満足できるものにするには足りない」ということです。この理由のために、エージェントは概略バッファーに二つの視覚効果を加えます。これらの効果は、オフラインのときにどの記事が利用できるかをいつも知らせるために、それぞれの記事のダウンロードの状態を表示します。

第一の視覚効果は‘`%0`’ 仕様です。`gnus-summary-line-format` をカスタマイズしてこの指示子を含めると、記事のダウンロードの状態を示すために単一の文字を表示する場所が加わります。エージェントかキャッシュのどちらかに取り込まれた記事は、`gnus-downloaded-mark` (ディフォルトは‘`+`’) を表示します。それら以外のすべての記事は`gnus-undownloaded-mark` (ディフォルトは‘`-`’) を表示します。エージェント化されていないグループを開くと、空白(‘ ’) が表示されます。

第二の視覚効果はダウンロードされていないことを示すフェースです。多くの Gnus の利用者に好感と嫌悪をもたらすであろう、記事のスコアを三段階(low, normal, high) で表示するフェースがあります。問題は、フェースの選択が条件検査とフェース名のリスト(`gnus-summary-highlight` 参照) によって制御されることです。それぞれの条件は、それがリストの中に現れる順に検査されるので、後の条件よりも前の条件が優先されます。これが意味するすべては、ダウンロードされていない記事に可視記事(ticked) の印を付けても、その記事は可視記事のフェースではなくて、ダウンロードされていない記事のフェースで表示し続けられるということです。

(記事を読むたびに同じ記事をダウンロードしないようにするため、または接続時間を最小にするために) エージェントをキャッシュとして使う場合は、ダウンロードされていない記事のフェースはおそらく良い考えのように思えるでしょう。ダウンロードされた記事に対してすべての仕事(印を付ける、読む、削除する) を行なえば、いつも通常のフェースが現れるからです。しかし、NOV をキャッシュすることによってオンライン性能を改善するためにエージェントを使っている利用者にとっては(Gnus 5.10.2 以降のほとんどの利用者にとっては)、ダウンロードされていない記事のフェースが見えるかもしれないことは、まったくひどいものでしょう。これは、それらのどの記事もエージェントに取り込まれていな

いので、ダウンロードされていない記事のフェースのために、すべての普通のフェースが目立たなくなってしまうだろうという問題です。

ダウンロードされていない記事のフェースを使いたい場合は、`agent-enable-undownloaded-faces` グループパラメーターを`t` に設定して、ダウンロードされていない記事のフェースを有効にしなければなりません。このパラメーターは他のすべてのエージェントパラメーターと同様に、エージェント分類(see Section 7.9.2 [Agent Categories], p. 218)、グループトピック(see Section 3.16.5 [Topic Parameters], p. 40)、あるいは個々のグループ(see Section 3.10 [Group Parameters], p. 23) に対して設定することができます。

エージェントを使うすべての利用者に共通した一つの問題は、それがディスクの容量をいかに速く使い尽くすことができるかということです。あなたが多くのグループでエージェントを使っている場合、事実上ディスク容量を回復することはさらにもっと困難です。一つの解決手段は`gnus-group-line-format` で用意されている'`%F`' 形式です。この形式は、エージェントとキャッシュの両方で取得した記事によって占められる実際のディスク容量を表示します。どのグループが最も多い容量を使うかを知ることによって、利用者は記事を「エージェント期限切れ消去」する場合に、どこに努力を集中するべきかがわかります。

7.9.5 キャッシュとしてのエージェント

Gnus が plugged であるときに、すでにヘッダーや記事がエージェントに格納されているのならば、それらを再びダウンロードするのは効率的ではありません。そのため Gnus は通常ヘッダーを一回だけダウンロードしてエージェントに格納します。それらのヘッダーは後に概略バッファーを生成するときに、plugged か unplugged にかかわらずに使われます。ディフォルトでは記事は(それはたくさんのディスク容量を浪費するかもしれないので) エージェントにキャッシュされませんが、すでにエージェントにダウンロードした記事があるならば、Gnus はサーバーから再び記事をダウンロードせずに、手元に格納されたコピーを使います。

あなたがそう望むのであれば、plugged な期間は常にヘッダーと記事をダウンロードするように、エージェント(`gnus-agent-cache` 参照Section 7.9.11 [Agent Variables], p. 228) を設定することができます。Gnus はほとんど確かにもっと遅くなりますが、サーバーとの同期は保たれます。nntp か nnimap バックエンドを使っている場合は、たぶんこの最後の点は意味をなさないでしょう。

7.9.6 エージェント期限切れ消去

エージェントバックエンド`nnagent` は期限切れ消去を扱いません。えーと、少なくとも他のバックエンドのようにそれを扱いません。その代わりに、`gnus-agent-expire-days` の日数よりも古い既読記事をすべて消去する、特別な`gnus-agent-expire` と`gnus-agent-expire-group` 命令があります。これらはあなたがディスク容量を使い切りそうだと思ったときに、いつでも実行することができます。どちらも特に速くも効率的でもなく、それらの一つをいったん始めてしまったら(*C-g* やその他で) 中断することもあまり良いことではありません。

他の関数がエージェントをグループに同期させるために`gnus-agent-expire` を実行するかもしれないことに注意してください。

`agent-enable-expiration` というエージェントのパラメーターを、選択したグループでの期限切れ消去を抑制するために使うことができます。

`gnus-agent-expire-all` が`nil` でなければ、エージェントの期限切れ消去コマンド群はすべての記事---未読、既読、可視、保留記事を消去します。もし`nil` (これがディフォルト) であれば、既読記事のみが消去の対象となり、未読、可視、さらに保留記事は無期限に保持されます。

グループの最後の(つまり最新の) 記事は、(内部的な簿記上の理由により) 普通は期限切れ消去されません。

期限切れ消去されるはずなのに残っている記事を見つけたならば、もしかするといくつかの Gnus エージェントファイルが壊れています。起こりうる問題を修復するために、gnus-agent-regenerate とgnus-agent-regenerate-group という特別なコマンドがあります。

7.9.7 エージェントを作り直す

nnagent によって使われるローカルのデータ構造は、ある例外的な条件によっておかしくなってしまうかもしれません。これが起こるとnnagent の機能性が下がるかもしれないし、失敗しさえするかもしれません。この問題の解決策は、内部の矛盾をすべて削除することによって、ローカルのデータ構造を修復することです。

例えば、記事をエージェントにダウンロードしている間にサーバーへの接続が切れてしまう場合、ローカルのデータ構造は接続が切れる前に記事が首尾良くダウンロードされたかどうかを知りません。gnus-agent-regenerate またはgnus-agent-regenerate-group を実行すると、そのような記事を二回ダウンロードしなくても済むようにデータ構造を更新します。

gnus-agent-regenerate コマンドは、すべてのエージェント化されたグループでgnus-agent-regenerate-group を実行します。どのバッファー上でもgnus-agent-regenerate を実行することができますが、最初にすべての概略バッファーを閉じることを強く勧めます。

gnus-agent-regenerate-group コマンドは、ローカルのNOV (ヘッダー) データベースを修復するために、個々の記事のローカルなコピーを使います。その後それは、どの記事がローカルに格納されるかを記録しておくための内部データ構造を更新します。引数を与えると、エージェントの中の記事に未読の印を付けます。

7.9.8 エージェントとフラグ

エージェントは Gnus のどんなバックエンドでも、例えばサーバーにフラグ(既読(read)、可視(ticked) など) を格納する nnimap のようなものでも動作します。しかし悲しいかな、エージェントはどのバックンドがそれらのフラグを.newsrc ではなく、そのバックエンドのサーバーで維持するかを、実際には知りません。そのためエージェントは、unplugged または接続されていない間に行なったすべてのフラグへの変更を、常に自身のファイルに記録します。

再び接続すると、Gnus は変更されたすべてのフラグを検査して、それらをサーバーと同期させるかどうかを尋ねます。この挙動はgnus-agent-synchronize-flags でカスタマイズすることができます。

gnus-agent-synchronize-flags がnil だったら、エージェントは自動的にフラグを同期させることはしません。それがディフォルトのask だったら、エージェントはあなたが再接続したときにあなたが何らかの変更を行なっていたかどうかを調べて、もしそうだったら、それらを同期させたいかどうかを尋ねます。それら以外の値だった場合は、すべてのフラグは自動的に同期させられます。

再接続したときに自動でフラグを同期させたくないなら、手動でそれを行なうこともできます。これにはグループバッファーの`J Y` キーに割り当てられたgnus-agent-synchronize-flags コマンドを使ってください。

技術的注釈: すべてのローカルなフラグをサーバーに「押し込む」同期のアルゴリズムは動作しませんが、利用者によって変更されたフラグだけを変更して、サーバー側で見えるフラグを一つずつ更新することは可能です。したがって、あなたが記事の一つのフラグをセットして、そのグループを抜け出てから再度そのグループを選択してそのフラグを消せば、あなたが「同期」の操作を行なったときに、そのフラグはセットされてサーバーからは削除されます。順番待ち(queue) に入れられたフラグに関する動作は、エージェントディレクトリーにあるサーバー毎の`flags` ファイルの中で見つかるでしょう。それらはあなたがフラグを同期させたときに空になります。

7.9.9 エージェントを IMAP で使う方法

エージェントは nnimap を含む Gnus のどんなバックエンドでも動作します。しかしNNTPとIMAP にはいくつかの概念の違いがあるので、この章ではサーバーとの接続が絶たれたモードでのIMAP のクライアントとして、Gnus エージェントをより円滑に使えるようにするための、いくつかの情報を提供します。

サーバーとの接続が絶たれているときのIMAP クライアントにあなたが期待するであろういくつかの機能は、現在のエージェントには盛り込まれていません。それらは以下の通りです:

- Unplugged のときの nnimap グループへのコピーと移動。
- Unplugged のときの nnimap グループの作成と削除。

7.9.10 差出用メッセージ

Gnus が unplugged のとき、ディフォルトではすべての差出用メッセージ(メールとニュースの両方) は下書きグループ“queue”(see Section 6.7 [Drafts], p. 143) に格納されます。投稿した後でも、そこでそのメッセージを見たり編集するのは意のままです。

送出するメールが queue される(順番待ちになる) 状況を制御することは可能です(`gnus-agent-queue-mail`, Section 7.9.11 [Agent Variables], p. 228, 参照)。Gnus が unplugged のとき、外に送り出すニュースは常に queue されるだけです。

下書きグループから、そこで使える特別な命令を使ってメッセージを送信することもできるし、グループバッファー内で*J s* を使って、下書きグループ内のすべての送信可能なメッセージを送信することもできます。ニュースの投稿は Gnus が plugged のときだけできますが、メールはいつでも送信することができます。

Unplugged のときにメールの送信ができなくて、かつ unplugged のときにうっかり*J s* を叩いてしまうことが心配ならば、Gnus にあなたの行動を確認させることができます(`gnus-agent-prompt-send-queue`, Section 7.9.11 [Agent Variables], p. 228, 参照)。

7.9.11 エージェント変数

`gnus-agent`

エージェントが有効になっているかどうか。ディフォルトは`t` です。最初に有効にされると、いくつかのバックエンドを自動的にエージェント化するために、エージェントは`gnus-agent-auto-agentize-methods` を使います。サーバーバッファーでエージェントのコマンドを使うことによって、どのバックエンドをエージェント化するかを変更することができます。

サーバーバッファーに入るには、グループバッファーで^ (`gnus-group-enter-server-mode`) を使ってください。

`gnus-agent-directory`
: Gnus エージェントがファイルを格納する場所です。ディフォルトは`~/News/agent/`です。

`gnus-agent-handle-level`
: この変数の値より高いレベル(see Section 3.6 [Group Levels], p. 19) のグループは、エージェントからは無視されます。ディフォルトは`gnus-level-subscribed`で、これはディフォルトでは、購読しているグループのみがエージェントの処理の対象となるということです。

`gnus-agent-plugged-hook`
: ネットワークに接続されたときに実行されるフックです。

`gnus-agent-unplugged-hook`
: ネットワークから切断されたときに実行されるフックです。

`gnus-agent-fetched-hook`
: 記事を取り込み終わったときに実行されるフックです。

`gnus-agent-cache`
: Plugged のときに、ローカルに格納されているNOV と記事を使うかどうかを制御する変数で、例えばエージェントをキャッシュとして使うには必須です。ディフォルトでは非-`nil` で、エージェントをキャッシュとして使います。

`gnus-agent-eagerly-store-articles`
: もし`nil` でなければ(それがディフォルト)、エージェント化されたグループで読まれたすべての記事をエージェントキャッシュに保存します。

`gnus-agent-go-online`
: `gnus-agent-go-online` が`nil` だったら、エージェントはオフライン状態のサーバーをオンライン状態にしません。`ask` だったら、それがディフォルトですが、エージェントは再接続するときにオフライン状態のサーバーをオンライン状態にしたいかどうかを尋ねます。それ以外の値だったら、オフライン状態のサーバーは自動的にオンライン状態になります。

`gnus-agent-mark-unread-after-downloaded`
: `gnus-agent-mark-unread-after-downloaded` が非-`nil` だったら、ダウンロードした後で記事に未読の印を付けます。これは通常、新しくダウンロードされた記事を明確に未読にするための安全な行為です。ディフォルトは`t` です。

`gnus-agent-synchronize-flags`
: `gnus-agent-synchronize-flags` が`nil` だったら、エージェントは決して自動的にフラグを同期させません。それが`ask` だったら(それがディフォルトです)、エージェントはすべての変更を検査して、再び接続したときにそれらを同期させるかどうかを尋ねます。`nil` でも`ask` でもなかったら、すべてのフラグが自動的に同期させられます。

`gnus-agent-consider-all-articles`
: `gnus-agent-consider-all-articles` が非-`nil` だったら、エージェントはすべての記事について、それらをダウンロードする必要があるかどうかをエージェントの述語に決定させます。`nil` だった場合、それがディフォルトですが、エージェントは未読の記事をダウンロードするかどうかだけを述語に決定させ

ます。これを有効にするのならば、後でエージェントが期限切れ消去する記事を何度も繰り返しダウンロードしないように、エージェントの期限切れ消去の設定(see Section 7.9.2.3 [Category Variables], p. 223) を見直す必要があるでしょう。

`gnus-agent-max-fetch-size`
: エージェントは、取得した記事を個々のファイルに入れるための解析を行なう前に、それらを一時的なバッファーへ取り込みます。最大のバッファーサイズを超過しないようにするために、記事がすべて取得されるまで、エージェントは取得と解析を交互に行ないます。`gnus-agent-max-fetch-size` は、繰り返しがどれくらい頻繁に起きるかを制御するための、サイズの限界を規定します。大きな値は性能を向上させます。小さな値は、万が一取得している間に接続が切れた場合に、遅れ時間を最小にします(グループの状態を更新するために`gnus-agent-regenerate-group` を実行する必要があるかもしれません。でも、接続が切れる前に解析されたすべての記事は、unplugged の期間に利用することができるでしょう。)。繰り返しに遭遇することは珍しいので、ディフォルトは 10M です

`gnus-server-unopen-status`
: エージェント変数ではないかもしれないけれどエージェントに密接に関連するこの変数は、Gnus がサーバーに接続できないときに何をするかを指示します。エージェントが活性化されると、ディフォルトの`nil` では、サーバーとの接続を絶つかエージェントを unplugged にするかを利用者に尋ねます。エージェントが不活性化されると、Gnus はいつも単にサーバーとの接続を絶ちます。この変数の他の選択肢には`denied` と`offline` があり、後者はエージェントを使う場合だけ有効です。

`gnus-auto-goto-ignores`
: おおかたの人は、エージェント変数ではないけれども密接に関連するもう一つの変数をここで探すでしょう。この変数は、ダウンロードされていない(ヘッダーだけがエージェントに格納された)、そして取り込まれていない(記事もヘッダーも格納されていない) 記事の周りでどう移動するかを概略バッファーに伝えます。

  有効な値は`nil` (どの記事にも移動する)、`undownloaded` (unplugged のときは取り込まれていない記事を無視する)、`always-undownloaded` (取り込まれていない記事を常に無視する)、`unfetched` (ヘッダーが取り込まれていない記事を無視する) です。

`gnus-agent-queue-mail`
: `gnus-agent-queue-mail` を`always` にすると、Gnus はメールをいきなり送信してしまうのではなく、常に queue (順番待ち) に入れます。`t` だったら Gnus は unplugged のときだけメールを queue に入れます。`nil` だったら queue に入れません。ディフォルトは`t` です。

`gnus-agent-prompt-send-queue`
: `gnus-agent-prompt-send-queue` が非-`nil` だったら、unplugged であるのにもかかわらず*J s* を叩いた場合に、Gnus は本当にそれを行なっても良いかどうかを確認します。ディフォルトは`nil` です。

`gnus-agent-auto-agentize-methods`
: あなたが以前にエージェントを使ったことが無い(もっと技術的には`~/News/agent/lib/servers` が無い場合)、Gnus はほんの少数のサーバーを自動的にエージェント化します。この変数はどのバックエンドを自動でエージェント化すべきかを制御します。一般に、エージェント化することが有用なのは遠隔バックエンドに対してだけです。自動的にエージェント化することは、サーバーに対して*J a* を実行するのと同じ効果があります(see Section 7.9.3.3 [Server Agent Commands], p. 225)。もしファイルが存在するならば、それらを追加したり削除するためにサーバーを手動で操作しなければなりません。この変数は最初に Gnus を起動したときだけ適用されます。ディフォルトは‘`nil`’ です。

7.9.12 設定例

あなたがこのマニュアルを読みたくなくて、ごく標準的な設定を行なっているのならば、`~/.gnus.el` ファイルとして何か以下のようなものを使って始めても良いでしょう。

```
;; Gnus がどのようにニュースを取得するかを定義します。ここ
;; では ISP のサーバーから NNTP で取ってくることにします。
(setq gnus-select-method '(nntp "news.your-isp.com"))

;; Gnus がどのようにメールを読むかを定義します。
;; ISP の POP サーバーからメールを読むことにします。
(setq mail-sources '((pop :server "pop.your-isp.com")))

;; Gnus がメールをどのように格納するかを指定します。
;; nnml グループを使うことにします。
(setq gnus-secondary-select-methods '((nnml "")))

;; Gnus をオフラインニュースリーダーにします。
;; (gnus-agentize) ; 旧式の設定。
;; (setq gnus-agent t) ; 現在のディフォルト。
```

基本的にはこれだけで良いはずです。これを`~/.gnus.el` ファイルに入れて、必要に応じて編集し、PPP (や何か) を起動して、*M-x gnus* とタイプしてください。

あなたが Gnus を走らせたのが初めてであれば、自動的にわずかなディフォルトのニュースグループが読めるようになります。おそらくもっとたくさんのグループを購読したくなるでしょう。そのためには、*A A* 命令でグループの完全なリストをNNTP サーバーに問い合わせなければなりません。これは普通はとても時間がかかりますが、一度だけしか実行する必要はありません。

読み込みと解析にしばらく時間を費やした後で、グループの一覧が現れます。そうしたら、読みたいグループを*u* 命令で購読できるようにしてください。読みたいグループを全部購読できるようにしたら、*l* で killed (削除された) グループをすべて画面から消去しましょう。(*A k* で killed グループはすべて戻ってきます。)

今やすぐにグループを読むこともできるし、*J s* 命令で記事をダウンロードすることもできます。あとはこのマニュアルの残りを読んで、他の億千万の項目からカスタマイズしたいことを見つけ出してください。

7.9.13 一括エージェント処理

Gnus エージェントに記事を取得させるのは(そしてあなたの書いた何かのメッセージを投稿するのは)、いったんものごとを正しく設定してしまえば非常に簡単です。以下のシェルスクリプトは必要なことをすべてやってくれるでしょう。

以下の呪文をコマンドラインで使うことによって、完全なバッチコマンドを走らせることができます:

```
#!/bin/sh
emacs -batch -l ~/.emacs -l ~/.gnus.el -f gnus-agent-batch >/dev/null 2>&1
```

7.9.14 エージェントの問題点

Gnus エージェントは、よくある他のオフラインニュースリーダーのようには動作しません。これらは架空の人々からの良くある質問です:

Plugged のときに記事を読んだら、それはエージェントに入るのですか?
: はいそうです。`gnus-agent-eagerly-store-articles` のディフォルト値がそうなっていますから。別のやり方は`gnus-select-article-hook` に関数`gnus-agent-fetch-selected-article` を加えることです。

Plugged のときに記事を読んで、エージェントに記事が存在している場合、もう一回ダウンロードされるのですか?
: いいえ、ただし`gnus-agent-cache` が`nil` でなかったら、ですが。

要約すると、Gnus が unplugged のときはローカルに保存された記事を見るだけです。Plugged のときは ISP と話し、かつローカルに持っている記事も使うでしょう。

8 スコア

ほかの人たちは「削除ファイル」(see Section 8.13 [Kill Files], p. 251) を使いますが、ここ Gnus タワーにいる私たちはスコアリングの方が好きなので、彼らとけんかをするよりは切り替えてしまおうとしています。スコアリングとスコアファイルの処理は、削除ファイルの処理よりも強力で高速です。しかもスコアリングは完全に違うことをするので、真っ直ぐに座って注意を払ってください!

すべての記事はディフォルトのスコア(gnus-summary-default-score) の値を持っていて、ディフォルトでは 0 です。このスコアは対話的に、またはスコアファイル(score file) によって、上げられるか下げられるかします。gnus-summary-mark-below よりも低いスコアを持っている記事には既読の印が付きます。

Gnus は概略バッファーを作成する前に、現在のグループに適用されるどんな「スコアファイル」も読み込みます。

現在の記事に基づいてスコアのエントリーを挿入する、複数の概略バッファーの命令があります。例えば、Gnus に特定の表題の記事のスコアを下げたり上げたりするように求めることができます。

二種類のスコア・エントリーがあります: 永続的なものと一時的なものです。一時的なスコア・エントリーは、自分自身で期限切れ消去するエントリーです。例えば一週間以上使われていないエントリーは、スコアファイルの大きさを小さくしておくために静かに削除されます。

8.1 概略スコア命令

スコア・エントリーを変更するスコア命令は、実際に本当のスコアファイルを修正するわけではありません。それはあまりに非効率です。Gnus は以前にロードされたスコアファイルのキャッシュを保持していて、その一つが「現在のスコアファイルの連想リスト」だと見なされます。スコア命令は単にこのリストにエントリーを挿入し、グループから出るときに、このリストは保存されます。

現在のスコアファイルは、実際にそのようなスコアファイルが存在しない場合でも、ディフォルトでグループのローカルスコアファイルになります。スコア命令を何か他のスコアファイル(例えばall.SCORE) に挿入するには、まずこのスコアファイルを現在のものにしなければなりません。

以下はスコアファイルを実際に変更しない、一般的なスコア命令です:

V s 現在の記事のスコアを設定します(gnus-summary-set-score)。

V S 現在の記事のスコアを表示します(gnus-summary-current-score)。

V t 現在の記事に使われているすべてのスコア規則を表示します(gnus-score-find-trace)。*Score Trace* バッファーにおいて、現在の行のスコア規則に対応するスコアファイルを編集するには*e* を、スコアファイルの清書(gnus-score-pretty-print) と編集を行なうためには*f* をタイプしてください。

V w スコアで使われている語のリストを表示します(gnus-score-find-favorite-words)。

V R 現在の概略でスコアの処理を実行します(gnus-summay-rescore)。Gnus には内緒でスコアファイルをいじり回して、その効果を見たいときに役立つでしょう。

`V c` 違うスコアファイルを現在のものにします(gnus-score-change-score-file)。

`V e` 現在のスコアファイルを編集します(gnus-score-edit-current-scores)。gnus-score-mode バッファーが現れるしょう(see Section 8.5 [Score File Editing], p. 244)。

`V f` スコアファイルを編集して、このスコアファイルを現在のものにします(gnus-score-edit-file)。

`V F` キャッシュされているスコアを捨てます(gnus-score-flush-cache)。これはスコアファイルを編集した後で役に立ちます。

`V C` 視覚的に快適な方法でスコアファイルをカスタマイズします(gnus-score-customize)。

以下の命令はローカルスコアファイルを変更します:

`V m` スコアの入力を求めて、それよりも低いスコアのすべての記事に既読の印を付けます(gnus-score-set-mark-below)。

`V x` スコアの入力を求めて、そのスコアより低いすべての記事を削除するためのスコア規則を現在のスコアファイルに付け加えます(gnus-score-set-expunge-below)。

スコア・エントリーを実際に作るためのキー操作は、非常に規則正しい様式にのっとっているので、それらすべての(何百もある) 命令を列挙する必要は無いでしょう。

1. 最初にタイプするキーは、スコアを増やすときは`I` (i の大文字) で、スコアを下げるときは`L` です。
2. 二番目のキーは、どのヘッダーでスコアを付けるかを指定します。以下のキーを使うことができます:

   `a` 著者(author) の名前でスコアを付けます。

   `s` 表題(subject) の行でスコアを付けます。

   `x` Xref 行、すなわちクロスポスト行でスコアを付けます。

   `r` References 行でスコアを付けます。

   `d` 日付(date) でスコアを付けます。

   `l` 行数(number of lines) でスコアを付けます。

   `i` Messsage-ID ヘッダーでスコアを付けます。

   `e` NNTP サーバーが追加のヘッダーのデータを overview で捕捉していれば、その「追加」のヘッダー(すなわち(gnus-extra-headers に設定されているもの) の一つでスコアを付けます。

   `f` フォローアップ(followup) でスコアを付けます---これは著者名と合致するかどうかを調べて、この著者へのフォローアップでスコアを加えます。(このキーを使うことは、ADAPT ファイルの生成をもたらします。)

   `b` 記事の本文でスコアを付けます。

   `h` ヘッダーでスコアを付けます。

   `t` スレッドでスコアを付けます。(このキーを使うことは、ADAPT ファイルの生成をもたらします。)

3. 三番目のキーは合致の型です。どの合致の型が有効なのかは、どのヘッダーでスコアを付けようとしているかに依ります。

   文字列(strings)

   e 正確な(exact) 合致です。

   s 文字列の一部の(substring) 合致です。

   f 大雑把な(fuzzy) 合致です(see Section 10.17 [Fuzzy Matching], p. 291)。

   r 正規表現(regexp) の合致です。

   日付(date)

   r 正規表現(regexp) の合致です。

   b 日付の前(before) です。

   a 日付の後(after) です。

   n その日付です。

   < それ未満の日数です。

   > それを越える日数です。

   数値(number)

   < 数値より小さいものです。

   = 数値と等しいものです。

   > 数値より大きいものです。

   本文の文字列(body-strings)

   これらの合致の型は'head' とbody のヘッダータイプで使うことができます。用できます。

   z 文字列の一部の(substring) 合致です。

   p 正規表現(regexp) の合致です。

4. 普通は最後になる四つ目のキーは、これが一時的な(すなわち期限切れ消去される) スコア・エントリーか、永続的な(すなわち期限切れ消去でない) スコア・エントリーか、またはスコアファイルに追加せずにただちにスコア付けを行なうか、のどれかを指定します。

   t 一時的な(temporary) スコア・エントリーです。

   p 永続的な(permanent) スコア・エントリーです。

   i ただちに(immediate) スコア付けを行ないます。

5. もし'e' (追加の(extra)) ヘッダーでスコア付けを行なっていると、それでスコア付けをしたいヘッダーの名前を尋ねられるでしょう。これはgnus-extra-headers にある名前でなければなりません。'TAB' による補完ができます。

そういうわけで、現在の著者への、正確な合致に基づいて、永続的なスコアを、増やしたい、という場合のキーは*I a e p* です。表題への、文字列の一部合致に基づいて、一時的

なスコア・エントリーを作り、そのスコアを下げたい、という場合のキーは*L s s t* です。ずいぶん簡単ですね。

ものごとを少し複雑にするためにショートカット・キーがあります。二番目か三番目のキーに大文字を使うと、Gnus は残る一つか二つのキーにディフォルト値を使います。ディフォルトは「文字列の一部」と「一時的」です。ですから*I A* は*I a s t* と同じで、*I a R* は*I a r t* と同じです。(これらのショートカットは本文の合致では使えません。)

これらの関数は、数値接頭引数とシンボル接頭引数を受け付けます(see Section 10.3 [Symbolic Prefixes], p. 272)。数値接頭引数はどのくらい記事のスコアを下げる(もしくは上げる) かを指定します。シンボル接頭引数`a` は、現在のスコアファイルの代わりに`all.SCORE` ファイルをその命令のために使うことを指示します。

`gnus-score-mimic-keymap` はこれらの命令がキーマップであるかのように振る舞うかどうかを指定します。

8.2 グループスコア命令

残念ながら、まだたくさんはありません。

W e すべてのグループに適用される all.SCORE ファイルを編集します。`gnus-score-mode` のバッファーが現れるでしょう(see Section 8.5 [Score File Editing], p. 244)。

W f 何度もスコア連想リストを再読み込みすることを避けるために、Gnus はそれらのキャッシュを保持しています。この命令はキャッシュを空っぽにします(`gnus-score-flush-cache`)。

以下のようなやり方で、コマンド行からスコア付けをすることができます:

```
& emacs -batch -l ~/.emacs -l ~/.gnus.el -f gnus-batch-score
```

8.3 スコア変数

`gnus-use-scoring`
`nil` であれば、Gnus はスコアファイルを調べず、一般的にはスコア関連の仕事をまったくしません。これはディフォルトで`t` です。

`gnus-kill-killed`
この変数が`nil` であれば、Gnus はすでに削除(kill) の処理を実行された記事に決してスコアファイルを適用しません。これはたくさんの時間を節約する一方、削除ファイルをグループに適用していて、削除ファイルを変更し、もっと多くの記事を削除するためにそれを再実行しても、それは動作しないということにもなります。それをするためにはこの変数を`t` にしなければなりません。(これはディフォルトで`t` です。)

`gnus-kill-files-directory`
すべての削除(kill) とスコアのファイルはこのディレクトリーに格納されます。その値は、ディフォルトでは環境変数`SAVEDIR` によって初期化されます。ディフォルトは`~/News/` です。

`gnus-score-file-suffix`
スコアファイルの名前を得るためにグループ名に加える接尾語です(ディフォルトは`SCORE` です)。

`gnus-score-uncacheable-files`
: 通常すべてのスコアファイルは、スコアファイルの過剰な再読み込みを避けるためにキャッシュされます。しかし、このために Emacs が大きく肥大化するかもしれません。そこで、再び必要とされそうもないスコアファイルを取り除くためにこの正規表現を使うことができます。`all.SCORE` のキャッシュをやめてしまうのは間違った考えですが、`comp.infosystems.www.authoring.misc.ADAPT` をキャッシュしないのは良い考えかもしれません。実際のところ、この変数のディフォルトは‘`ADAPT$`’ で、適応スコアファイルはキャッシュされません。

`gnus-save-score`
: もし本当に複雑なスコアファイルを持っていて、たくさんの一括(batch) スコア付けを行なうのであれば、この変数を`t` に設定しても良いでしょう。これは Gnus にスコアを`.newsrc.eld` ファイルに保存させます。

  これを`t` に設定しないと、手動で付けたスコア(*V s* (`gnus-summary-set-score`) で設定したようなもの) は訪れたグループ群を横切って保持されることはありません。

`gnus-score-interactive-default-score`
: スコアを上げ/下げするために、すべての対話的スコア上げ/下げ命令によって使われるスコアです。ディフォルトは 1000 で、過剰に思えるかもしれませんが、適応スコア付けをするための十分な余地を確保するためです。手で入力したデータを、適応スコア付けからの小さな変更で上書きされたくはないのです。

`gnus-summary-default-score`
: 記事のスコアのディフォルトで、ディフォルトでは 0 になっています。

`gnus-summary-expunge-below`
: この変数より低いスコアを持つ記事は概略の行に表示されません。ディフォルトは`nil` で、どの記事も隠されないということです。これは各概略バッファーにおけるローカル変数で、`gnus-summary-mode-hook` によって設定されなければなりません。

`gnus-score-over-mark`
: ディフォルトのスコアより大きなスコアを持つ記事に対して(概略行の三桁目で) 使われる印です。ディフォルトは‘`+`’ です。

`gnus-score-below-mark`
: ディフォルトのスコアより小さなスコアを持つ記事に対して(概略行の三桁目で) 使われる印です。ディフォルトは‘`-`’ です。

`gnus-score-find-score-files-function`
: 現在のグループのためのスコアファイルを見つけるために使われる関数です。この関数はグループ名を引数として呼ばれます。

  あらかじめ定義されている使用可能な関数は:

  `gnus-score-find-single`
  : グループ自身のスコアファイルだけを適用します。

  `gnus-score-find-bnews`
  : すべての合致するスコアファイルを bnews 構文を使って適用します。これがディフォルトです。例えば、現在のグルー

プが‘gnu.emacs.gnus’ ならば、gnu.all.SCORE, not.alt.SCOREとgnu.all.SCORE がすべて適用されます。要するに、スコアファイル名の‘all’ が‘.*’ に変換され、それから正規表現の合致がなされます。

これは、すべてのグループに適用したいスコア・エントリーがいくつかある場合は、それらのエントリーをall.SCORE ファイルに入れるということです。

Gnus は、より一般的なスコアファイルを、より特定のスコアファイルより前に適用しようとするものの、スコアファイルはややランダムな順番で適用されます。これはスコアファイル名の要素の数を調べることによって行なわれます—‘all’ 要素を取り除いて。

gnus-score-find-hierarchical
: すべての親グループから、順にすべてのスコアファイルを適用します。これはall.SCORE のようなスコアファイルを持つことはできないけれど、SCORE, comp.SCORE およびcomp.emacs.SCORE を、それぞれのサーバーに対して持つことができるということです。

この変数は関数のリストであることもできます。その場合、これらすべての関数がグループ名を引数として呼ばれ、返されたすべてのスコアファイルのリストが適用されます。これらの関数は、直接スコア連想リストのリストのリストを返すこともできます。その場合、それらのファイルでないスコア連想リストを返す関数は、返される最後のスコアファイルがローカルスコアファイルであることを確実にするために、おそらく「本当の」スコアファイル関数よりも前に置かれるべきでしょう。ふぅ。

例えば、サーバーを特定しない全体スコアファイルを使って、階層的に(親グループから順に) スコア付けを行なうならば、次の値を使えば良いでしょう。

```
(list (lambda (group) (list "all.SCORE"))
      'gnus-score-find-hierarchical)
```

gnus-score-expiry-days
: この変数は、使われていないスコアファイルエントリーが期限切れ消去されるまでに、どのくらいの日数が経過すべきかを指定します。この変数がnil であると、スコアファイルエントリーは削除されません。ディフォルトは 7 です。

gnus-update-score-entry-dates
: この変数がnil でないと、一時的に合致したスコア・エントリーは日付が更新されます。(これは Gnus が期限切れ消去を操作している方法です---すべての合致しないエントリーは古くなりすぎるのに対して、合致するエントリーは新鮮で若いままです。) しかし、この変数をnil に設定すると、合致するエントリーでさえも古くなって、ああ、そう、あの冷酷な死神と直面することになるでしょう。

gnus-score-after-write-file-function
: スコアファイルが書かれた直後に、その名前を引数として呼ばれる関数です。

gnus-score-thread-simplify
: この変数がnil でないと、記事の表題は表題でスコア付けを行なうために、スレッドと同じやり方で(現在のgnus-simplify-subject-functions の値に従

って) 単純化されます。スコア・エントリーが「文字列の一部への合致」か「正確な合致」を使っていると、その合致もこのやり方で単純化されます。

8.4 スコアファイル様式

スコアファイルは普通は単一の様式だけを含むemacs-lisp ファイルです。無頓着な利用者はこれを編集しないでください。すべては概略バッファーから変更することができます。

にもかかわらず、それを自分でいじってみたくなったのなら、例があります:

```
(("from"
  ("Lars Ingebrigtsen" -10000)
  ("Per Abrahamsen")
  ("larsi\\|lmi" -50000 nil R))
 ("subject"
  ("Ding is Badd" nil 728373))
 ("xref"
  ("alt.politics" -1000 728372 s))
 ("lines"
  (2 -100 nil <))
 (mark 0)
 (expunge -1000)
 (mark-and-expunge -10)
 (read-only nil)
 (orphan -10)
 (adapt t)
 (files "/home/larsi/News/gnu.SCORE")
 (exclude-files "all.SCORE")
 (local (gnus-newsgroup-auto-expire t)
        (gnus-summary-make-false-root empty))
 (eval (ding)))
```

この例はたいていのスコアファイルの要素を説明しています。別のやり方については、Section 8.15 [Advanced Scoring], p. 253, を見てください。

これがとても Lisp コードのように見えるとしても、実際はここにあるものは何もeval (評価) されません。しかしこの様式を読み込むために Lisp リーダーが使われるので、意味的に有効でないとしても、文法的には正当なものです。

この連想リストでは六つのキーがサポートされています:

文字列(STRING)
: キーが文字列だったら、それは合致が実行されるヘッダーの名前です。スコア付けはこれら八つのヘッダーだけで行なうことができます: From, Subject, References, Message-ID, Xref, Lines, Chars およびDate です。これらのヘッダーに加えて、Gnus に、記事全体を取得して記事のより大きな部分で合致を行なわせる三つの文字列があります: Body は記事の本文で合致を行ない、Head は記事のヘッダーで合致を行ない、All は記事全体で合致を行ないます。これら最後の三つのキーを使うと、グループに入る速度をかなり 遅くしてしまうことに気を付けてください。スコアを付けることができる最後の「ヘッダー」はFollowup です。これらのスコア・エントリーは、これらのスコア・エント

リーに合致する記事へのすべてのフォローアップのための追加が行なわれている、新しいスコア・エントリーに帰着するでしょう。

このキーに続くのは任意の数のスコア・エントリーで、それぞれのスコア・エントリーは一つから四つまでの要素を持ちます。

1. 最初の要素は「合致要素」です。これはたいていのヘッダーでは文字列ですが、Lines と Chars ヘッダーでは整数でなければなりません。
2. もし二番目の要素があるなら、それは数値の「スコア要素」でなければなりません。この数値は負の無限大から正の無限大までの間の整数でなければなりません。合致が成功すると、この数値が記事のスコアに加えられます。この要素が存在していない場合は、代わりに`gnus-score-interactive-default-score` の数値が使われます。ディフォルトは 1000 です。
3. もし三番目の要素があるなら、それは数値の「日付要素」でなければなりません。この日付は最後にこのスコア・エントリーが合致した時刻を示し、それはスコア・エントリーを期限切れ消去するための機構を提供します。この要素が存在していないと、スコア・エントリーは永続になります。日付は紀元前 1 年 12 月 31 日からの経過した日数で表されます。
4. もし四番目の要素があるなら、それはシンボルの「型要素」でなければなりません。この要素は、このスコア・エントリーが記事に合致するかどうかを調べるために、どの関数が使われるべきかを指定します。

   From, Subject, References, Xref, Message-ID
   : たいていのヘッダー型のために、`r` と`R` (正規表現(regexp))、`s` と`S` (文字列の一部(substring)) 型、`e` と`E` (正確な合致(exact match))、および`w` (語の合致((word match)) 型があります。もしこの要素が無いと、Gnus は文字列の一部の合致が用いられるべきであると仮定します。`R`, `S`, `E` は、合致が大文字と小文字を区別する方法で行なわれる点で他のものと異なります。これらすべての一文字型は、本当は`regexp`, `exact`, `word` 型の短縮形で、この方が好みならば代わりに使うことができます。

   Extra
   : overview ヘッダーの標準の文字列に的を絞って`gnus-extra-headers` を使っていれば、それらのヘッダーの値でスコアを付けることができます。この場合スコア・エントリーの五番目の要素が、スコアを付けるヘッダーの名前になります。NNTP サーバーが overview で‘`NNTP-Posting-Host`’ を捕捉しているならば、`all.SCORE` ファイルの以下のエントリーは、単一のホストを起源とする spam の攻撃に対して有効です:

   ```
   ("111.222.333.444" -1000 nil s
    "NNTP-Posting-Host")
   ```

   Lines, Chars
   : これらの二つのヘッダーは別の合致の型を使います: `<`, `>`, `=`, `>=`, `<=` です。

   これらの述語は

   ```
   (PREDICATE HEADER MATCH)
   ```

の評価が`nil` ではない場合に真です。例えば、上級合致(`"lines" 4 <`) (see Section 8.15 [Advanced Scoring], p. 253) は結果として以下の式になります:

```
(< header-value 4)
```

言い換えると、4 を合致として`<` を`Lines` で使っているときは、記事が 4 行よりも少ないときにスコアが加算されるということです。(混乱して、それが反対ではないかと考えがちです。でも、そうではないのです。私が思うに。)

合致を`Lines` で行なっていると、いくつかのバックエンド(`nndir` のようなもの) は`Lines` ヘッダーを作成しないので、すべての記事が 0 行であるとして扱われてしまうことに気を付けてください。これは、少しの行しかない記事のスコアを下げている場合に、変な結果を導くことがあり得ます。

Date Date (日付) ヘッダーには三つのなんとなくばかげている合致の型があります: `before`, `at`, `after` です。私は本当にこれが役立つような機会を想像できないのですが、この関数を提供しないのもなんとなくばかげています。そうした場合のためにあるのです。いつ必要になるかは誰にもわかりません。転ばぬ先の杖。羹(あつもの)に懲りて膾(なます)を吹く。本をカバーで判断してはいけません。初めてのデートでエッチしてはいけません。(しかし、私は少なくとも一人、引用しますが、「この関数は欠かせないものであることがわかった」と言った人がいると聞いています。)

もっと役に立つ合致の型は「正規表現」です。それによって、日付の文字列に正規表現を使って合致させることができます。日付はまず ISO8601 の短縮様式(compact format) に標準化されます—*YYYYMMDD*`T`*HHMMSS* です。例えば、すべての年の 4 月 1 日に投稿されたすべての記事に合致させたいのであれば、合致文字列として‘`....0401.........`’ を使うことができます。(日付は元々の標準時で保存されているので、その記事が投稿された場所での 4 月 1 日に投稿された記事に合致することに注意してください。“Time zones”は家族全員の健全な楽しみですね? (訳注: いくつかある“Time zones”というタイトルの曲のことを言っているのかもしれません。))

最後は、実際に日付に役立つ 2 つの合致形式`<` と`>` です。これらは記事の相対的な期間(日数) のスコア付けを可能にします。そのメソッドを使ったスコアファイルの例です:

```
(("date"
  (7 10 nil <)
  (7 -10 nil >)
  (14 -10 nil >)))
```

これによって 1 週間未満の記事は 10 ポイント増加、1 週間以上前の記事は 10 ポイント減少、2 週間以上前の記事は累積 20 ポイント減少します。

日数は浮動小数点数にすることもできます。1 時間以内の記事をスコア付けするには‘(0.04 10 nil <)’ としてください。

Head, Body, All
これらの三つの合致のキーはFrom ヘッダー(など) と同じ合致の型を使います。

Followup この合致のキーはやや特別で、それはFrom ヘッダーに合致し、合致した記事だけでなくその記事へのすべてのフォローアップのスコアにも影響します。これは、あなた自身の記事へのフォローアップのスコアを増やしたり、良く知られた問題児へのフォローアップ記事のスコアを下げたりするのに使われます。From ヘッダーが使うのと同じ型の合致を使います。(この合致キーを使うと、ADAPT ファイルを作ることになります。)

Thread この合致キーはFollowup 合致キーと同じ方針に沿って動作します。Message-ID x で始まっているスレッド(または副スレッド) にスコアを付けたいのであれば、‘thread’ 合致を付け加えてください。これはReference ヘッダーにx を持つそれぞれの記事に、新しい‘thread’ 合致を追加します。(これらの新しい‘thread’ 合致はこれらの合致する記事のMessage-ID を使います。) これはスレッドのいくつかの記事が完全なReferences ヘッダーを持っていなかったとしても、スレッド全体のスコアを上げ/下げできることを保証します。これを使うと、スレッドの記事に決定的でないスコアが付くかもしれないということに注意してください。(この合致キーを使うと、ADAPT ファイルを作ることになります。)

score-fn このエントリーの値は、括弧で囲まれた 1 つまたはそれ以上のユーザーが定義した関数名です。それぞれの関数は順番に呼び出されます。戻り値は整数でなければなりません。

```
(score-fn (custom-scoring))
```

ユーザー定義関数は、キーnumber subject from date id refs chars lines xref extra に、その関数を実行する前の記事のスコアが続く連想リストを引数として呼び出されます。

次の(多少工夫された) 例は、記事の日付の年も件名で言及されている場合に、記事のスコアを 10 増やすユーザー定義関数の使用方法を示しています。

```
(defun custom-scoring (article-alist score)
  (let ((subject (cdr (assoc 'subject article-alist)))
        (date (cdr (assoc 'date article-alist))))
    (if (string-match (number-to-string
                       (nth 5 (parse-time-string date)))
                      subject)
        10)))
```

score-fn エントリーは永続的なので、直接SCORE ファイルで追加または変更することしかできません。

mark このエントリーの値は数値でなければなりません。この数値より低いスコアのどんな記事にも既読の印が付きます。

`expunge` このエントリーの値は数値でなければなりません。この数値より低いスコアのどんな記事も概略バッファーから削除されます。

`mark-and-expunge`
このエントリーの値は数値でなければなりません。この数値より低いスコアのどんな記事にも既読の印が付き、概略バッファーから削除されます。

`thread-mark-and-expunge`
このエントリーの値は数値でなければなりません。スコアの総計がこの数値より低いスレッドのすべての記事には既読の印が付き、概略バッファーから削除されます。`gnus-thread-score-function` はスレッドのスコアの総計をどのように計算するかを指定します。

`files` このエントリーの値は任意の数のファイル名でなければなりません。それらのファイルもスコアファイルであるとみなされ、これがされたのと同じ方法で読み込まれます。

`exclude-files`
このエントリーの手がかりは任意の数のファイル名でなければなりません。これらのファイルが何らかの理由で普通は読み込まれるようになっていたとしても、読み込まれません。

`eval` このエントリーの値は`eval` (評価) されます。この要素はグローバルスコアファイルを扱っているときは無視されます。

`read-only`
読み込み専用スコアファイルは更新されたり保存されたりしません。グローバルスコアファイルはこのアトムを使用するべきです(see Section 8.12 [Global Score Files], p. 250)。(注意:「グローバル」はここでは本当に「全体的」という意味です。個人的なすべてのグループに適用するスコアファイルのことではありません。)

`orphan` このエントリーの値は数値でなければなりません。親記事を持たない記事のスコアにこの数値が加えられます。‘`comp.lang.c`’ のような流通量の多いニュースグループを追いかけていると想像してください。おそらくほんの少しのスレッドだけを追いたいでしょう。さらに新しいスレッドは見たいでしょう。

以下の二つのスコアファイルエントリーによって、それをすることができます:

```
(orphan -500)
(mark-and-expunge -100)
```

最初にこのグループに入ったときは、新しいスレッドだけを見るでしょう。そうしたら、興味を持ったスレッドのスコアを上げ(*I T* または*I S* で)、残りを無視(*c y*) してください。次にグループに入ったときは、興味を持ったスレッドの新しい記事と、まったく新しいスレッドを見ることになります。

すなわち orphan (孤児) スコアアトムは、普通のスコア規則では自動的に発見できない、興味深いスレッドが少し存在しする、流通量が多いグループのためにあります。

`adapt` このエントリーは適応スコア付けを制御します。これが`t` だったら、ディフォルトの適応スコア規則が使われます。`ignore` だったら、このグループでは適応スコア付けは行なわれません。もしリストだったら、そのリストが適応ス

コア規則として用いられます。もしそれが存在しないかt やignore 以外の何かだったら、ディフォルトの適応スコア規則が使われます。たいていのグループで適応スコア付けを使いたいのであれば、gnus-use-adaptive-scorint をt に設定し、適応スコア付けをしたくないグループに(adapt ignore) を挿入すればよいでしょう。少しのグループでだけ適応スコアを行ないたいのであれば、gnus-use-adaptive-scoring をnil に設定し、それを行ないたいグループのスコアファイルに(adapt t) を挿入しましょう。

adapt-file
すべての適応スコア・エントリーは、このエントリーによって名づけられたファイルに入ります。さらにそれはグループに入るときにも適用されます。このアトムは、多くのグループで同じ適応スコアファイルを用いることによって、複数のグループに一度に適応スコアを付けたいときに便利でしょう。

local このエントリーの値は(*var value*) の形式の対のリストでなければなりません。それぞれの*var* は現在の概略バッファーでバッファーローカルになり、指定された値(*value*) に設定されます。これは少し風変わりですが、フックがあまり好きでないならば、いくつかのグループで変数を設定するのに便利な方法です。*value* は評価されないことに注意してください。

8.5 スコアファイルの編集

普通はすべてのスコア命令を概略バッファーから発行しますが、手でそれらを編集したくなることもあるかもしれないので、それのためのモードを用意しています。

それは以下に列挙する命令を使えるように、少しカスタマイズしただけのemacs-lisp モードです:

C-c C-c あなたが行なった変更を保存して概略バッファーに戻ります(gnus-score-edit-exit)。

C-c C-d 現在の日付を数値の様式で挿入します(gnus-score-edit-insert-date)。これはどのようなものだろうと考えているのなら、これは本当に日の数値です。

C-c C-p 適応スコアファイルは整形されずに保存されます。もしこれらのファイルの一つを読むつもりなら、まず*pretty print* (整形して印字) したいでしょう。この命令(gnus-score-pretty-print) がそれを行ないます。

このモードを使うには*M-x gnus-score-mode* とタイプしてください。

gnus-score-menu-hook がスコアモードのバッファーで実行されます。

概略バッファーでは、*V f*、*V e* および*V t* のような命令でスコアファイルの編集を始めることができます。

8.6 適応スコア付け

これらのスコア付けはあなたを憂鬱にさせてしまうかもしれないので、Gnus にはこれらをすべて自動的に---まるで魔法でも使ったように作成する方法があります。いやむしろ、人工無能によって、という方が正確かな。

記事を読んだとき、記事に既読の印を付けたとき、あるいは記事を削除したときに、その印を残しておいてください。グループから出るときに、Gnus はそれらの印の辺りを嗅ぎ回り、何の印を見つけたかに応じてスコア要素を追加します。この機能

はgnus-use-adaptive-scoring をt か(line) に設定すると有効になります。もしスコアを、表題に現れる個別の単語をもとに適応させたければ、この変数を(word) に設定してください。両方の適応方法を使いたければ、この変数を(word line) に設定してください。

スコア付けの処理を完全に制御するためにgnus-default-adaptive-score-alist 変数をカスタマイズしてください。例えば、こんな感じになるでしょう:

```
(setq gnus-default-adaptive-score-alist
      '((gnus-unread-mark)
        (gnus-ticked-mark (from 4))
        (gnus-dormant-mark (from 5))
        (gnus-del-mark (from -4) (subject -1))
        (gnus-read-mark (from 4) (subject 2))
        (gnus-expirable-mark (from -1) (subject -1))
        (gnus-killed-mark (from -1) (subject -3))
        (gnus-kill-file-mark)
        (gnus-ancient-mark)
        (gnus-low-score-mark)
        (gnus-catchup-mark (from -1) (subject -1))))
```

ご覧のように、この連想リストの各要素は、キーとして印(変数名か「実際の」印すなわち文字のいずれか) を持ちます。このキーの後には任意の数のヘッダー/スコアの組が続きます。もしそのキーの後にヘッダー/スコアの組が一つもなければ、そのキーが記事の印として付いている記事に対しては適応型スコア付けは行なわれません。例えば上記の例では、gnus-unread-mark が付いている記事は、適応型スコア付けのエントリーを持ちません。

各記事は一つしか印を持ち得ないので、それぞれの記事にはこれらの規則のうちの一つだけが適用されます。

gnus-del-mark を例に取りましょう---この連想リストでの意味は、この印(すなわち‘e’ の印) が付いている記事はすべて、From ヘッダーをもとに−4 下げられSubject で−1 下げられるスコア・エントリーが追加されます。これをあなたの偏見に合わせて変更してください。

もし 10 個の記事に同じ subject でgnus-del-mark の印が付いていたとすると、この印に対する規則は十回適用されます。それはつまり、その subject は−1 の十倍のスコアを得ます。その値は、私が大きく誤解していないかぎり−10 のはずです。

もし自動期限切れ消去(メール) グループ(see Section 7.4.9 [Expiring Mail], p. 181) があれば、既読記事にはすべて‘E’ 印が付けられます。これはおそらく適応型スコア付けをちょっとばかりやりにくくするので、自動期限切れ消去と適応型スコア付けは、一緒には現実にあまりうまくやっていけません。

スコアを付けられるヘッダーにはfrom, subject, message-id, references, xref, lines, chars およびdate があります。さらにfollowup にもスコア付けができて、これは現在の記事のMessage-ID を使ってReferences ヘッダーに合致、すなわちこれに続くスレッドに合致する適応型スコア・エントリーを作成します。

この機構を使うならば、ときどき記事を既読にしてしまう小さな変更を避けるために、スコアファイルのmark アトムを何か小さい値---ひょっとすると−300 くらいに設定しておくべきです。

適応型スコア付けを一週間かそこら使ってくると、Gnus はそれ相応に調教され、あなたが何も言わなくても、あなたの好きな投稿者を強調し、あまり好きではない投稿者を消去するようになるはずです。

どのグループにおいて適応型スコア付けを作動させるかは、スコアファイル(see Section 8.4 [Score File Format], p. 239) を使うことによって制御できます。またこれを使って、違ったグループに対して違った規則を使うようにもできます。

適応型スコア・エントリーは、グループ名にgnus-adaptive-file-suffix を付加した名前のファイルに入れられます。ディフォルトはADAPT です。

適応型スコアファイルは巨大になり得るので、人の手で編集されることは想定されていません。gnus-adaptive-pretty-print がnil (ディフォルト) であると、それらのファイルは人に読めるような形式では書かれません。

適応型スコア付けを行なうときは、部分文字列一致やファジーな一致を行なった方が、おそらくほとんどの場合において良い結果が得られるでしょう。しかし、ヘッダーの一致する部分が短い場合、意図に反する動作をする可能性が大きいので、gnus-score-exact-adapt-limit より短い長さしか一致しない場合は完全一致が行なわれます。この変数がnil であれば、この問題が起こらないように常に完全一致が行なわれます。

上で述べたように、個別の単語ででもヘッダー全体ででも適応を行なうことができます。単語で適応を行なう場合には、それぞれの単語の事例が、ある印にどんなスコアを加えるかを、gnus-default-adaptive-word-score-alist 変数によって指定します。

```
(setq gnus-default-adaptive-word-score-alist
      `((,gnus-read-mark . 30)
        (,gnus-catchup-mark . -10)
        (,gnus-killed-mark . -20)
        (,gnus-del-mark . -15)))
```

これがディフォルト値です。単語での適応を有効にすると、gnus-read-mark の印が付いている記事の表題に現れるすべての単語が、スコアに 30 点追加するというスコア規則を生み出します。

gnus-default-ignored-adaptive-words のリストに現れる単語は無視されます。無視したい単語を追加したいときは、この変数ではなくgnus-ignored-adaptive-words リストの方を使ってください。

短い単語では適応型スコア付けを作動させるべきではないと思う人もいるでしょう。もしそうならgnus-adaptive-word-length-limit に整数を設定することができ、この数値より短い単語は無視されます。この変数のディフォルトはnil です。

スコア付けが行なわれるとき、gnus-adaptive-word-syntax-table が実際に使われるシンタックステーブルです。これは標準シンタックステーブルと似ていますが、数字を単語の構成要素ではない文字だと認識します。

もしgnus-adaptive-word-minimum に数値が設定されていると、単語適応型スコア付け処理において、記事のスコアがこの数値よりも小さくなることはありません。ディフォルトはnil です。

gnus-acaptive-word-no-group-words がt に設定されていると、Gnus はグループ名のすべての語について、単語適応型スコア付けをしません。ほとんどの表題が‘emacs’ という語を含んでいる‘comp.editors.emacs’ のようなグループで便利です。

この機構をしばらく使ってみた後で、規則を解析することによってあなたがどんな単語が好きでどんな単語が嫌いかを診断するgnus-psychoanalyze-user (利用者精神分析命令) を書いてみると良いかもしれません。いや、良くないかな。

単語適応型スコア付けは高度に実験的なものなので、将来変更されるであろうことは心に留めておいてください。第一印象では、これは現状ではまったく使い物にならないように思えます。これをもっと使えるようにするためには、(より厳密な統計的手法を巻き込んで)さらなる作業が行なわれる必要があるでしょう。

8.7 ホームスコアファイル

新しいスコアファイルエントリーが入れられるスコアファイルは、ホームスコアファイル*home score file* と呼ばれます。これは通常(ディフォルトで) そのグループ自身のためのスコアファイルになります。例えば‘gnu.emacs.gnus’ 用のホームスコアファイルはgnu.emacs.gnus.SCORE です。

しかしながら、これはあなたのお望みではないかもしれません。たくさんのグループの間で共通のホームスコアファイルを共有することはしばしば便利です---例えばすべての‘emacs’ グループが、ことによると同じホームスコアファイルを使うことができます。

これを制御する変数がgnus-home-score-file です。これは以下の値を取り得ます:

1. 文字列。この場合、このファイルがすべてのグループでホームスコアファイルとして使用されます。
2. 関数。この関数の結果がホームスコアファイルとして使用されます。この関数はグループの名前を引数として呼び出されます。
3. リスト。このリストの要素は以下の値を取り得ます:
    1. (*regexp file-name*)。*regexp* がグループ名に合致すると、*file-name* がホームスコアファイルとして使用されます。
    2. 関数。この関数がnil 以外を返せば、その戻り値がホームスコアファイルとして使用されます。グループ名が引数として関数に渡されます。
    3. 文字列。この文字列をホームスコアファイルとして使用します。

    このリストは、合致するものを探すために先頭から終りに向かってなぞられます。

というわけで、単一のスコアファイルだけを使いたい場合は、以下のようにすれば良いでしょう:

```
(setq gnus-home-score-file
      "my-total-score-file.SCORE")
```

すべての‘gnu’ グループに対してgnu.SCORE を、すべての‘rec’ グループに対してrec.SCORE (等々) を使いたい場合は、このように設定することができます:

```
(setq gnus-home-score-file
      'gnus-hierarchial-home-score-file)
```

これは利用者の便宜のために、あらかじめ提供されている関数です。他に以下の関数があります:

gnus-current-home-score-file
: 「現在の」標準スコアファイルを返します。これはスコア命令群に「最深」の合致するスコアファイルにエントリーを加えさせます。

‘emacs’ グループ用に一つのスコアファイルを、それとは別のものを‘comp’ グループ用に用意する一方、他のすべてのグループではそれぞれ独自のスコアファイルを使うようにしたいなら、こんな設定で良いでしょう:

```
(setq gnus-home-score-file
```

```
;; 正規表現 "\\.emacs" に合致するすべてのグループ
'(("\\.emacs" "emacs.SCORE")
  ;; すべての comp グループを単一のスコアファイルで
  ("^comp" "comp.SCORE")))
```

gnus-home-adapt-file はgnus-home-score-file とまったく同じやり方で動作しますが、代わりにこれで、何をホーム適応スコアファイルにするかを指定します。すべての新しい適応ファイルエントリーは、この変数で指定されるファイルに入れられ、同じ文法を使うことができます。

gnus-home-score-file とgnus-home-adapt-file を使うことに加えて、グループパラメーター(see Section 3.10 [Group Parameters], p. 23) とトピックパラメーター(see Section 3.16.5 [Topic Parameters], p. 40) を使っても、ほぼ同様のことを成し遂げることができます。グループ、トピックパラメーターはこの変数よりも優先されます。

8.8 自分自身へのフォローアップ

Gnus は現在のバッファーからMessage-ID ヘッダーを見つけ出すための二つの命令を提供します。そして Gnus は、他の記事のReferences ヘッダーにあるこのMessage-ID を使ってスコアを付けるためのスコア規則を追加します。これは事実上、現在のバッファーにある記事に返答したすべての記事のスコアを増加させます。これは、あなたが言ったことに人々が答えたら、それに容易に気付かせてもらいたいときに、とても便利です。

gnus-score-followup-article
: これは、あなた自身の記事に直接フォローアップした記事にスコアを加えます。

gnus-score-followup-thread
: これは、あなたの記事より「下」のスレッドに現れるすべての記事にスコアを加えます。

これら二つの関数は、本来どちらもmessage-sent-hook のようなフックの中で、例えばこのように使うためのものです:

```
(add-hook 'message-sent-hook 'gnus-score-followup-thread)
```

自分のMessage-ID をじっくりと眺めてみると、はじめの二〜三文字は常に同じであることに気が付くでしょう。以下の二つは私のものです:

```
<x6u3u47icf.fsf@eyesore.no>
<x6sp9o7ibw.fsf@eyesore.no>
```

したがって、このマシンでは‘x6’ で「私」かどうかを見分けることができます。これは使えます---以下の規則は、私自身へのすべてのフォローアップのスコアを上げるでしょう:

```
("references"
 ("<x6[0-9a-z]+\\.fsf\\(_-_\\)?@.*eyesore\\.no>"
  1000 nil r))
```

「あなたの」が最初の二文字になるか最初の三文字になるかは、システムに依存します。

8.9 他のヘッダーにスコアを付ける

Gnus が「伝統的」なヘッダー—‘From’, ‘Subject’ など---にスコアを付けるのはとても速いです。ですが、他のヘッダーにスコアを付けるにはhead のスコアのための規則を書く必要があり、それは合致を探すために Gnus が毎回バックエンドから単独の記事を取り寄せなければならないことを意味します。これは大きなグループでは長い時間がかかります。

ヘッダーまたは本文の遅いスコア付けは、変数gnus-inhibit-slow-scoring を設定することによって禁止することができます。もしgnus-inhibit-slow-scoring が正規表現だったら、グループがその正規表現に合致する場合に遅いスコア付けが禁止されます。それがt だったら、すべてのグループで遅いスコア付けが禁止されます。

さて、これに関してニュースグループでの遅さのためにできることは多くはありませんが、メールグループのためにはより優れた手段があります。Section 4.1.2 [To From Newsgroups], p. 50, の章でこの機構がどう働くかが詳しく説明されていますが、ここではどうしたらnnml で‘To’ と‘Cc’ ヘッダーにスコアを付けることができるかの調理の例を挙げましょう。

以下を~/.gnus.el ファイルに置いてください。

```
(setq gnus-extra-headers '(To Cc Newsgroups Keywords)
      nnmail-extra-headers gnus-extra-headers)
```

Gnus を再起動して、*M-x nnml-generate-nov-databases* コマンドでnnml の overview ファイルを作り直してください。たくさんのメールを持っていると、これには長い時間がかかるでしょう。

そして*I e s p To* RET <*your name*> RET のようにすると、‘To’ と‘Cc’ ヘッダーに“extra headers”としてスコアを付けることができます。

わかったかな? 簡単だよね。

8.10 スコア付けの奥義

クロスポスト

クロスポストのスコアを低くしたければ、合致させるべき行はXref ヘッダーです。

```
("xref" (" talk.politics.misc:" -1000))
```

複数のクロスポスト

ある数、例えば三つ以上のグループにクロスポストされている記事のスコアを低くしたければ、

```
("xref"
  ("[^:\n]+:[0-9]+ +[^:\n]+:[0-9]+ +[^:\n]+:[0-9]+"
   -1000 nil r))
```

本文への合致

これは一般的にはあまり良い考えではありません---とても長い時間がかかるからです。実際 Gnus は、それぞれの記事を個別にサーバーから取得してこなければならないのです。でも、とにかくあなたはやりたいのでしょうね。合致させるキーは三つ(Head, Body, All) あるのですが、それぞれのスコアファイルで一つを選んで、それに固執すべきです。もし二つを使うと、それぞれの記事は二回 取得されてしまいます。もしHead でちょっとだけ、Body でもちょっとだけ合致させたい、というのであれば、素直にAll を使って全部合致させてください。

既読として印を付ける

ある一定の値より低いスコアを持つ記事には、おそらく既読の印を付けてしまいたくなるでしょう。これはall.SCORE ファイルに以下のものを入れておくことによって、最も簡単に実現できます。

```
((mark -100))
```

同様のことをexpunge で行なうことを考えても良いでしょう。

否定文字クラス
: もし[^abcd]* みたいなものを指定すると、期待外れの結果で終わるかもしれません。これは改行文字にも合致してしまうので、えーと、未知のものまで引きずり出してしまうもしれません。代わりに[^abcd\n]* を使いましょう。

他のヘッダーでスコア付けするときの継続行
Continuation lines when scoring on other headers
: head またはAll 合致キーと正規表現合致を使って他のヘッダーでスコア付けする場合、正規表現はヘッダーの継続行を考慮する必要があります。例えばTo フィールドの特定のアドレスを含むメッセージに合致させるこの素朴な試みは:

```
("head" "^To: .*\\bspwhitton@spwhitton\\.name\\b" r)
```

以下のようなTo ヘッダーを持つメッセージに合致させることはできません:

```
To: A long description of the Emacs devel list <emacs-devel@gnu.org>,
 spwhitton@spwhitton.name, 12345@debbugs.gnu.org
```

この問題は、この形式の正規表現で取り扱うことができます:

```
("head" "^To: .*\\(?:\n[\s\t].*\\)*\\bspwhitton@spwhitton\\.name\\b" r)
```

8.11 逆スコア

もし、表題ヘッダーに‘Sex with Emacs’ という文字がある記事だけを残して、その他の記事すべてを消去してしまいたければ、スコアファイルに以下のようなものを入れることができます:

```
(("subject"
  ("Sex with Emacs" 2))
 (mark 1)
 (expunge 1))
```

これで‘Sex with Emacs’ に合致するすべての記事のスコアが上がって、残りの記事には既読の印が付き、おまけにそれらは消去されるでしょう。

8.12 グローバルスコアファイル

間違いなく、他のニュースリーダーは「グローバル削除ファイル(global kill file)」を持っています。それらは普通、すべてのグループに適用される、利用者のホームディレクトリーに格納されている一つの削除ファイル以上の何物でもありません。ふふん! つまらない、低能なニュースリーダーだね。

私がここで話しているのはグローバルスコアファイルです。全世界の、あらゆる地域の利用者のスコアファイル。それはすべての国家を巨大な一つの幸せなスコアファイル同盟に団結させる! スコア天使! 新しい、でもテストされていない!

他人のスコアファイルを使うためにしなければならないのは、gnus-global-score-files 変数を設定することがすべてです。それぞれのスコアファイルにつき一つ、またはそれぞれのスコアファイルディレクトリーにつき一つのエントリーになります。Gnus はどのスコアファイルをどのグループに使うのが適切であるかを自分で決定します。

例えばスコアファイル/ftp@ftp.gnus.org:/pub/larsi/ding/score/soc.motss.SCORE および/ftp@ftp.some-where:/pub/score ディレクトリーにあるすべてのスコアファイルを使いたければ、このように設定してください:

```
(setq gnus-global-score-files
      '("/ftp@ftp.gnus.org:/pub/larsi/ding/score/soc.motss.SCORE"
        "/ftp@ftp.some-where:/pub/score/"))
```

単純でしょう? ディレクトリー名は‘/’ で終わらなくてはなりません。これらのディレクトリーは、一般に Gnus を使う一回の期間中に一回だけしか読み込まれません。もし遠隔ディレクトリーを手動で再読み込みすることが必要だと思ったら、`gnus-score-search-global-directories` 命令を使ってください。

ただし、現時点ではこのオプションを使うと、グループに入るのがいくらか遅くなります。(つまり---かなり、ですけど。)

他の人たちに使ってもらうためにスコアファイルを維持管理したくなったら、単にあなたのスコアファイルを匿名 ftp に置いて、世界に公表してください。逆行司会者になりましょう! その後に続いて間違いなく起こる逆行司会者戦争、すなわち人々の共感を勝ち取るための戦いに参加して、彼らのスコアファイルに偽りの前提を使わせるように誘導するのです! やったね! これでネットは救われる!

> ‘`retro-`’ を「逆行」と訳しました。日本では「レトロ」を「古き善き時代の」のような肯定的な意味で使うことが少なくないのですが、ここでは本来の「時代に逆行した」「使えねー」のような意味で使っています。

以下に、逆行司会者なりたがりのための秘技をいくつか、即席で述べます:

- 非常に多くの場所にクロスポストされている記事は間違いなく屑である。
- 一個の不適切な記事を減点するには、`Message-ID` で減点する。
- 特に素晴らしい投稿者たちは永続的な基準で加算して良い。
- そのグループの憲章を無視した投稿を頻繁に繰り返す投稿者は、絶滅させてしまって差し支えない。
- `mark` と`expunge` アトムを設定し、汚らわしい記事を完全に葬り去る。
- 期限切れ消去のスコア・エントリーを使って、ファイルの大きさを小さく抑える。もっとも、サイトによって古い記事を長期間保存するように、おそらく長い期限切れ消去の期間を取るでしょうけれども。

. . . 果たして他のニュースリーダーは、将来グローバルスコアファイルをサポートするでしょうか? うふふ。そう、どう考えてみたって、Blue Wave や xrn や 1stReader とかいったニュースリーダーは、スコアをサポートするべきですね。今は固唾を飲んで見守ることにしましょうか?

8.13 消去ファイル

Gnus はまだ、あのうざったい古い消去ファイルをサポートしています。実際消去ファイルの項目はもう消してもよいのですが、それは Daniel Quinlan がスコアファイルを考え出す前に私が書いたものなので、そのコードはまだ残してあるのです。

要するに、消去処理はスコア処理よりもかなり(私に言わせればものすごく) 遅いので、あなたの消去ファイルはスコアファイルに書き換えた方が良いかもしれません。

いずれにせよ、消去ファイルは普通の`emacs-lisp` ファイルです。このファイルの中にはどんな形式でも入れることができます。つまり消去ファイルをグループに入ったときに実行する一種の原始的なフック関数のように使うことができます。まあそれがあまりいい方法ではないとしてもね。

通常の消去ファイルは以下のようになります:

```
(gnus-kill "From" "Lars Ingebrigtsen")
(gnus-kill "Subject" "ding")
(gnus-expunge "X")
```

これは私が書いたすべての記事に既読の印を付け、概略バッファーから印の付いた記事を削除します。とっても便利です。あなたもそう思うでしょ。

他のプログラムではまったく違う消去ファイルの構文を使っています。Gnus はrn の消去ファイルらしきものに出会うと、何とかそれを解釈しようとします。

GNUS 消去ファイルを編集するための二つの概略バッファー関数があります:

M-k このグループの消去ファイルを編集します(gnus-summary-edit-local-kill)。

M-K 一般消去ファイルを編集します(gnus-summary-edit-global-kill)。

消去ファイルを編集する二つのグループモード関数があります:

M-k このグループの消去ファイルを編集します(gnus-group-edit-local-kill)。

M-K 一般消去ファイルを編集します(gnus-group-edit-global-kill)。

消去ファイル変数:

gnus-kill-file-name
: 'soc.motss' グループ用の消去ファイルは通常soc.motss.KILL という名前です。このファイル名を得るためにグループ名に付加される接尾語は、gnus-kill-file-name 変数で与えられます。「グローバル」消去ファイルは(スコアファイルの意味での「グローバル」じゃないよ、もちろん) 単にKILL という名前です。

gnus-kill-save-kill-file
: この変数がnil 以外であれば、Gnus は処理の後に消去ファイルを保存します。これは期限切れ消去を行なう消去を使っているときに必要です。

gnus-apply-kill-hook
: グループに消去ファイルを適用するために呼び出されるフック。これはディフォルトでは(gnus-apply-kill-file) です。同じグループのためのスコアファイルがある場合に消去ファイルを無視したければ、このフックを(gnus-apply-kill-file-unless-scored) に設定してください。消去ファイルを処理させたくなければ、この変数をnil に設定してください。

gnus-kill-file-mode-hook
: 消去ファイルモードのバッファー内で呼び出されるフック。

8.14 消去ファイルの変換

あなたが古い消去ファイルをどっさり持っているのであれば、それらをスコアファイルに変換したくなるでしょう。もしそれらが「普通の」ものであれば、gnus-kill-to-score.el パッケージを使うことができます。そうでなければ、手で変換しなければならないでしょう。

消去ファイルからスコアファイルへの変換パッケージは、標準では Emacs には含まれていません。それは Gnus の配布の contrib ディレクトリー、またはhttp://heim.ifi.uio.no/~larsi/ding-various/gnus-kill-to-score.el から入手することができます。

あなたの消去ファイルが非常に複雑なのであれば---それに`gnus-kill` 形式以外のものがたくさん含まれているのなら、それらを手で変換しなければならないでしょう。あるいは、単そのままにしておいてください。Gnus は以前と同様にそれらを使ってくれるでしょう。

8.15 上級スコア付け

表題や From ヘッダーにスコアを付けるのは十分素敵ですが、本当に興味があるのが、特定の表題に関してある人が言っていることだけだった場合はどうすれば良いでしょう? もしくは、A さんが B さんにフォローアップしているときは彼女が言っていることを読みたくないけれど、C さんにフォローアップしているときは何を言っているかを知りたいという場合は?

上級スコア規則を使えば、どんな複雑なスコアのパターンでも作成することができます。

8.15.1 上級スコア付け構文

普通のスコア規則では、規則の最初の要素が文字列です。上級スコア付け規則では、最初の要素はリストです。二番目の要素は、最初の要素が`nil` でない値として評価されたときに適用されるスコアです。

これらのリストは三つの論理演算子、一つのリダイレクト演算子(訳注: 本文では間接演算子と表記されています)、および様々な合致演算子で構成することができます。

論理演算子:

`&`
`and`
: この論理演算子は、それぞれの引数を順に評価して、ある評価の結果が`false` になったら停止します。すべての引数が`true` の値に評価された場合、この演算子は`true` を返します。

`|`
`or`
: この論理演算子は、それぞれの引数を順に評価して、ある評価の結果が`true` になったら停止します。どの引数も`true` でなかったら、この演算子は`false` を返します。

`!`
`not`
`¬`
: この論理演算子はたった一つの引数を取ります。その引数の値の論理否定を返します。

スコア付けされている現在の記事の先祖たちに対して、その引数群を適用する「間接演算子」があります。例えば`1-` は、現在の記事の親にもスコア規則を適用します。`2-` は現在の記事の祖父母にスコア規則を適用します。代わりに`^^` を書くこともでき、`^` (caret== キャレット) の数でどのくらい祖先の記事までさかのぼるかを示します。

最後に合致演算子があります。これらが本当の仕事をするものです。合致演算子はヘッダー名の文字列で、その後に合致と合致の型が続きます。典型的な合致演算子は‘`("form" "Lars Ingebrigtsen" s)`’ のようなものです。ヘッダー名は単純なスコア付けをするときのものと同じで、合致の型も同じです。

8.15.2 上級スコア付けの例

以下の例はスコアファイルの規則であることに注意してください。それらを使って完璧なスコアファイルを作るには、別の括弧の組でそれらを囲んでください。

Lars が Gnus に関して話をしているときに、彼によって書かれた記事のスコアを増やしたいとしましょう:

```
((&
  ("from" "Lars Ingebrigtsen")
  ("subject" "Gnus"))
 1000)
```

ふん、簡単すぎるかな?

彼が長い記事を書くとき、時々何か素敵なことを言います:

```
((&
  ("from" "Lars Ingebrigtsen")
  (|
   ("subject" "Gnus")
   ("lines" 100 >)))
 1000)
```

しかし、彼が Reig Eigil Logge によって書かれたものに反応しているときは、彼が書いたものを読みたくありません:

```
((&
  ("from" "Lars Ingebrigtsen")
  (1- ("from" "Reig Eigil Logge")))
 -100000)
```

Redmondo が消えた靴下について書いたときにフォローアップしたすべての人のスコアが上げられますが、それは彼らが白い靴下について語っているときのみです。しかし Lars が靴下について話をしているときは、たいていあまりおもしろくありません:

```
((&
  (1-
   (&
    ("from" "redmondo@.*no" r)
    ("body" "disappearing.*socks" t)))
  (! ("from" "Lars Ingebrigtsen"))
  ("body" "white.*socks"))
 1000)
```

大量の記事が流れているグループを読んでいて、返答にしか興味が無いとしましょう。そういう場合にやることは、"Re:"、"Fw:" または"Fwd:" で始まる表題を持っていないすべての記事のスコアを下げて、返答の印で始まる表題を持っている記事のすべての親のスコアを上げることです。

```
((! ("subject" "re:\\|fwd?:" r))
  -200)
((1- ("subject" "re:\\|fwd?:" r))
  200)
```

可能性は無限大です。

8.15.3 上級スコアのちょっとした秘訣

`&` と `|` 論理演算子は、無意味な処理を迂回する論理(原典: short-circuit logic) に基づいて動作します。すなわち、その処理の結果が明らかになった時点で、引数を処理することを止

めます。例えば& の引数の一つがfalse に評価されると、残りの引数を評価する意味がありませんから。これは遅い合致(‘body’ や‘header’) を最後に持ってきて、速い合致(‘from’ や‘subject’) を最初に持ってくるべきであることを示唆します。

間接演算子(1- など) は、それらの引数をスレッドの一世代前に作用させます。次のようなことをすると:

```
...
(1-
 (1-
  ("from" "lars")))
...
```

これは「現在の記事の祖父母の from ヘッダーでスコアを付ける」ということを意味します。間接演算子の処理はとても速いのですが、以下のやり方の方が:

```
(1-
 (&
  ("from" "Lars")
  ("subject" "Gnus")))
```

次のものより良いです:

```
(&
 (1- ("from" "Lars"))
 (1- ("subject" "Gnus")))
```

8.16 スコアを減衰させる

スコアは(特に適応スコアを使っていると) 際限無く膨れ上がる傾向があることに気が付くでしょう。スコアが大きくなりすぎると、それらはすべての意味を失います---それらは単に最大値に達してしまうので、意味のある方法で使うことは難しくなります。

Gnus はこの問題の解決を助けるためにスコアを減衰させる機構を提供します。スコアファイルが読み込まれて、gnus-decay-scores がnil ではないと、Gnus はスコアファイルを減衰機構に通して、すべての永続でないスコア規則のスコアを下げます。もしgnus-decay-scores が正規表現だったら、それに合致するスコアファイルだけが扱われます。例えば*adaptive* スコアファイルだけを減衰させるには、それを‘\\.ADAPT\\’’ に設定すれば良いでしょう。減衰そのものはgnus-decay-score-function 関数によって実行され、ディフォルトはgnus-decay-score です。以下はその関数の定義です:

```
(defun gnus-decay-score (score)
  "`gnus-score-decay-constant' と `gnus-score-decay-scale' に従って
SCORE を減衰させます。"
  (let ((n (- score
              (* (if (< score 0) -1 1)
                 (min (abs score)
                      (max gnus-score-decay-constant
                           (* (abs score)
                              gnus-score-decay-scale)))))))
    (floor n)))
```

gnus-score-decay-constant はディフォルトで 3、gnus-score-decay-scale は 0.05 です。これは以下のようなことを引き起こします:

1. この関数が呼ばれたときに–3 から 3 の間のスコアは 0 に設定されます。
2. 3 から 60 までの間の大きさのスコアは 3 減らされます。
3. 60 より大きいスコアはスコアの 5% が減らされます。

もしこの減衰関数がお気に召さないなら、自分用の関数を書いてください。それは減衰させるべきスコアを唯一の引数として呼ばれ、新しいスコアを整数で返さなければなりません。

Gnus は一日に一回スコアを減衰させようとします。例えば Gnus を四日間走らせていないと、Gnus はスコアを四回減衰させます。

9 検索

Gnus には特定の基準(特定の著者、特定の主題など) に一致する記事を検索するさまざまな方法があります。最も簡単な方法は、グループに入ってから制限コマンドを使って目的の記事に概略バッファーを「制限」する(see Section 4.8 [Limiting], p. 67)、または概略バッファー内のメッセージを検索する(see Section 4.27.2 [Searching for Articles], p. 118) のいずれかです。

制限コマンドと概略バッファー検索は、サーバーから既に取り込まれた記事に対して機能しますが、それらのコマンドはサーバーに追加された記事を検索しません。したがってこれらの方法は単純ですが、目的の記事が複数のグループにまたがる場合、またはグループが大きすぎてすべての記事を取り込むことが実用的でない場合は不十分です。

関連する検索エンジンを使用して、バックエンドをより徹底的に検索することができます。一部のバックエンドには独自の検索エンジンが付属しています: 例えば IMAP サーバーは独自の検索を実行します。他のバックエンド、たとえばローカルに実装された`nnmaildir` では、ユーザーが何らかの検索インデックスを手動で構築する必要があるかもしれません。バックエンドとエンジンのディフォルトの関連付けは`gnus-search-default-engines` で定義でき、エンジンはバックエンドごとに定義することもできます(see Section 9.1 [Search Engines], p. 257)。

いったん検索エンジンを設定すると、1 つまたはそれ以上のバックエンドに由来する複数のグループ内のメッセージを検索し、その結果は 1 つのグループ内に表示することができます。検索結果を保持する(訳注: 複数の) グループは nnselect バックエンドで作成され、それらは一時的または永続的のいずれも可能です(see Section 9.2 [Creating Search Groups], p. 259)。

検索の問い合わせは、検索するグループを担当するエンジンの構文を使用するか、Gnus の一般化された検索構文を使用するかの 2 つの方法のいずれかで指定できます。一般化された構文を使用するには、オプション`gnus-search-use-parsed-queries` を`nil` 以外の値に設定します。この構文の利点は、異なるエンジンによってインデックス付けされた複数のバックエンドがある場合、検索しているバックエンドを覚えておく必要がないことです—異なるエンジンによってインデックス付けされた複数のグループに対して同じ問い合わせを発行することも可能です、それも同時に。また、相対的な日付の解析や他の Emacs パッケージへの結合など、他にもいくつかの便利な機能があります。Gnus の問い合わせ言語の詳細については、See Section 9.3 [Search Queries], p. 260.

9.1 検索エンジン

特定のサーバーからメッセージを検索するには、そのサーバーに検索エンジンが関連付けられている必要があります。IMAP サーバーは独自の検索を実行します。IMAP グループを検索するのであれば追加の設定は必要ありませんが、それ以外の場合はすべて、ユーザーが使用するエンジンを手動で指定する必要があります。これは、サーバーの種類ごと、またはサーバーごとの 2 つの異なるレベルで実行することができます。

オプション`gnus-search-default-engines` は、サーバータイプごとに検索エンジンを割り当てます。その値はサーバータイプ(例: `nnmaildir` または`nnml`) を示すシンボルを検索エンジンクラスを示すシンボルに割り当てる連想リストです。組み込まれている検索エンジンのシンボルは次の通りです。

- `gnus-search-imap`
- `gnus-search-find-grep`

- gnus-search-notmuch
- gnus-search-swish-e
- gnus-search-swish++
- gnus-search-mairix
- gnus-search-namazu
- gnus-search-mu

より細かくする必要がある場合は、検索エンジンをgnus-search-engine キーを使ってサーバー定義(.gnus.el 構成ファイル、または Gnus のサーバーバッファーにあるもの) の中で指定することができます。それは次のようになるでしょう。

```
'(nnmaildir "My Mail"
   (directory "/home/user/.mail")
   (gnus-search-engine gnus-search-notmuch
     (config-file "/home/user/.mail/.notmuch_config")))
```

notmuch、namazu、mairix および mu などの検索エンジンは、動作が似ています。それらはローカルで実行可能なファイルを使ってメッセージ貯蔵物のインデックスを作成して、それらのメッセージに対してコマンドライン検索問い合わせを実行し、一致するメッセージの絶対ファイル名のリストを返します。

これらのエンジンには、いくつかの共通の設定パラメーターがあります。一般的なパラメータは次のとおりです。

`program` 実行形式の名前。ディフォルトはnotmuch やnamazu などの素のプログラム名です。

`config-file`
: このエンジンのための設定ファイルの絶対ファイル名。

`remove-prefix`
: 検索の問い合わせによって返されるファイル名から削除されるディレクトリーの部分。この絶対パスには、メッセージ貯蔵庫のトップレベルまでのすべてが含まれている必要があります。これは実際のメッセージの場所へのパスであって、検索エンジンのインデックスではないことに注意してください(Mairix の場合は、そのシンボリックリンクのディレクトリーでもありません)。

`switches` 検索プログラムに与える追加のコマンドライン引数。このパラメータの値は、文字列のリストである必要があり、引数ごとに 1 つの文字列です。

上記のオプションは、そのタイプのすべてのエンジンに設定されているカスタマイズオプションを使うか、またはサーバー設定ファイルでエンジンごとに設定するか、の 2 つの方法のいずれかで設定することができます。

カスタマイズオプションはgnus-search-*engine-parameter* というパターンで形成されます。例えば、標準ではない notmuch プログラムを使うにはgnus-search-notmuch-programを/usr/local/bin/notmuch に設定します。これはすべての notmuch エンジンに当てはまります。このようなオプションを使うエンジンは“notmuch”、“namazu”、“mairix”、“mu”、“swish-e” および“swish++” です。

または、上記のnnmaildir の例のように、オプションを Gnus のサーバー定義に直接設定することもできます。サーバーオプションはgnus-search-engine sexp の一部であって、

オプションのシンボルと値は cons セルではなく、2 つの要素で構成するリストであることに注意してください。

namazu エンジンと swish-e エンジンには、それぞれインデックスファイルを保存する場所を指定する 1 つの追加オプションがあります。namazu のための`index-directory` は、単一のディレクトリーパスである必要があります。swish-e の場合の`index-files` は、文字列のリストでなければなりません。

すべてのインデックス付き検索エンジンには、新しく到着したメッセージを含めるために検索インデックスを更新する独自の方法があります。Gnus は現在、このための便利なインターフェースを提供していないので、更新を自分で管理する必要がありますが、これは将来変更される可能性があります。

最後に、すべての検索エンジンは`raw-queries-p` オプションを受け入れます。これは、このタイプのエンジン(またはこの特定のエンジン) が常に生の、解析されない問い合わせを使うべきであることを示しています(see Section 9.3 [Search Queries], p. 260)。

9.2 検索グループを作る

グループバッファーで*G G* をタイプすると`gnus-group-read-ephemeral-search-group` を呼ぶことによって現在行のグループを検索します。これは検索語の入力を促して、それに合致する記事を含む一時的な`nnselect` グループを作り、それらの記事がある概略バッファーを表示します。そうしたら、後はいつものコマンドを使って記事を読んだり、移動したり、削除したりすれば良いのです。

このように作られた`nnselect` グループは一時的(`ephemeral`) なものなので、そのグループを出ると消失します。ただしグループで行われた変更は、記事が作成された実際のグループに永続的に反映されます。概略バッファーを出た後でも残る永続的 (*persistent*) なグループを作成したい場合は`gnus-group-make-search-group` (*G g* にバインド) を呼び出してください。

永続(`persistent`) グループとは違って、一時的(`ephemeral`) グループはディフォルトでは終了時の記事の期限切れの処理を実行しません。一時的(`ephemeral`) 検索グループで期限切れの処理を起こさせたい場合は、変数`nnselect-allow-ephemeral-expiry` をカスタマイズしてください。すべての場合において、期限切れの処理は基になるグループの期限切れパラメーターを使います。

そうすると、一時的な検索ではなく、あなたが望むようにその結果がとてもすてきな、永続的な検索を実行したいでしょう? 問題ありません。一時的な概略バッファーから`gnus-summary-make-group-from-search` (*C-c C-p* にバインド) を呼び出せば、そのようなグループを作ることができます。

ときには元のグループで検索して見つかった記事を表示すると便利でしょうね。*A W* (つまり`gnus-warp-to-article`) で、現在行の記事の元の記事にワープ (ジャンプ) することができます。

現在行のグループだけでなく、他のグループも検索したいですって? 問題ありません。単に検索したいグループにプロセス印を付けてください。さらにもっと? それならトピックの見出しにカーソルを置いて検索を開始すれば、その見出しの下のすべてのグループを検索してくれるでしょう。

まだ足りないんですか? よろしい、サーバーバッファーで`gnus-group-read-ephemeral-search-group` (ここでは*G* に割り当てられています) を使えば、現在行のサーバーにあるすべてのグループを検索できます。多すぎますか? 検索する際に spam グループのようなグル

ープを無視したいということですね? であればnnir-ignored-newsgroups をカスタマイズしてください。この正規表現に合致しないグループは検索されなくなります。

9.3 検索構文

Gnus は、サポートされているすべての検索エンジンで使うことができるオプションの統合検索構文を提供します。これは、さまざまな検索構文を覚えておく必要がないので便利です。異なるエンジンによってインデックス付けされた複数のグループにマークを付けて、それらに対して 1 回の検索を発行することもできます。

これを有効にするにはオプションgnus-search-use-parsed-queries をnil 以外に設定してください— ディフォルトではnil です。nil 以外であっても、エンジンのクラスまたは単一のエンジン(see Section 9.1 [Search Engines], p. 257) に解析をやめさせることが可能ですし、どんな検索コマンドに対してでも接頭引数を与えることによって単一の検索をやめさせることができます。

検索構文はかなり単純です: キーと値はコロンで区切り、複数の単語の値は引用符で囲んでください。‘‘and” は暗黙的です。‘‘or” は明示的です。‘‘not” (またはキーの前に“-” を付けること) は後続の式を否定し、括弧を使って副論理節をグループ化することができます。例えば:

```
(from:john or from:peter) subject:"lunch tomorrow" since:3d
```

構文はさまざまなエンジンで受け入れられるように作成されているため、有効かどうかに関係なく、ほとんどの入力を問題なく受け入れます。一部の用語は、一部のエンジンにのみ意味があります。他のエンジンはそれらを静かに無視します。

キーの補完はTAB で提供されますが、解析中に展開される短縮キーを使用して問い合わせを入力することもできます。キーがあいまいな点まで省略されている場合(例えば‘‘s:” は‘‘subject:” または‘‘since:” である可能性があります) はエラーが発生します。

サポートされているキーには、通常のすべてのメールヘッダー(‘‘from’’、‘‘subject’’、‘‘cc” など) が含まれます。その他のキーは次のとおりです:

‘body’ メッセージの本文。

‘recipient’
‘to or cc or bcc’ と等価。

‘address’ ‘from or recipient’ と等価。

‘id’ ‘message-id’ キーと‘id’ キーはこれと等価。

‘mark’ ‘flag’、‘seen’、‘read’、‘replied’、または Gnus が一文字で表現するどんな印も、例えば‘mark:R’ を‘read’ として許容します。

‘tag’ IMAP では‘keyword’、notmuch では‘tag’ として解釈されます。

‘attachment’
添付ファイルの名前に合致します。

‘before’ 日付より前; 日付の解析は下記参照。

‘after’ 日付より後; ‘since’ も使えます。

‘thread’ メッセージのスレッド全体(個々のメッセージではない) を返します。

‘raw’ この特定の検索を解析しない。

‘limit’ 結果をこの数のメッセージに制限します。複数のグループを検索する場合は並べ替えの前に制限が行なわれるので、望ましくない結果を生じる可能性があります。

‘grep’ mairix などの「ローカルインデックス」エンジンにのみ適用されます。grep コマンドがあるシステムでは、この項の値を grep 正規表現として使用して、結果をさらに濾過します。

ELisp ベースの連絡先管理パッケージ(例: BBDB または EBDB) の補完テーブルを変数gnus-search-contact-tables に追加すると、‘from’ や‘to’ などのキーの連絡先名とアドレスの自動補完が可能になります。

9.3.1 日付の値の解析

日付タイプのキー(gnus-search-date-keys を参照) は、さまざまな値を受け入れます。第一にparse-time-string が解析できるものはすべて許容します。値が無い日付は、その最新のものとして解釈されます。例えば“march 03” は最新の 3 月 3 日です。“3d” (3 日前) などの相対的な仕様も使用することができます。この形式は w、m、および y も受け入れます。

永続的な検索グループを作成する場合、検索は解析されずに保存され、グループが更新されるたびに再解析されます。したがって、次のような問い合わせを持つ永続的な検索グループには

```
from:"my boss" mark:flag since:1w
```

常に過去 7 日間のメッセージのみが含まれます。

9.4 nnmairix

(訳注: Mairix はASCII 文字しかサポートしません。)

mairix はnnselect とともに永続的な検索グループを含む通常の検索エンジンとして使用できるので、この項は現在ではほとんど廃れています。

この項は、メールに索引を付けて Gnus 内で検索するために、どうやって mairix とバックエンドnnmairix を設定するかを説明します。さらに mairix 検索に結び付けられて自動的に更新される恒久的な“賢い” (smart な) グループを作ることができます。

9.4.1 メール検索エンジン mairix について

Mairix はローカルに格納されたメールに索引を付けたり単語を検索するための道具です。書いたのは Richard Curnow で GPL でライセンスされます。Mairix は最もポピュラーな GNU/Linux の配布に付属していますが Windows (ただし cygwin を併用)、macOS および Solaris のもとでも動作します。ウェブサイトはhttp://www.rpcurnow.force9.co.uk/mairix/index.html です。

swish++ や namazu のような他の検索ツールほどには mairix は融通がきかないかもしれませんが、それには信じられないほど速いというすばらしい特長があります。現在のシステムでは 1 秒で何千通ものメールのヘッダーと記事のボディーの隅々までを容易に捜すことができます。検索するために必要なデータベースの構築には 1〜2 分かかるかもしれませんが、一度それを完全に行なえば良いのです。それ以後、更新は追加的(インクリメンタル)に行なわれることもあって、本当に速いのです。付け加えておくと、mairix の設定はとてもやさしいです。

しかし最高速で動かすために mairix はMaildir またはMH の形式(これはnnml バックエンドを含みます) で格納されたメールで使わなければなりません。もっとも mbox でも動作

するのですけれど。Mairix は実際のメッセージファイルを指し示すシンボリックリンクを「仮想」の maildir/MH フォルダーに置くことによって検索結果を提示します(mbox を使っている場合はコピーが作られます)。Mairix はそのような仮想フォルダーにすでに検索結果を提示しているので、あるメール検索の結果を提示する「賢い」メール・フォルダーを作成するために外部のプログラムとして使用するのに非常によく適しています。

9.4.2 nnmairix を使うために必要なこと

Mairix はローカルなメールを検索します---つまり mairix はメール・フォルダーを必ず直接にアクセスできなければなりません。もしメールが別のサーバー(例えばIMAP サーバー)にあって、たまたま shell でアクセスすることができるのならば、`nnmairix` は例えば ssh を介して mairix を遠隔で動作させることもできます。

加えて、`nnmairix` は Gnus のバックエンド`nnml`、`nnmaildir` および`nnimap` だけをサポートします。`nnmairix` を使うには、必ずこれらのバックエンドの一つを使わなければなりません。`nnmbox`、`nnfolder` または`nnmh` のような他のバックエンドでは動作しないでしょう。

もしどうしても mbox を使わなければならなくて、それでも`nnmairix` を使いたいのならば、ローカルなIMAP サーバーを立ち上げることによって`nnimap` を介してアクセスすることができます。これはいくつかの mbox ファイルにアクセスするためにしてはかなり大がかりな作業になるので、もう MH か Maildir に変えてしまいましょう。それでも mbox を使うことに本当に本当に情熱を持っているのなら、Emacs に付属している`mairix.el` パッケージを研究する必要があるでしょう。

9.4.3 nnmairix は実際に何をするのか?

バックエンド`nnmairix` は、ある検索語で mairix に尋ねたりデータベースを更新させるために Gnus から mairix を呼ぶことを可能にします。概略バッファーでメッセージを見ている間、あらかじめ用意されている mairix を呼ぶためのいくつかのショートカットを使うことができます。例えば現在見ているメッセージの送信者からのすべてのメールをすばやく探したり、メールが異なったフォルダーにあっても、そのメッセージに関連している全体のスレッドを表示するなどです。

さらに、ある mairix 検索に結び付いた恒久的な`nnmairix` グループを作ることができます。この例では、ある送信者から来て、かつ、ある表題行(または Message-ID に基づく一つの特定のスレッドさえ) を持つメールを含むグループを容易に作ることができます。これらのフォルダーで新しいメールをチェックする(例えば*g* または*M-g* を押す) と、mairix を呼ぶことによってそれら自体が自動的に更新します。

Mairix はすでにグループを作っていて、Gnus でアクセスできるようにそれらのメールへのリンクを用意しているのに、いったいなぜ`nnmairix` が必要なのかと尋ねるかもしれませんね。そうではありませんか? えーと、それは動くかもしれませんが、だめなことが多いでしょう---問題無しには。最もありそうなのは、記事数がおかしくなって、しかも時々は Gnus があるはずだと言い張るメールがすでにキャンセルされていてアクセスできないことを思い知らされることでしょう。これは、ものごとが Gnus の後ろに隠れて起こっているとき、Gnus は本当は不満に思っているという事実のためです。例えばIMAP サーバーで mairix を使っているのなら、もう一つの問題はメールバックエンド自体であるかもしれません(mairix が検索グループの内容を変えたとき、Dovecot は間違った索引ファイルについて私に不平を言いました)。`nnmairix` を使えば、これらの問題は回避されるはずです。

実は`nnmairix` はメールバックエンドではありません---それは mairix が検索結果を格納する「本当の」メールバックエンドと Gnus フロントエンドの間に位置していて、むしろ実

際にはラッパーに似ています。Mairix フォルダーのために三つの異なるメールバックエンドnnml、nnmaildir またはnnimap の中から選ぶことができます。nnmairix は検索結果をこのメールバックエンドのzz_mairix-<NAME>-<NUMBER> という名前のフォルダーに格納するために mairix バイナリーを呼びますが、それらのフォルダーは Gnus フロントエンドには名前が<NAME> だけになって渡されます。すでにメールを格納している既存のメールバックエンドを使うことができますが、あなたの他のメールと並べて新しいメールグループを作るnnmairix が気持ち悪いのであれば、例えば新しいnnmaildir またはnnml サーバーを mairix 専用に作ることもできます。ただし、それらのサーバーが間違って新着メールを取り込んでしまわないようにしてください(see Section 9.4.9 [nnmairix caveats], p. 269)。もしnnimap とともにIMAP サーバーで mairix をリモートに使いたいのであれば、特別な事情が生じます---ここでは mairix フォルダーと他のメールが同じnnimap バックエンド上になければなりません。

9.4.4 mairix の設定

まずはメールフォルダーのバックアップを作りましょう(see Section 9.4.9 [nnmairix caveats], p. 269)。

Mairix の設定は簡単です。(少なくとも) 以下のエントリーを含む.mairixrc ファイルを作ってください:

```
# Your Maildir/MH base folder
base=~/Maildir
```

これはすべてのメールの基点になるフォルダーです。以下のすべてのディレクトリーはこのフォルダーを基点に相対的な値をとります。nnmairix をnnimap で使いたい場合、この基点のディレクトリーはIMAP サーバーがメールフォルダーを格納するメールディレクトリーを表すものでなければなりません!

```
maildir= ... 索引を付ける maildir フォルダー ...
mh= ... 索引を付ける nnml/mh フォルダー ...
mbox= ... 索引を付ける mbox フォルダー ...
```

Mairix で索引を付けたいすべてのメールフォルダーと mbox ファイルをこれで(基点ディレクトリーへの相対値で!) 指定します。nnml バックエンドはメールを MH 形式で保存するので、それらのディレクトリーをmh 行に置いておかなければならないことに注意してください。さらに詳しいことについては、この章の最後にある例と mairixrc の man ページを見てください。

```
omit=zz_mairix-*
```

これは mairix の検索結果に偶然に索引付けをしてしまわないためのものです。これらのフォルダーの接頭辞は、変数nnmairix-group-prefix で変えることができます。

```
mformat= ... 'maildir' または 'mh' ...
database= ... データベース・ファイルの置き場所 ...
```

mformat の設定は mairix 検索フォルダーへの出力形式を指定します。検索結果にnnml でアクセスしたい場合は、これをmh に設定してください。さもなければmaildir を選びましょう。

要約するために、私の.mairixrc ファイルを短くしたものを例に挙げましょう:

```
base=~/Maildir
maildir=.personal:.work:.logcheck:.sent
mh=../Mail/nnml/*...
```

```
mbox=../mboxmail/mailarchive_year*
mformat=maildir
omit=zz_mairix-*
database=~/.mairixdatabase
```

この場合、基点のディレクトリーは`~/Maildir` で、そこに私のすべての Maildir フォルダーが格納されています。ご覧のようにそれぞれのフォルダーはコロンで区切られています。なぜどのフォルダーもドットで始まるのかって? それは私がIMAP サーバーとして Dovecot を使い、さらにそれは`Maildir++` フォルダーを使うからです。`nnmairix` をテストするために`~/Mail/nnml` にセーブされている`nnml` のメールもいくつか持っています。これは`base` ディレクトリーへの相対値で指定しなければならないので`../Mail` の表記が必要です。`*...` で終わる行は、このディレクトリーにあるすべてのファイルを再帰的に走査するためであることに注意してください。三個のドット無しのワイルドカード`*` では再帰的に働きません。さらに私は`~/mboxmail` のあたりにアーカイブされたメールを含む古い`mbox` ファイルを持っています。その他の行の意味は言うまでもないですね。

詳細およびさらなるオプションについては`mairixrc` の man ページを見てください。とりわけワイルドカードの使い方は、今まで使っていたのとは少し違うでしょう。

さあ、それでは最初に`mairix` を実行して索引を作りましょう。これには数分かかるでしょう。でもあらゆる索引が追加的(インクリメンタル) に更新を行なうので非常に速いです。

9.4.5 nnmairix バックエンドの設定

グループモードで*G b c* (`nnmairix-create-server-and-default-group`) をタイプしてください。これは必要なすべての情報を質問して、`nnmairix` サーバーを外部(foreign) グループとして作成します。以下を指定する必要があるでしょう:

- `nnmairix` サーバーの**名前** です---好きなものを選んでください。
- Mairix がその検索結果を格納する**バックエンド・サーバー** の名前です。これは`nnml:mymail` のような完全なサーバー名でなければなりません。今のところ`nnmaildir`、`nnimap` および`nnml` でアクセスするサーバーがサポートされています。上で説明したように、ローカルに格納されるメールのためには、あなたがメールを格納している既存のサーバーにすれば良いでしょう。しかし`nnmairix` 専用に、例えば新しい`nnmaildir` または`nnml` サーバーを第二の(secondary) 選択方法に加えることもできます(see Section 2.1 [Finding the News], p. 3)。もしまさに第二の`nnml` サーバーを mairix 用に使っているのであれば、サーバー変数`nnml-get-new-mail` を確実に`nil` に設定してください。さもないとメールを失うことになりかねません(see Section 9.4.9 [nnmairix caveats], p. 269)。Mairix をIMAP サーバー上で遠隔動作させたいのならば、そこでそれに対応する`nnimap` サーバーを選ばなければなりません。
- Mairix バイナリーを呼ぶための**コマンド** です。通常これは単に`mairix` で良いのですが、例えばIMAP サーバー上で mairix を遠隔動作させたいのであれば、`ssh SERVER mairix` のようなものにすることもできます。Mairix へのディフォルトのオプションを加えたい場合、それをここでやっても構いません。でも、代わりに変数`nnmairix-mairix-search-options` を使う方が良いです。
- **ディフォルトの検索グループ** の名前です。これは mairix のすべての検索結果、つまり恒久的な`nnmairix` グループに結び付けられないすべての検索結果を格納するグループです。好きなものを選んでください。
- もしメールバックエンドが`nnimap` か`nnmaildir` だったら、**Maildir++** つまり隠された(= '.' で始まる) maildir フォルダーを使うかどうかを尋ねられるでしょう。例えば

Dovecot IMAP サーバーを使っている場合は、そこで‘yes’ と回答しなければなりません。それ以外の場合は‘no’ と答えるべきです。

9.4.6 nnmairix で利用できるショートカットキー

グループモードで:

G b c nnmairix サーバーと、このサーバーのためのディフォルトの検索グループを作ります(nnmairix-create-server-and-default-group)。これはすでに行なっておいてある必要があります(see Section 9.4.5 [Configuring nnmairix], p. 264)。

G b s Mairix に送られる検索語を与えます。検索結果はディフォルトの検索グループに格納され、それは自動的に表示されます(nnmairix-search)。

G b m Mairix の検索または恒久的なグループをもっと快適に、常にグループのカスタマイズに似たグラフィックなウィジェットを使って作るようにします。それがどんなものかを知るには、まずは試してみてください(nnmairix-widget-search)。

G b i もう一つの快適な mairix 検索コマンドですが、ミニバッファーしか使いません(nnmairix-search-interactive)。

G b g 検索に関連させられた恒久的なグループを作ります(nnmairix-create-search-group)。このグループを*g* または*M-g* で更新するときにnnmairix バックエンドは自動的に mairix を呼びます。

G b q カーソル位置のnnmairix グループの検索条件を変更します(nnmairix-group-change-query-this-group)。

G b t カーソル位置のnnmairix グループの’スレッド’ パラメーターをトグルに切り替えます。つまり見つかったメッセージのすべてのスレッドを見たい場合に使います(nnmairix-group-toggle-threads-this-group)。

G b u データベースを更新するために mairix バイナリーを呼びます(nnmairix-update-database)。ディフォルトのパラメーターは、これをできるだけ速くするための-F および-Q です(これらのディフォルトのオプションを定義するには変数nnmairix-mairix-update-options を見てください)。

G b r このnnmairix グループの記事を常に既読または未読に保つか、または印を変更しないようにします(nnmairix-group-toggle-readmarks-this-group)。

G b d 「本当の」メールバックエンドでnnmairix グループを作り直します(nnmairix-group-delete-recreate-this-group)。nnmairix グループの記事数がいつも間違っている場合に、これを行なうことができます。

G b a カーソル位置のnnmairix グループのためのallow-fast パラメーターをトグルに切り替えます(nnmairix-group-toggle-allowfast-this-group)。ディフォルトのnnmairix の動作は、グループを更新したり入ったときに mairix の検索を行なうことです。allow-fast パラメーターが設定されると、グループ入ったときではなく、明示的にグループを更新するときだけ mairix が呼ばれます。これはグループに入るときに速くなりますが、更新したときとまだ mairix データベースに無いグループに入るときの期間に何かが変化した場合に、実体の無いシンボリックリンクを生じさせるかもしれません。

G b p このグループの印を伝搬させるかどうかをトグルに切り替えます(nnmairix-group-toggle-propmarks-this-group)。(see Section 9.4.7 [Propagating marks], p. 266)。

G b o 印を主動で伝搬させます(nnmairix-propagate-marks)。これはnnmairix-propagate-marks-upon-close がnil である場合だけ必要です。

概略モードで:

G G m 現在のメッセージに基づいた mairix の検索またはグループを、グラフィックなウィジェットを使って作ります(nnmairix-widget-search と同じです) (nnmairix-widget-search-from-this-article)。

G G g 現在のメッセージに基づいた検索条件で新しい検索グループを対話的に作ります。グラフィックなウィジェットの代わりにミニバッファーを使いますが(nnmairix-create-search-group-from-message)。

G G t 現在の記事のスレッドを捜します(nnmairix-search-thread-this-article)。事実上これは現在の記事の'm:msgid' でnnmairix-search を呼んでスレッドを得るためのショートカットです。

G G f 現在の記事の送信者からのすべてのメッセージを探します(nnmairix-search-from-this-article)。これは'f:From' でnnmairix-search を呼ぶためのショートカットです。

G G o (これが使えるのはnnmairix グループだけです!) この記事に正しい投稿様式(posting styles) とグループパラメーターを適用して返信するために、この記事が元々来たグループを特定して、そのグループでその記事を表示しようとします(nnmairix-goto-original-article)。この関数はもし利用可能ならレジストリーを使いますが、代替手段として記事ファイルの名前を分析することもできます。

G G u 元の記事から、もしあれば可視記事の印(tick mark) を取り除きます(nnmairix-remove-tick-mark-original-article) (see Section 9.4.8 [nnmairix tips and tricks], p. 268)。

9.4.7 nnmairix グループの印を伝搬させる方法

初めに: 印を伝搬させる機能を効率良く使うためには、実際にはパッチを当てた mairix のバイナリーが必要です。そうしないといつも mairix データベースを更新しなければならないでしょう。パッチはここで手に入ります:

```
http://www.randomsample.de/mairix-maildir-patch.tar
```

このパッチには mairix v0.21 のソースコードが必要です。それに付いている readme ファイルでどんなことも説明されています。印の伝搬を使わなくても良いと思うならこれらのパッチを当てなくても構いませんが、それでもなお、それらは maildir フラグの変更にまつわるやっかいごとを修正してもくれるので有用でしょう。

パッチを当てた mairix のバイナリーとともにnnmairix をメール分割(see Section 7.4.6 [Fancy Mail Splitting], p. 175) の代わりとして使うことができます。例えば'david@foobar.com' からのすべてのメールをあるグループに放り込む代わりに、単に'f:david@foobar.com' を探す検索グループを作ることができます。実のところこれが「賢いフォルダー」の肝心なところで、単にすべてを一つのメールフォルダーに放り込んだら、分割する代わりに動的

に検索結果を作るのです。これは、したいときはいつでもフォルダーを変更できるので、より融通が効きます。このことは、あなたが実際のメールグループの代わりに、いつもはnnmairix グループにあるメールを読むであろうことも暗示します。

しかし、一つ問題があります。'david@foobar.com' から新しいメールを受け取ったとすると、それは二つのグループに現れるのです。「実際」のグループ(例えば INBOX)およびnnmairix 検索グループです(後者は、もし mairix データベースを更新してあれば)。nnmairix グループに入ってそのメールを読むと既読の印が付きますが、それはnnmairix グループでだけです---「実際」のメールグループでは未読のままです。

そのメールグループをキャッチアップする(すべての記事を既読にする) ことはできます。しかしこれは退屈だし、そのためのnnmairix グループを作っていないメールを見落とすかもしれない点で事故を起こしやすいでしょう。もちろん最初にnnmairix-goto-original-article (see Section 9.4.6 [nnmairix keyboard shortcuts], p. 265) を使ってから元のグループでそのメールを読むことはできますが、それはもっとやっかいでしょう。

明らかに、元の記事にどうにかして印を自動的に付けることできれば、それが最も楽なやり方でしょう。これがまさに印の伝搬 が行なおうとしていることです。

ディフォルトでは印の伝搬はしないようになっています。あるグループのためのそれはnnmairix-group-toggle-propmarks-this-group (*G b p* にバインドされています) で有効にすることができます。この関数はディフォルトの検索グループで使おうとすると警告を発します。なぜかと言うとディフォルトの検索グループは一時的な検索のためにあるので、印が偶然にこのグループから伝搬してしまいやすいからです。もっとも本当にそれをやりたいのなら、その警告を無視することはできます。

印の伝搬を有効にしてあるとnnmairix グループで付けたすべての印が元のグループに伝搬するはずです。例えば記事に可視(tick) の印を(ディフォルトでは*!* で) 付けると、この印は魔法のように元の記事にも付くはずです。

あなたが知っている必要がある、または無い、さらなることがらがあります:

印はすぐにではなく、グループを閉じたときだけ付けられます。これは印の伝搬を性急に行なわない以外に maildir ファイルを扱うときのシンボリックリンクにまつわる問題の回避もします(フラグの変更はファイル名の変化を伴うので)。印の伝搬をnnmairix-propagate-marks-upon-close を通じて制御することもできます(詳しくは変数の説明を見てください)。

当然ながら、あなたが印を付けたいあらゆる記事のためにnnmairix は元のグループを調べなければならないでしょう。元のグループを特定するためにnnmairix は、もし使えるなら最初にレジストリーを使います。レジストリーは非常に速いです。したがって印の伝搬を使うなら本当にレジストリーを使えるようにすべきです。本当に。RAM とディスクの容量に心配が無いならgnus-registry-max-entries を十分に大きな値に設定してください。大事を取るためには、おおよそ mairix で索引を付けるメールの量を選んでください。

レジストリーを使いたくない、またはレジストリーがまだ元の記事を見たことが無い場合、その記事のファイル名を特定するためにnnmairix は追加の mairix 検索を行ないます。もちろんこれはレジストリーより遅くなります---もしこのやり方で数百ないしは数千の印を付けると、いくらか時間がかかるかもしれません。この状況はnnmairix-only-use-registryをt に設定することによって避けることができます。

おそらくあなたは逆方向にも印を伝搬させたいでしょう。つまり「本当の」メールグループで記事に可視(tick) 印を付けたらnnmairix グループにある同じ記事も可視になって欲しいということです。いくつかのもっともな理由により、これは maildir を使う場合だけ効率的に行なうことができます。すぐに矛盾したことを言いますが、それはnnmaildir で

は働きません。なぜならnnmaildir は印を外部のしかもファイルではないところに格納するからです。したがってnnmairix グループへの印の伝搬は、通常はファイル形式としてmaildir を使うIMAP サーバーの場合だけ働きます。

今この設定作業を行なっているのならnnmairix-propagate-marks-to-nnmairix-groupsをt に設定して何が起きるか見てください。もしあなたが見るものが好きではないならば、再びそれをnil に戻しましょう。一つの問題は未読記事の数を間違えることかもしれません。これは元のグループで記事を消去したり、または期限切れ消去されたときに普通に起きます。これが起きたらそのバックエンドで*G b d* を使って、そのnnmairix グループを作り直してください。

9.4.8 nnmairix のヒント、こつ、およびいくつかの例

- メールのチェック
  私は大事なメールのグループをグループレベル 1 にしています。Mairix グループのグループレベルは 5 なので、それらは起動時にチェックされません(see Section 3.6 [Group Levels], p. 19)。
  メールをチェックするために私は以下を使っています:

```
(defun my-check-mail-mairix-update (level)
  (interactive "P")
  ;; 接頭引数が与えられなかったらレベルを 1 に設定する。
  (gnus-group-get-new-news (or level 1))
  (nnmairix-update-groups "mairixsearch" t t)
  (gnus-group-list-groups))


(keymap-set gnus-group-mode-map "g" 'my-check-mail-mairix-update)
```

  ‘"mairixsearch"’ の代わりにあなたのnnmairix サーバーの名前を使いましょう。詳しくはnnmairix-update-groups の説明を見てください。
- 可視(ticked) 記事のための検索グループの例
  例えばすべての可視記事のためのグループを作ることができます。そこでは記事が常に未読になっています:
  G b g をたたいてグループ名を入力し、検索条件として‘F:f’ を使ってください。そしてスレッドを含めないようにします。
  次に*G b p* を使ってこのグループのための印の伝搬を有効にしましょう。そうしたら*G b r* を二回使って「常に未読」機能を有効にします。
  これはこれで良いでしょう---しかしどうやってnnmairix グループの可視印を消したら良いでしょうか? それには二つの選択肢があります: nnmairix-remove-tick-mark-original-article (*G G u* にバインドされています) を使って元の記事から可視印を取り除くことができます。もう一つの可能なことはnnmairix-propagate-marks-to-nnmairix-groups をt に設定することです。しかし上記のこのオプションに関するコメントを見てください。もしそれが動作するなら可視印はnnmairix グループにあるはずで、それらはいつものように、例えば記事を既読にすることによって取り除くことができます。
  元の記事から可視印を取り除くと、mairix データベースを更新しかつグループを更新した後で、その記事はnnmairix グループから消え去るはずです。幸いにまさにそれを行なうための関数があります: nnmairix-update-groups です。詳しくは前述のコードの断片と関数の説明を見てください。

- メールグループの自動購読の取り扱い
  先に説明したように、すべてのnnmairix グループは実際には‘zz_mairix-<NAME>-<NUMBER>’ の様式でメールバックエンドに格納されます。それらはサーバーバッファーでそのバックエンドに入ると見ることができます。これらのグループを購読してはいけません! 残念ながらこれらのグループはnnmaildir またはnnml を使うと通常自動購読 になります。つまり‘zz_mairix*’ 様式のグループが突然グループバッファーに現れるのを見ることになります。もしこれが起こったら単に*C-k* でそれらのグループを kill してください。これを避けるにはgnus-auto-subscribed-groups をnil に設定して自動購読を恒久的に無効にする(see Section 2.4.3 [Filtering New Groups], p. 7, か、またはもしこの機能を維持したいのであれば‘zz_’ ではじまるすべてのグループでそれを無効にする以下のその場しのぎの手を使ってください:

```
(setq gnus-auto-subscribed-groups
      "^\\(nnml\\|nnfolder\\|nnmbox\\|nnmh\\|nnbabyl\\|nnmaildir\\).*:\\([^z]\\|z$\
```

9.4.9 nnmairix でさらに知っておく必要があること

- 第二のnnml サーバーを、まさに nnmairix のために作ることができます。しかしその場合、対応するサーバー変数nnml-get-new-mail を明示的にnil に設定する必要があります。そうしておかないと新着メールがこの第二のサーバーに取り込まれてしまうかもしれません(そして、それらを二度と見ることはないでしょう)。サーバー定義の例です:

  ```
  (nnml "mairix" (nnml-directory "mairix") (nnml-get-new-mail nil))
  ```

  (nnmaildir にもサーバー変数get-new-mail があるのですが、それはディフォルトでnil なので、nnmaildir サーバーをまさに mairix 用に使う場合でも、それを明示的に設定する必要はありません。)
- もし Gnus レジストリーを使っているなら、nnmairix グループでレジストリーを使わないください(ディフォルトでgnus-registry-unfollowed-groups に含まれています)。もしgnus-registry-split-fancy-with-parent を使っているなら特別な注意が必要です。分割されてnnmairix グループに入ったメールは、通常そのグループの新着メールをチェックしたとたんに永久に無くなってしまいます(はい、私はそれをやってしまいました...)。
- したがって: 絶対に、断じて「本当の」メールをnnmairix グループに入れてはいけません(とにかくそれができるべきではありません)。
- もし Gnus エージェント(see Section 7.9 [Gnus Unplugged], p. 216) を使っているなら: nnmairix グループをエージェント化しないでください(もっとも、それをやったら何が起きるか私は知りませんが)。
- Mairix はUS-ASCII 文字だけをサポートします。
- nnmairix は mairix が呼ばれた後で Gnus にメールバックエンドのグループを完全に読み直させる、かなり力まかせな手段を使います---つまり単純にそのメールバックエンドのグループを消して作り直します。これまでのところ、これは何ら問題無く働き、nnmairix がそれ自身のもの以外のメールグループを削除するとは思えません。しかしとにかく本当に、メールフォルダーのバックアップを持つべきです。
- すべての必要な情報はグループパラメーターに格納されます(see Section 3.10 [Group Parameters], p. 23)。これにはアクティブファイルを必要としないという利点がありますが、それは暗にnnmairix グループを kill すると永久に失われてしまうことをも意味します。

- たくさんの`nnmairix` グループを作って kill すると、メールバックエンドのサーバーに“zz_mairix-*” という名前のグループが溜まってしまいます。もはや不要になった古いグループを削除するには`nnmairix-purge-old-groups` を呼んでください。これは当然`zz_mairix-<NAME>-<NUMBER>` の様式ではどんな「本当の」メールもフォルダーにセーブしないことを想定していることに注意してください。`nnmairix` グループの接頭辞は変数`nnmairix-group-prefix` を変更することによって変えることができます。
- 以下は先に言及した mairix のためのパッチ(see Section 9.4.7 [Propagating marks], p. 266) を使わない 場合だけに当てはまります:

  `nnmairix` を maildir フォルダーとともに使うと問題が起きる場合があります。それは maildir がメールのフラグを‘`Seen`’ または‘`Replied`’ のように、それぞれ文字‘`S`’ および‘`R`’ をメッセージのファイル名に付け加えることによって格納するという事実によります。これは、今のところ、新しいメールが届いたときだけでなくメールのフラグが変化したときも、mairix のデータベースを更新しなければならないであろうことを暗示しています。同じことが、まだ‘`new`’ フォルダーにあるうちに索引が作られたものの Gnus がそのメールを見たときに‘`cur`’ に移された新しいメールにも当てはまります。これが起きた後でデータベースを更新しないと、mairix 検索は存在しないファイルを指すシンボリックリンクをもたらします。Gnus では通常それらのメッセージはヘッダーに“(none)” が表示されて現れ、アクセスすることができません。もしこれが起こった場合、普通は*`G b u`* を使い、かつグループを更新すれば解決します。

9.5 nnir からの移行

以前の Gnus の検索エンジンは nnir と呼ばれるものでしたが、Emacs バージョン 28 で廃止されました。もし Emacs をアップグレードしたあとで`nnir-*` 変数に関する obsolete-variable の警告が表示されるようになったなら、移行するのはかなり簡単です。警告によって表沙汰になった変数だけでなく、以前のすべての検索エンジン固有の変数は、接頭辞`nnir-` を`gnus-search-` に置き換えるだけで更新できます。例えば`nnir-notmuch-program` は`gnus-search-notmuch-program` になりました。

10 いろいろ

10.1 プロセス/接頭引数

多くの関数、その中でも記事の移動、デコード、保存をするための関数は、「プロセス/接頭引数の習慣」として知られているものを使います。

これは、利用者がどの記事に命令を実行したいかを見つけるための方法です。

それはこのような感じです:

数値接頭引数が N だったら、現在の記事を含めた次の N 個の記事に対して作業を実行します。もし数値接頭引数が負だったら、現在の記事を含めた前の N 個の記事に対して作業を実行します。

`transient-mark-mode` が`nil` ではなく、リージョンが設定されていたら、リージョンにあるすべての記事で作業が行なわれます。

数値接頭引数が無くても、いくつかの記事はプロセス印が付いている場合には、プロセス印が付いている記事で作業が実行されます。

数値接頭引数やプロセス印の付いている記事が無い場合は、現在の記事でだけ作業を実行します。

これは実際とても単純なのですが、びっくりされないためにも、はっきりさせておく必要があります。

プロセス印に反応するコマンドは、現在プロセス印が付いている記事のリストをスタックに積んで、記事のすべてのプロセス印を消去します。前回の設定を*M P y* で復旧させることができます(see Section 4.7.6 [Setting Process Marks], p. 66)。

多くの人々をぎょっとさせ、恐がらせると思われることの一つは、例えば*3 d* が、本当に*d d d* と同じことをすることです。それぞれの*d* (これは現在の記事に既読の印を付けます) は、ディフォルトでは印を付けた後で次の未読記事に移動するので、*3 d* は概略バッファーがどうなっていても、次の三つの未読記事を既読にします。動作をもっと分かりやすくするには、`gnus-summary-goto-unread` を`nil` に設定してください。

多くのコマンドはプロセス/接頭引数の習慣を使いません。それをしないすべてのコマンドは、このマニュアルで明記されています。そういうコマンドにプロセス/接頭引数の習慣を適用するには、*M-&* コマンドを使ってください。例えば、そのグループのすべての記事を期限切れ消去可能として印を付けるには*M P b M-& E* とします。

10.2 利用者との相互作用

`gnus-novice-user`
: この変数が`nil` でないのは、あなたは Usenet の世界の新参者か非常に慎重な人のどちらかだというです。これは本当に良いことです。何か危険なことをする前に、「本当にこれをしたいのですか?」というような質問を受けます。これはディフォルトでは`t` です。

`gnus-expert-user`
: この変数を`nil` でない値にすれば、Gnus から質問を受けることが滅多に無くなるでしょう。これは単純に、どんな変なことをしても、あなたが何をしているかをわかっているものと見なします。例えば`.newsrc` ファイルを保存せずに Gnus を終了したり、既読情報などを更新せずにグループを抜けたり、グルー

プの記事をすべて既読にしたり、期限切れ消去される記事を消去したり、ニュース記事にメールで返信しようとしたときに、確認を求められなくなります。

`gnus-interactive-catchup`
: `nil` でないと、グループに追いつく(catchup, 未読の記事を読んだことにしてしまう) 前に、確認を求めます。ディフォルトで`t` です。

`gnus-interactive-exit`
: もし`nil` 以外だったら Gnus を終了するときに確認を求めます。`quiet` にしておくと、活きている概略バッファーを確認無しで自動的に更新します。ディフォルト値は`t` です。

10.3 シンボルの接頭引数

非常に多くの Emacs の命令が(数値) 接頭引数に反応します。例えば*C-u 4 C-f* はポイントを 4 文字先に移動し、*C-u 9 0 0 I s s p* は 900 のスコア(永続、`Suject`、文字列の一部、という規則) を現在の記事に加えます。

これはすべて素敵で良いのですが、命令にもう少し追加の情報を与えたいときはどうすれば良いのでしょう? えーと、たいていの命令がしていることは「生の」接頭引数を何らかの特別な方法で解釈することです。例えば*C-u 0 C-x C-s* は、現在の記事を保存するときにバックアップファイルを作らないで欲しいことを意味します。でも、バックアップファイルを作らないで保存するのと同時に、Emacs に閃光を放って、素敵な音楽を演奏して欲しいときはどうすれば良いでしょう? それができなくても、あなたは申し分なく幸せですね(?)。

私はそうではありません。そこで、私は二つめの接頭引数「シンボル接頭引数」を加えました。接頭キーは*M-i* (`gnus-symbolic-argument`) で、次に押される文字が値です。いくらでも*M-i* 接頭語を積み重ねることができます。*M-i a C-M-u* は「*C-M-u* 命令にシンボル接頭引数`a` を与える」ということです。*M-i a M-i b C-M-u* は「*C-M-u* 命令にシンボル接頭引数`a b` を与える」ということです。趣旨はわかりましたね。

シンボル接頭引数を受け付けない命令にそれを打ち込んでも何も悪いことをしませんが、良いことも何もしません。現在のところ、あまり多くの関数がシンボル接頭引数を利用しているわけではありません。

Gnus がこれを実装しているやり方に興味があるなら、Section 12.7.7 [Extended Interactive], p. 382, を見てください。

10.4 書法仕様変数

このマニュアルを通して、おそらく`gnus-group-line-format` または`gnus-summary-mode-line-format` のように呼ばれるたくさんの変数があることに気付いたでしょう。これらは Gnus が色々なバッファーでどのように行を出力するかを制御します。非常にたくさんのものがあります。幸運なことに、それらはすべて同じ構文を使うので、あまり嫌な目には会わないでしょう。

Gnus は Emacs のほとんどのモードで使われている font-lock の仕掛けを使わないので、Gnus のグループ/概略/記事バッファーで`font-lock-mode` を ON に切り替えても、通常は何の役にも立ちません— 代わりに Gnus がすでにバッファーに入れている face を台無しにするだけです。(これは`whitespace-mode` のような font-lock の仕掛けを使う他のマイナーモードにも当てはまります。)

書法仕様(format) 指定の例です(グループバッファーより):
‘%M%S%5y: %(%g%)\n’。それは極めて醜く、たくさんのパーセント記号がいたるところにあります。

現在のところ Gnus は以下の書法仕様変数を使います: gnus-group-line-format, gnus-summary-line-format, gnus-server-line-format, gnus-topic-line-format, gnus-group-mode-line-format, gnus-summary-mode-line-format, gnus-article-mode-line-format, gnus-server-mode-line-format, およびgnus-summary-pick-line-format。

これらすべての書法仕様変数は任意の elisp 式であることもできます。その場合、それらは要求される行に挿入するためにeval (評価) されます。

Gnus は、あなたが自分用の書法仕様指定を作っているときに、手助けをする命令を備えています。*M-x gnus-update-format* は現在の式をeval し、当の仕様を更新し、行を生成するための Lisp 式を検査することができるバッファーに移動します。

10.4.1 書法仕様の基本

それぞれの‘%’ の要素は、当のバッファーが作成されるときに何らかの文字列や他のもので置き換えられます。‘%5y’ は「‘y’ 指定を挿入して、5 文字の場所を得るために空白を詰め込みなさい」ということです。

普通の C や Emacs Lisp の書法仕様(format) 文字列と同じように、‘%’ と書法仕様の型の文字の間の数値修飾子は、常に少なくともその長さになるように、出力に(空白文字などを)「詰め込み」ます。‘%5y’ はその場所が常に(少なくとも) 5 文字の長さになるように、左に空白を詰め込みます。もし‘%-5y’ とすれば、代わりに右側に詰め込みます。

特に広い幅の値に対して保護するために、その場所の長さを制限したいこともあるでしょう。そのために‘%4,6y’ などと指定することができます。これは、その場所は決して 6 文字を超える幅にはならず、かつ 4 文字より少ない幅にもならないということです。

Gnus は‘%&user-date;’ のような、いくつかの拡張様式指示もサポートします。

10.4.2 モード行書法仕様

モード行書法仕様変数(例えばgnus-summary-mode-line-format) は、以下の二つの違い以外は、バッファー行に適応した書法仕様変数(see Section 10.4.1 [Formatting Basics], p. 273) と同じ規則に従います:

1. 最後に改行(‘\n’) があってはなりません。
2. 特別な‘%%b’ 仕様をバッファー名を表示するために使うことができます。えーと、実はそれは仕様ではないのです—‘%%’ というものは、Emacs が‘%b’ を受け取って、そのモード行表示機能に「バッファー名を表示しなさい」と解釈させるために、単に書法仕様の処理系を無傷で通り抜けることができるように‘%’ を囲う方法なのです。Emacs が理解するモード行指定の完全な一覧については、変数mode-line-format の説明文を見てください。

10.4.3 上級書法仕様

表示された領域を後で何らかの方法で処理するのは、しばしば役に立ちます。詰め込み、制限、切り取り、および特定の値の抑制は、「チルダ修飾子」を使うことによって実現することができます。よくあるチルダ仕様は、‘%~(cut 3)~(ignore "0")y’ のように見えるでしょう。

これらは有効な修飾子です:

`pad`
`pad-left` 領域の左側に、要求された長さになるまで空白を詰め込みます。

`pad-right`
領域の右側に、要求された長さになるまで空白を詰め込みます。

`max`
`max-left` 指定された長さになるように、文字列の左側を切り取ります。

`max-right`
指定された長さになるように、文字列の右側を切り取ります。

`cut`
`cut-left` 指定された数の文字を左側から切り落とします。

`cut-right`
指定された数の文字を右側から切り落とします。

`ignore` 領域が指定された値と等しい(equal) ならば、空文字列を返します。

`form` '`@`' 仕様が使われたときに、指定された式を領域の値として使います。
これは例です:

```
"~(form (current-time-string))@"
```

例を出してみましょう。概略モード行での'`%o`' 仕様は ISO0861 様式の凝縮された日付('`19960809T230410`' のようなもの) を返します。これはとても発音しにくいので、世紀を表す数と時刻を削ぎ落として、6 文字の日付を残したいと思います。それは'`%~(cut-left 2)~(max-right 6)~(pad 6)o`' となるでしょう。(切り落とし(cutting) は最大幅の制限(maxing) より先に行なわれるので、表示欄での見栄えを良くするために、日付が 6 文字より少なくならないことを保証する詰め込み(padding) が必要になります。)

無視(ignore) が最初に行なわれます。それから切り落とし(cutting)、次に最大幅の制限(maxing)、そして最後の操作である詰め込み(padding) が行なわれます。

10.4.4 利用者定義の指定

すべての仕様に、利用者が定義した'`u`' で始まる述語を挿入することができます。書法仕様文字列の次の文字は、アルファベットでなければなりません。'`%u`' に続くアルファベットが'`X`' だったら、Gnus は関数`gnus-user-format-function-`'`X`' を呼びます。関数には単一の引数が与えられますが、その引数の意味は関数がどのバッファーから呼ばれているかによって変わります。関数は文字列を返さなければなりません。それは他の述語によって生成される情報とまったく同じように、バッファーに挿入されます。関数は意味の無い値と共に呼ばれる場合もあるので、それの対策をしておくべきです。

Gnus は利用者定義仕様を拡張した'`%u&foo;`' のような形式もサポートします。この場合は`gnus-user-format-function-`'`foo`' という関数を呼び出します。

新しい関数を定義しなくても、ほとんど同じことをチルダ修飾子(see Section 10.4.3 [Advanced Formatting], p. 273) を使って達成できるでしょう。例です:

'`%~(form (count-lines (point-min) (point)))@`'

ここで与えられた式は評価されて現在の行番号をもたらし、それから挿入されます。

10.4.5 書法仕様フォント

すべての書法仕様変数によって共有される、ハイライト(強調表示) のための仕様があります。述語‘%(’ と述語‘%)’ で囲まれたテキストには特別なmouse-face 属性が与えられ、そこにマウスのポインターを置いたときに(gnus-mouse-face によって) ハイライトされます。

述語‘%{’ と述語‘%}’ で囲まれたテキストには、普通のフェースであるgnus-face-0 (ディフォルトでbold) が与えられます。‘%1{’ を使うと、代わりにgnus-face-1 が与えられ、以下同様です。欲しいだけたくさんのフェースを作ってください。同じことがmouse-face 仕様にも言えます。‘hello’ がマウスを置いたときにgnus-mouse-face-3 でハイライトされるためには、‘%3(hello%)’ とすれば良いでしょう。

述語‘%《’ と述語‘%》’ で囲まれたテキストでは、特別なballoon-help 属性がgnus-balloon-face-0 に設定されます。‘%1《’ とするとgnus-balloon-face-1 が使われ、以下同様です。gnus-balloon-face-* 変数は、文字列か文字列を返す関数を指すシンボルのどちらかでなければなりません。この属性が設定されているテキストの上をマウスが通過すると、吹き出しが現れて文字列を表示します。これの詳しい情報はSection “Tooltips” in *The Emacs Manual* の説明を参照してください。(技術的な理由のために、ギィメ(guillemets) はこの節では‘《’ と‘》’ で近似されました。)

> 訳注: guillemets (仏語) はギュメとも表記されます。実際に Gnus で有効なのは次の二つです:
>
> ```
> (string (make-char 'latin-iso8859-1 43)) ;; 《
> (string (make-char 'latin-iso8859-1 59)) ;; 》
> ```
>
> 日本語の「」に当たるもので、口頭表現を表記したり、強調したい単語を囲む、何かからの引用部分を囲む、書物等のタイトルを記す等様々に使われます。

これはグループバッファーで使うことができる、もう一つの調理法です:

```
;; 三つのフェースを作ります。
(setq gnus-face-1 'bold)
(setq gnus-face-3 'italic)

;; 記事の数をボールドで緑のフェースにしたいので、
;; my-green-bold という新しいフェースを作ります。
(copy-face 'bold 'my-green-bold)
;; 色を設定します。
(set-face-foreground 'my-green-bold "ForestGreen")
(setq gnus-face-2 'my-green-bold)

;; 新しい特製の書法仕様を設定します。
(setq gnus-group-line-format
      "%M%S%3{%5y%}%2[:%] %(%1{%g%}%)\n")
```

あなたがこの案を使って、完全に読めなくて非常に下品な表示を作ることができることを確信しています。楽しんでください!

‘%(’ 指定(やその類のもの) は、モード行変数ではまったく意味をなさないことに注意してください。

10.4.6 ポイントの移動

Gnus は通常ほとんどのバッファーで、ポイントを各行のあらかじめ決められた場所に移動します。ディフォルトでは、ポイントは行の最初のコロンに移動します。この振るまいは、三つの違う方法でカスタマイズすることができます。

また、コロンを行のどの場所にでも移動することができます。

コロンの位置にポイントを移動させるための関数を定義し直すことができます。その関数は`gnus-goto-colon` と呼ばれています。

でも、行にコロンを含めたくないならば、これを扱うためのおそらく最も手ごろな方法は‘`%*`’ という述語を使うことです。あなたの行の書法仕様の定義に‘`%*`’ を入れておけば、Gnus はそこにポイントを置きます。

10.4.7 整列

通常は、空白文字を詰め込んだり端を切り落とすことによって、文字列をディスプレイに並べることができます。でも大きさが違う異なる文字列を連結させる場合は、単に文字列を出力してしまうのがより手ごろであることが多いはずで、しかしそうするとその後に続くテキストを並べるのに悩むことになります。

それを行なうために、Gnus は整列子(tabulator) の仕様‘`%=`’ を備えています。これには二つの形式*hard tabulators* および*soft tabulators* があります。

‘`%50=`’ は文字列が 50 桁までの場所を占めるように空白文字を詰め込みます。もし挿入するテキストの末端が 50 桁より後ろになる場合は何も挿入しません。これは穏やか(soft) な整列子です。

‘`%-50=`’ もまた、文字列が 50 桁までの場所を占めるように空白文字を詰め込みます。ですが、もし挿入するテキストの末端が 50 桁より後ろになる場合は、50 桁より後ろになる過剰なテキストは削除されます。これは厳密(hard) な整列子です。

10.5 ウィンドウの配置

いえ、X に関することはありませんから、おとなしくしてください。

もし`gnus-use-full-window` が`nil` でないと、Gnus はすべての他のウィンドウを消して、Emacs の画面全体を占有します。これはディフォルトで`t` です。

この変数を`nil` に設定してもそれなりに動作しますが、問題もあります。危険を覚悟の上で使ってください。

`gnus-buffer-configuration` はそれぞれの Gnus のバッファーがどのくらいの空間を与えられるべきかを現します。これはこの変数の抜粋です:

```
((group (vertical 1.0 (group 1.0 point)))
 (article (vertical 1.0 (summary 0.25 point)
                        (article 1.0))))
```

これは連想リストです。「キー」は何らかの動作を名付けるためのシンボルです。例えば、グループバッファーを表示するときは、ウィンドウを設定するための関数は`group` をキーとして使います。使用可能な名前の完全な一覧は下に挙げられています。

「値」(すなわち「分割」) は、それぞれのバッファーがどれくらいの空間を占めるべきかを指定します。`article` の分割を例に取ると:

```
(article (vertical 1.0 (summary 0.25 point)
                       (article 1.0)))
```

この「分割」は概略バッファーが画面の上の 25% を占めるべきで、それは記事バッファーの上に配置されると言っています。お気づきのように、100% + 25% は実際は 125% です(えぇ、皆さんの計算はこの様になったと思います。) しかし、特別な数値1.0 は、残りのバッファーが必要なものを取り去った後に、使用可能な残りの空間すべてを吸い取る、ということを合図するために使われます。1.0 の大きさを指定するバッファーは、一つの分割につき一つだけでなくてはなりません。

ポイント(カーソル) は省略可能な三つ目の要素、point を持つバッファーに置かれます。frame 分割では、frame-focus タグが含まれている枝葉の分割を持っている副分割の、最後のもののフレームがフォーカスを得ることになります(frame-focus タグは、それを含んでいる枝葉リストにおいて、point タグが無ければ三番目の、あれば四番目の要素になります。

次はもっと複雑な例です:

```
(article (vertical 1.0 (group 4)
                       (summary 0.25 point)
                       (article 1.0)))
```

もし大きさの指定が浮動小数点数の代わりに整数だったなら、それは割合ではなく、どのくらい多くの行をバッファーが占めるべきかを指定するために使われます。

もし「分割」がeval (評価) されるもののように見えるときは(正確に言うと---分割のcarが関数か原始関数(subr) であるときは)、この分割はeval されます。結果がnil でないなら、それは分割として用いられます。

まだ複雑ではないですって? それでは、大きさとしてこれを試してみてください:

```
(article (horizontal 1.0
             (vertical 0.5
                 (group 1.0))
             (vertical 1.0
                 (summary 0.25 point)
                 (article 1.0))))
```

おぉっと。二つのバッファーに謎の 100% タグが付いています。そしてhorizontal って何でしょう?

もし分割の一つの最初の要素がhorizontal であったなら、Gnus はウィンドウを水平に分割し、二つのウィンドウを横に並べます。これらのそれぞれの小片の中では、あなたのやりたいことをすべて普通の流儀で行なうことができます。horizontal の後の数値は、この小片に画面のどれくらいの割合が与えられるかを指定します。

それぞれの分割では、100% のタグを持つ要素が必ず 一つある必要があります。分割は決して正確ではないので、分割によって余ったすべての行を、このバッファーが分捕ります。

もう少し形式的に、有効な分割がどのようになるかの定義を挙げておきましょう:

```
split      = frame | horizontal | vertical | buffer | form
frame      = "(frame " size *split ")"
horizontal = "(horizontal " size *split ")"
vertical   = "(vertical " size *split ")"
buffer     = "(" buf-name " " size *[ "point" ] *[ "frame-focus"] ")"
size       = number | frame-params
buf-name   = group | article | summary ...
```

制限として、frame は最も上位階層の分割としてしか現れることができないというものがあります。form は有効な分割を返す Emacs Lisp の式(form) でなければなりません。それぞれの分割は完全に再帰的で、任意の数のvertical とhorizontal 分割を含むことができます。

正しい大きさを見つけることは、少し複雑になります。どのウィンドウもgnus-window-min-height (ディフォルトは 1) の文字の高さよりも小さくてはならないし、少なくともgnus-window-min-width (ディフォルトは 1) の文字幅でなくてはなりません。Gnus は分割を適用する前にこれを強制しようと試みます。もし標準の Emacs のウィンドウの幅/高さ制限を使いたいなら、この二つの変数をnil にするだけで良いです。

Emacs の用語になじんでいないのなら、horizontal とvertical の分割は、期待したものと反対の動作をするでしょう。horizontal 分割の中のウィンドウは横に並んで表示され、vertical 分割の中のウィンドウは上下に表示されます。

ウィンドウの配置に関して実験をしてみたいのであれば、良い方法は分割を引数にして直接gnus-configure-frame を呼ぶことです。これはバッファーを分割するときにすべての実際の仕事をする関数です。下のものは五つのウィンドウを作るかなりばかげた設定です。二つをグループバッファーに、三つを記事バッファーのために充てます。(だから、ばかげていると言ったでしょ。) もし下の文を評価 すると、普通の Gnus の経路を使わないで、すぐにそれがどのように見えるかの直観を得ることができます。満足するまでそれで遊んで、それからgnus-add-configuration を使って新しい作品をバッファー配置リストに加えてください。

```
(gnus-configure-frame
 '(horizontal 1.0
    (vertical 10
      (group 1.0)
      (article 0.3 point))
    (vertical 1.0
      (article 1.0)
      (horizontal 4
        (group 1.0)
        (article 10)))))
```

複数のフレームも欲しいかもしれません。問題ありません—frame 分割を使うだけです:

```
(gnus-configure-frame
 '(frame 1.0
         (vertical 1.0
                   (summary 0.25 point frame-focus)
                   (article 1.0))
         (vertical ((height . 5) (width . 15)
                    (user-position . t)
                    (left . -1) (top . 1))
                   (picon 1.0))))
```

この分割の結果は、最初の(もしくは「主たる」) フレームに見慣れた概略/記事ウィンドウを配置し、小さな追加のフレームが picon を表示するために作られます。ご覧の通り、普通の最上位階層の1.0 の定の代わりに、それぞれの追加の分割が大きさの指定として、フレームパラメーターの連想リストを持たなければなりません(see Section “Frame Parameters”

in *The GNU Emacs Lisp Reference Manual*)。gnus-buffer-configuration で使うことができるすべてのキーの一覧は、そのディフォルト値で見つけることができます。

キーmessage はgnus-group-mail およびgnus-summary-mail-other-window の両方で使われることに注意してください。もし二つを区別するほうが望ましいなら、このような物を使うことができます:

```
(message (horizontal 1.0
                     (vertical 1.0 (message 1.0 point))
                     (vertical 0.24
                               (if (buffer-live-p gnus-summary-buffer)
                                   '(summary 0.5))
                               (group 1.0))))
```

良くある複数のフレーム分割の要望は、メールとニュースの作成には別のフレームを使い、元のフレームはそのままに残すというものです。これの達成には、以下のようなものでできます。

```
(message
  (frame 1.0
         (if (not (buffer-live-p gnus-summary-buffer))
             (car (cdr (assoc 'group gnus-buffer-configuration)))
           (car (cdr (assoc 'summary gnus-buffer-configuration))))
         (vertical ((user-position . t) (top . 1) (left . 1)
                    (name . "Message"))
                   (message 1.0 point))))
```

訳注: これを高度に発展させたものがhttp://www.jpl.org/elips/message-multiple-frames.el.gz として入手できます。使い方はファイルの冒頭に書かれています。

変数gnus-buffer-configuration はとても長く複雑なので、単一の設定の変更を簡単にするための関数があります: gnus-add-configuration です。例えばarticle の設定を変えたいのなら、次のようにできます:

```
(gnus-add-configuration
 '(article (vertical 1.0
                     (group 4)
                     (summary .25 point)
                     (article 1.0))))
```

普通はこれらのgnus-add-configuration の呼び出しを~/.gnus.el ファイルに入れるか、何らかの起動時のフックに入れるでしょう---それらは Gnus が読み込まれた後で実行されなければなりません。

もし分割の設定で指定されたすべてのウィンドウがすでに見えているのであれば、Gnus はウィンドウの配置を変更しません。常に「正しい」ウィンドウ設定を強制したいのであれば、gnus-always-force-window-configuration をnil でない値に設定してください。

木表示(see Section 4.25 [Tree Display], p. 112) を使っていて、木ウィンドウが垂直方向に次の別のウィンドウで表示されるなら、ウィンドウの大きさが変更されることを避けるためにgnus-tree-minimize-window をいじるのが良いでしょう。

最後に、gnus-use-atomic-windows をt に設定することによって Gnus のウィンドウ配置を“atomic” (see Section “Atomic Windows” in *The GNU Emacs Lisp Reference Manual*)

にすることができます。これによってポップアップされるバッファー(例えばヘルプや補完バッファーなど) が Gnus ウィンドウ配置全体の下または横に表示されるので、例えば概略バッファーと記事バッファーの間で押しつぶされることはありません。

10.5.1 ウィンドウ配置の名称

以下は現在知られているほとんどのウィンドウ配置とそれらの用途です:

`group`
: グループバッファー。

`summary`
: あるグループに入ってその概略(だけ) を表示。

`article`
: 記事の選択。

`server`
: サーバーバッファー。

`browse`
: サーバーバッファーから閲覧するグループ。

`message`
: (新しい) メッセージの作成。

`only-article`
: 記事バッファーだけの表示。

`edit-article`
: 記事の編集。

`edit-form`
: グループパラメーターなどの編集。

`edit-score`
: サーバーの定義の編集。

`post`
: ニュース記事の作成。

`reply`
: 記事に元記事の引用無しで返信またはフォローアップ。

`forward`
: 記事の転送。

`reply-yank`
: 記事に元記事の引用付きで返信またはフォローアップ。

`mail-bounce`
: メールの弾き返し。

`pipe`
: 記事を外部プロセスに送る。

`bug`
: バグリポートを送る。

`score-trace`
: スコア規則の表示。

`score-words`
: スコアに使う単語の表示。

`split-trace`
: 分割されるメールの行方の表示。

`compose-bounce`
: 弾き返されたメールの再送信。

`mml-preview`
: MIME パートの送信する前の下見。

10.5.2 ウィンドウ配置の例

- 左側を狭めてグループバッファーに。右側を分割して概略バッファー(上 1/6) と記事バッファー(下) に。

```
(gnus-add-configuration
 '(article
   (horizontal 1.0
               (vertical 25 (group 1.0))
               (vertical 1.0
                         (summary 0.16 point)
                         (article 1.0)))))

(gnus-add-configuration
 '(summary
   (horizontal 1.0
               (vertical 25 (group 1.0))
               (vertical 1.0 (summary 1.0 point)))))
```

10.6 Tab によるインターフェース

Gnus は専用のタブ上で tab bar を使って色々なバッファーの表示を選択することをサポートしています。

グループバッファーを‘`Gnus`’と名付けられた新規の tab で開くには、これを使ってください:

```
(push '("\\`\\*Group\\*\\'" .
        (display-buffer-in-tab
         (tab-name . "Gnus")))
      display-buffer-alist)
```

すべての概略バッファーを明確に名付けられた別々の tab で読むにはこうです:

```
(push '("\\`\\*Summary .*\\*\\'" .
        (display-buffer-in-tab
```

```
        (tab-name . (lambda (buffer _alist)
                      (setq buffer (buffer-name buffer))
                      (when (string-match "\\`\\*Summary \\(.*\\)\\*\\'" buffer)
                        (format "Group %s" (match-string 1 buffer)))))))
    display-buffer-alist)
```

10.7 フェースとフォント

かつてフォントとフェースをいじくるのは非常に難しかったのですが、今日では非常に簡単です。単に*M-x customize-face* とやって、変えたいフェースを選び出して、標準のカスタマイズインターフェースを使って変更することができます。

10.8 モード行

gnus-updated-mode-lines はどのバッファーがそれらのモード行を常に最新のものにしておくかを指定します。それはシンボルのリストです。使うことのできるシンボルはgroup, article, summary, server, browse, tree などです。もし対応するシンボルが存在すると、Gnus は該当する情報でモード行を更新します。この変数がnil ならば、画面の再描画はもっと速いでしょう。

ディフォルトでは、Gnus は概略バッファーと記事バッファーのモード行に現在の記事の情報を表示します。Gnus が表示したい情報(例えば記事の表題) はしばしばモード行よりも長いことがあるので、どこかで切り落とされなければなりません。変数gnus-mode-non-string-length はその行の他の要素(すなわち情報でない部分) がどのくらいの長さであるかを指定します。もしモード行に追加の要素を入れたなら、この変数を修正する必要があります:

```
(add-hook 'display-time-hook
          (lambda () (setq gnus-mode-non-string-length
                           (+ 21
                              (if line-number-mode 5 0)
                              (if column-number-mode 4 0)
                              (length display-time-string)))))
```

もしこの変数がnil であるなら(これがディフォルトですが)、モード行は切り落とされず、詰め込みもされません。ディフォルトでは、バッファーの完全なパーセント表示さえもモード行から追いやられる可能性もあるので、おそらく望ましい設定ではないことに注意してください。利用者が自分の設定に合うようにこの変数を適切に設定しなければなりません。

10.9 ハイライトとメニュー

変数gnus-visual は Gnus を素敵にするたいていの方面の操作をします。nil であると、Gnus はメニューを作ったり、素敵な色やフォントを使ったりしようとしません。これはさらにgnus-vis.el ファイルを読み込むことも禁止します。

この変数は有効にされている視覚的なプロパティーのリストであることができます。以下の要素は有効で、ディフォルトですべて含まれています:

group-highlight
: グループバッファーでハイライト(強調表示) をします。

`summary-highlight`
: 概略バッファーでハイライトをします。

`article-highlight`
: 記事バッファーでハイライトをします。

`highlight`
: すべてのバッファーでハイライトをするようにします。

`group-menu`
: グループバッファーでメニューを作成します。

`summary-menu`
: 概略バッファーでメニューを作成します。

`article-menu`
: 記事バッファーでメニューを作成します。

`browse-menu`
: ブラウズバッファーでメニューを作成します。

`server-menu`
: サーバーバッファーでメニューを作成します。

`score-menu`
: スコアバッファーでメニューを作成します。

`menu`
: すべてのバッファーでメニューを作成します。

ですから、記事バッファーだけをハイライトしたくて、すべてのバッファーでメニューを作りたい場合は、このようにすることができます:

```
(setq gnus-visual '(article-highlight menu))
```

もしハイライトだけで、メニューの類は欲しくないときは、次のようにできます:

```
(setq gnus-visual '(highlight))
```

`gnus-visual` が`t` であると、ハイライトとメニューはすべての Gnus のバッファーで使用されます。

他のすべてのバッファーの外見に影響する総合的な変数は:

`gnus-mouse-face`
: これは Gnus でマウスのハイライトに使われるフェース(すなわちフォント) です。`gnus-visual` が`nil` であると、マウスハイライトはなされません。

まったく違ったメニューを作成するために、関連するフックがあります:

`gnus-article-menu-hook`
: 記事モード(article mode) のメニューを作成した後に呼ばれるフックです。

`gnus-group-menu-hook`
: グループモード(group mode) のメニューを作成した後に呼ばれるフックです。

`gnus-summary-menu-hook`
: 概略モード(summary mode) のメニューを作成した後に呼ばれるフックです。

`gnus-server-menu-hook`
: サーバーモード(server mode) のメニューを作成した後に呼ばれるフックです。

`gnus-browse-menu-hook`
: 概観モード(browse mode) のメニューを作成した後に呼ばれるフックです。

`gnus-score-menu-hook`
: スコアモード(score mode) のメニューを作成した後に呼ばれるフックです。

10.10 デーモン

Gnus、それは(言い伝えによれば) かつて書かれたいかなるプログラムよりも大きく、あなたがやって欲しいと思うさまざまな奇妙なことを、あなたのいないところで行なってくれるものです。例えば、あなたは時たま新着メールをチェックしてもらいたいかもしれません。あるいは Emacs をしばらく放っておいたときすべてのサーバーの接続を切断してもらいたくなるかもしれません。他にも何かそういったことです。

Gnus はさまざまな「ハンドラー」(処理を行なわせるためのもの) を定義することによってそのようなことを可能にします。各ハンドラーは三つの要素から成ります:「関数」、「時間」、「アイドル」(何もしていない状態を示すもの) パラメーターです。

これは Emacs のアイドル状態が三十分続いたときに接続を切断するハンドラーの例です:

```
(gnus-demon-close-connections nil 30)
```

これは Emacs がアイドルのとき、一時間毎にPGP ヘッダーを走査するハンドラーです:

```
(gnus-demon-scan-pgp 60 t)
```

この「時間」パラメーターと「アイドル」パラメーターは、奇妙かつ素晴らしいやり方で一緒に動作します。基本的に「アイドル」が`nil` だったら、関数は「時間」分毎に呼び出されます。

「アイドル」が`t` だったら、関数は Emacs がアイドルだったときに限って「時間」分後に呼び出されます。したがって Emacs がアイドルにならなければ、関数は呼び出されません。いったん Emacs がアイドル状態になると、この関数は「時間」分毎に呼び出されます。

「アイドル」が数値で「時間」も数値だった場合、Emacs のアイドル状態が「アイドル」分続いた場合に限って、「時間」分毎に関数が呼び出されます。

「アイドル」が数値で「時間」が`nil` だった場合、関数は Emacs のアイドル状態が「アイドル」分続く度に一度呼び出されます。

そして「時間」が文字列だった場合(それは‘`07:31`’ のような形式でなければなりません)、関数は毎日その時刻の頃になると一度呼び出されます。もちろん「アイドル」パラメーターによって動作が変わります。

(ここで「分」と言ったとき、それは実際には`gnus-demon-timestep` 秒のことです。これはディフォルトでは 60 です。もしこの変数を変更すると、すべてのハンドラーの計時に影響を与えます。)

というわけで、ハンドラーを追加したければ、`~/.gnus.el` ファイルに以下のようなものを書き込めば良いでしょう:

```
(gnus-demon-add-handler 'gnus-demon-close-connections 30 t)
```

このための既製の関数がいくつか作成されています: `gnus-demon-add-disconnection`, `gnus-demon-add-nntp-close-connection`, `gnus-demon-add-scan-timestamps`, `gnus-demon-add-rescan`, および`gnus-demon-add-scanmail` です。これらの機能を必要とするならば、単にこれらの関数を`~/.gnus.el` に入れてください。

gnus-demon-handlers に直接ハンドラーを追加した場合には、それを効かせるためにgnus-demon-init を実行してください。すべてのデーモンを取り消すには、gnus-demon-cancel 関数を使うことができます。

デーモンの追加をやりすぎるのはかなりマズいことです。すべてのサーバーからすべてのニュースとメールを二秒毎に調べまわす関数を付け加えたりすることは、どんな立派なシステムからも確実に追い出される方法です。お行儀良くしましょう。

10.11 通知

Gnus が提供するgnus-notifications を使うと、gnus-demon ハンドラーと組み合わせて、新しいニュースやメールが届いたときにデスクトップ通知によって知らせることができます。以下はあなたの~/.gnus.el ファイルに入れることができる基本的な設定です

```
(require 'gnus-notifications)
(add-hook 'gnus-after-getting-new-news-hook #'gnus-notifications)
(gnus-demon-add-handler #'gnus-demon-scan-news 15 3) ; 15 分毎
```

上記のものはgnus-after-getting-new-news-hook フック(ニュースやメールをgnus-demon-scan-news の処理によって取得した後で起動するフック) が実行されるときに、新着の通知を発行します。デーモンハンドラーは(Emacs が 3 分間アイドル状態になった後) 15 分毎にニュースをスキャンするように設定されています。

> 重要: gnus-notifications は、レベルが*gnus-notifications-minimum-level* (ディフォルトは 1) と等しいか低いグループの通知のみを発行するので、通知するグループのレベルを1 に設定するか、*gnus-notifications-minimum-level* の値を必要に応じて調整してください。

10.12 やり直し

実行したことのやり直しができると、とても便利です。Emacs の普通のバッファーでは十分に簡単です---単にundo ボタンを押すだけです。しかし Gnus のバッファーでは、それは簡単ではありません。

Gnus がバッファー内に表示しているものは、Gnus にとってはまったく何の価値もありません---これはみんな、利用者に奇麗に見えるようにデザインされているただのデータなのです。*C-k* でグループバッファーからグループを消去すると、その行は消え去りますが、それは実際の動作---当のグループを Gnus の内部構造体から削除すること、の単なる副作用でしかありません。これらのやり直しは、通常の Emacs のundo 関数では行なうことができません。

Gnus は利用者がすることを憶えておいて、利用者がすることの逆を行なうことによって、これを多少は救済しようとします。利用者がundo キーを押すと、一段階または数段階前までの操作を元に戻すコードを実行します。しかし、すべての操作が簡単に逆戻りできるわけではないので、現在 Gnus は、やり直し可能なキーの機能を僅かしか提供していません。これらはグループの削除、グループの貼り付け、およびグループの既読記事のリストの変更です。実際それだけです。将来はもっと機能が追加されるかもしれませんが、追加されるそれぞれの機能は保存するべきデータを増やすので、決して Gnus は完全にやり直し可能にはならないでしょう。

やり直し機能はgnus-undo-mode マイナーモードによって提供されます。これはgnus-use-undo がnil 以外であれば使用され、これがディフォルトです。*C-M-_* キーがgnus-undo 命令を実行します。これは通常の Emacs のundo 命令にいくぶん似ているはずです。

10.13 述語指示子

いくつかの Gnus の変数は「述語指示子」(predicate specifiers) です。これは、その多くをすべて記述する必要なしに、述語の仕様に融通を効かせることができる特別な形式です。

これらの指示子は関数、シンボルおよびリストからなるリストです。

例です:

```
(or gnus-article-unseen-p
    gnus-article-unread-p)
```

利用できるシンボルは`or`、`and` および`not` です。関数はすべて一つのパラメーターを受け取ります。

呼ぶことができる関数を作るために、Gnus はこれらの指示子について内部的に`gnus-make-predicate` を使います。この関数へのこの入力パラメーターは、述語指示子の中のすべての関数に渡されます。

10.14 司会役

もしあなたが司会者(モデレーター) ならば、`gnus-mdrtn.el` パッケージを使うことができます。これは標準の Gnus パッケージには含まれていません。‘`larsi@gnus.org`’ に、どのグループの司会を行なうのかを述べたメールを書いてください。そうすればコピーを手に入れることができます。

司会者用パッケージは概略バッファーのマイナーモードとして実装されています。

```
(add-hook 'gnus-summary-mode-hook 'gnus-moderate)
```

をあなたの`~/.gnus.el` ファイルに入れてください。

あなたが‘`rec.zoofle`’ の司会者だとすると、これは以下のように動作するようになっています:

1. 受信したメールを‘`Newsgroups:.*rec.zoofle`’ に合致させることによって分割します。これは投稿されようとしているすべての記事を、あるメールグループ---例えば‘`nnml:rec.zoofle`’ に入れます。
2. あなたは時折このグループに入り、*e* (edit-and-post) あるいは*s* (just send unedited) 命令を使って記事を投稿します。
3. ‘`rec.zoofle`’ ニュースグループを読んでいる途中で、もしあなたが承認していない記事をたまたま見つけたとしたら、*c* 命令で取り消しできます。

二つのグループで司会者モードを使うとすれば、こうなります:

```
(setq gnus-moderated-list
      "^nnml:rec.zoofle$\\|^rec.zoofle$")
```

10.15 グループを取得する

時々「Gnus が起動しているかどうかを気にしないでこのグループを読みたい。」ということができれば便利なことがあります。これは、利用者よりもプログラムのコードを書く人に便利な機能ですが、どちらにしろ`gnus-fetch-group` コマンドはこの機能を提供します。それはグループの名前を引数としてとります。

10.16 画像の拡張

Emacs は画像などを表示できるので、Gnus はそれを利用しています。

10.16.1 X-Face

`X-Face` ヘッダーは、メッセージの著者を表わすことになっている 48×48 画素の白黒(深さ 1 bit の) の絵を描きます。これは進化し続けるあまたのメールとニュースリーダーによってサポートされるでしょう。

`X-Face` ヘッダーを見るには‘`compface`’ をサポートしている Emacs か、変換または表示のための適切なプログラムをインストールしてあることが必要です。あなたの Emacs が自前で画像の表示をサポートしているならば、ディフォルトで`From` ヘッダーの前に顔が表示されます。Emacs が自前で`X-Face` をサポートしていない場合、Gnus は`pbmplus` パッケージとその仲間の外部プログラム(下記参照) を使って`X-Face` ヘッダーを変換しようとします。画像をサポートしていない Emacs では、ディフォルトでは表示のための処理を`display` というプログラムに委ねます。

GNU/Linux システムの場合、ImageMagick パッケージに含まれている`display` プログラムを使います。外部プログラムとしては`netpbm`、`libgr-progs` および`compface` のような名前のものを探します。Windows では`https://gnuwin32.sourceforge.net` にある`netpbm` および`compface` パッケージを使っても良いです。`PATH` 環境変数に`bin` ディレクトリーを追加する必要があります。

変数`gnus-article-x-face-command` で、`X-Face` ヘッダーを表示するために何のプログラムを使うかを制御します。この変数が文字列ならば、この文字列がサブシェルで実行されます。関数ならば、この関数が顔を引数として呼ばれます。もし`gnus-article-x-face-too-ugly` (これは正規表現です) が`From` 欄に合致すれば、顔は表示されません。

(注: 変数/関数名には`xface` ではなく`x-face` が使われます。)

フェースと変数:

`gnus-x-face`
: X-Face を表示するためのフェース。このフェースの色が表示される X-Face の前景色と背景色として使われます。ディフォルトの色は黒と白です。

`gnus-face-properties-alist`
: Face (see Section 10.16.2 [Face], p. 288) と X-Face 画像に適用される、画像の形式とプロパティーの連想リストです。ディフォルト値は`((pbm . (:face gnus-x-face)) (png . nil))` です。例を挙げましょう:

```
;; From ヘッダーにおける Face と X-Face の高さを指定します。
(setq gnus-face-properties-alist
      '((pbm . (:face gnus-x-face :ascent 80))
        (png . (:ascent 80))))

;; Face と X-Face を凹んだボタンのように表示します。
(setq gnus-face-properties-alist
      '((pbm . (:face gnus-x-face :relief -2))
        (png . (:relief -2))))
```

: いろいろな画像の形式で利用可能なプロパティーについてはSee Section “Image Descriptors” in *The Emacs Lisp Reference Manual*, を参照してください。今のところ`pbm` が X-Face 画像に使われ、`png` が Face 画像に使われます。

投稿様式(posting style) を使うのであれば、gnus-posting-styles にx-face-file の項を加えれば良いでしょう(see Section 6.6 [Posting Styles], p. 140)。さもなければ、外に出すメッセージに X-Face ヘッダーを簡単に挿入できるようにするために Gnus が提供する、いくつかの便利な関数と変数を利用することができます。これらの機能のためには、前述の ImageMagick、netpbm または他の画像を変換するパッケージ(何が必要かは、下記の変数群の値によります) も必要です。

gnus-random-x-face はgnus-x-face-directory にあるすべての‘pbm’ ファイルをくまなく探してランダムに一つを選び取り、シェルコマンドgnus-convert-pbm-to-x-face-command を使ってそれを X-Face の形式に変換します。‘pbm’ ファイルは 48×48 画素の大きさでなければなりません。それは X-Face ヘッダーのデータを文字列で返します。

gnus-insert-random-x-face-header はgnus-random-x-face を呼んで、ランダムに生成されたデータによる X-Face ヘッダーを挿入します。

gnus-x-face-from-file はパラメーターとして GIF ファイルを受け取り、シェルコマンドgnus-convert-image-to-x-face-command を使ってそのファイルを X-Face の形式に変換します。

一番目の関数の一般的な使い方を示します。以下のようなものを~/.gnus.el ファイルに書き込んでください:

```
(setq message-required-news-headers
      (nconc message-required-news-headers
             (list '(X-Face . gnus-random-x-face))))
```

最後の関数を使うのは、このようになるでしょう:

```
(setq message-required-news-headers
      (nconc message-required-news-headers
             (list '(X-Face . (lambda ()
                                (gnus-x-face-from-file
                                 "~/My-face.gif"))))))
```

10.16.2 Face

Face ヘッダーは、本質的にX-Face をよりファンキーに変形したものです。それらは、メッセージを書いた人を象徴することになっている 48×48 画素のカラー画像を描きます。

Face ヘッダーの内容は base64 でエンコードされた PNG の画像でなければなりません。正確な仕様について、https://quimby.gnus.org/circus/face/ を参照してください。

変数gnus-face-properties-alist は表示される Face 画像の外観に影響します。See Section 10.16.1 [X-Face], p. 287.

Face ヘッダーを見るには Emacs が PNG 画像を表示できる必要があります。

Gnus は外に出すメッセージに Face ヘッダーを簡単に挿入できるようにするための、便利な関数と変数を少しばかり提供します。

gnus-convert-png-to-face は 726-byte 以下の 48×48 の PNG の画像を受け取って、それを Face に変換します。

gnus-face-from-file は JPEG ファイルの名前をパラメーターとして受け取り、シェルコマンドgnus-convert-image-to-face-command を使ってそのファイルを Face フォーマットに変換します。

この関数の代表的な使い方を挙げておきましょう。以下のようなものを~/.gnus.el ファイルに入れてください:

```
(setq message-required-news-headers
      (nconc message-required-news-headers
             (list '(Face . (lambda ()
                              (gnus-face-from-file "~/face.jpg"))))))
```

10.16.3 スマイリー

スマイリー*smiley* は Gnus とは別のパッケージですが、スマイリーを使っているパッケージは現在 Gnus だけなので、ここで説明します。

ひとことで言えば---Gnus でスマイリーを使うには、以下を~/.gnus.el ファイルに書き込んでください。

```
(setq gnus-treat-display-smileys t)
```

スマイリーは、文字の顔マーク—':-)', '8-)', ':-(' などといったもの---を絵に割り当てて、文字の顔マークの代わりにその絵を表示します。この変換は文字に合致する正規表現と、それに割り当てられたファイル名のリストで制御されます。

使われる連想リストは、変数smiley-regexp-alist で設定します。各要素の最初の項目は合致する正規表現で、二番目の要素は絵で置き換えられる正規表現のグループ番号、そして三番目の要素は表示されるファイルの名前です。

以下の変数でスマイリーの見栄えをカスタマイズします:

smiley-style
: スマイリーの形式を指定します。あらかじめ定義されている形式はemoji (絵文字を使う)、low-color (13×14 画素で 3 色の小さい画像)、medium (16×16 画素でもっとカラフルな画像) およびgrayscale (14×14 画素のグレースケール画像) です。ディフォルトはディフォルト face の高さに依存します。

smiley-data-directory
: スマイリーが顔ファイルを探す場所です。もうこの変数はいじらないで、代わりにsmiley-style をカスタマイズしてください。

gnus-smiley-file-types
: スマイリーのファイル名として試してみる拡張子のリストです。

10.16.4 Picons

それで...、あなたはこのニュースリーダーをさらにもっと遅くしたいってわけですね! これはそうするのにぴったりな方法です。さらにこれは、あなたがニュースを読んでいるんだということを、あなたの肩越しに見つめている人に印象づけるための素晴らしい方法でもあります。

Picon とはなんでしょう? Picons ウェブサイトから直接引用しましょう。

> *Picon* とは「個人アイコン(personal icons)」の略です。これは、ある電子メールアドレスのための適切な画像を見つけることができるように、無理矢理小さくしてデータベースにまとめられた画像たちで、ネット上の利用者やドメインを表現するために使われます。利用者とドメイン以外に、Usenet ニュースグループや天気予報のための picon データベースがあります。picon は白黒のXBM 形式、またはカラーのXPM 形式およびGIF 形式のいずれでも構いません。

Picon データベースの入手とインストールの手順については、ウェブブラウザーで`https://cs.indiana.edu/ftp/faces/picons/` を訪ねてみてください。

もし Debian GNU/Linux を使っているのなら、‘`apt-get install picon-.*`’ と言えば、Gnus が見つけることができる picon がインストールされます。

Picon の表示ができるようにするためには、picon データベースがあるディレクトリーが、ただ単に`gnus-picon-databases` に設定されているようにしてください。

変数`gnus-picon-style` は picon をどのように表示するかを制御します。`inline` だったらテキスト形式の表現が置き換えられます。`right` だったら、テキスト形式の表現の右側に picon が加えられます。

変数`gnus-picon-properties` の値は picon に適用される属性のリストです。

ものごとの所在を管理するために、以下の変数を設けています。

`gnus-picon-databases`
: Picon データベースの場所です。これは`news`, `domains`, `users` (などなど) のサブディレクトリーが含まれているディレクトリーのリストです。`("/usr/lib/picon" "/usr/local/faces")` がディフォルトです。

`gnus-picon-news-directories`
: `gnus-picon-databases` からニュースグループ用のフェースを探すためのサブディレクトリーのリストです。ディフォルトは`("news")` です。

`gnus-picon-user-directories`
: `gnus-picon-databases` から利用者のフェースを探すためのサブディレクトリーのリストです。`("local" "users" "usenix" "misc")` がディフォルトです。

`gnus-picon-domain-directories`
: `gnus-picon-databases` からドメイン名のフェースを探すためのサブディレクトリーのリストです。ディフォルトは`("domains")` です。このリストに‘`"unknown"`’ を追加しておきたくなる人もいるでしょう。

`gnus-picon-file-types`
: Picon のファイル名として試してみる順に並べられた拡張子のリストです。ディフォルトは`("xpm" "gif" "xbm")` から Emacs に組み込まれていないものを除外したものです。

`gnus-picon-inhibit-top-level-domains`
: もし`nil` ではない(それがディフォルト) 値だったら、‘`.net`’ や‘`.de`’ のようなものについて picons を表示しません。普通それらはあまり面白いものではありません。

10.16.5 Gravatars

Gravatar はそれぞれの電子メールアドレスに対応して登録された画像です。

あなたのものを`https://en.gravatar.com/` にオンラインで置くことができます。

以下の変数は、それらがどのように表示されるかの制御を提供します。

`gnus-gravatar-size`
: Gravatars の画素のサイズ。Gravatars は常に正方形なので、サイズの指定には単一の数値で十分です。もし`nil` ならば`gravatar-size` の値がディフォルトとして使われます。

`gnus-gravatar-properties`
: Gravatar 画像に適用する属性のリストです(see Section “Image Descriptors” in *The Emacs Lisp Reference Manual*)。

`gnus-gravatar-too-ugly`
: 表示されるべきではないアバターの持ち主たちのメールアドレスまたは名前に合致する正規表現です。`nil` であるとすべてのアバターを表示します。ディフォルトは`gnus-article-x-face-too-ugly` の値です(see Section 10.16.1 [X-Face], p. 287)。

Gravatar を From フィールドで見たいなら、こうしてください:

```
(setq gnus-treat-from-gravatar 'head)
```

もしそれらを Cc と To フィールドで見たいなら、こうです:

```
(setq gnus-treat-mail-gravatar 'head)
```

10.16.5.1 ツール・バー

`gnus-group-tool-bar`
: グループバッファー用ツール・バーの指定です。リスト、またはその値がリストである変数シンボルです。

`gnus-summary-tool-bar`
: 概略バッファー用のツール・バーの指定です。リスト、またはその値がリストである変数シンボルです。

10.17 ファジーな一致

Gnus はスコア付け、スレッドの形成、およびスレッドの比較などを行なうときに、`Subject` 行のファジーな合致*fuzzy matching* を提供します。

正規表現による合致とは違って、ファジーな合致はとってもファジーです。あまりにもファジーすぎて、何がファジーであるかという定義さえ無いし、実装も何度も変更されています。

基本的に、これは比較の前に行から邪魔物を取り除こうとします。‘`Re:` ’、挿入句の印、および空白文字等々が文字列から除去され、その結果を比較します。これはほとんどの場合妥当な結果をもたらします---たとえニュースリーダーの仮面をかぶった文字列切り刻み機で生成された文字列が差し出されても、です。

10.18 spam メールの裏をかく

ここ最近の USENET では、宣伝のハゲタカどもが彼らの詐欺や製品を押し付けるための電子メールアドレスを探そうとして、気違いのようにニュース上をうろついて grep しまくっています。これに対する反動として、多くの人々が無意味なアドレスを`From` 行に入れはじめるようになってしまいました。私はこれは逆効果を招くと思います---あなたが書いたことに対する返信として人々が正当なメールを送ることを面倒にさせるだけでなく、誰が書いたものなのかを分かりづらくします。こんな書き換えは、結局は spam それ自体よりも大きな脅威となるかもしれません。

私にとっての spam メールの最大の問題は、嘘の口実で入ってくるからです。私が*g* を押すと、Gnus は十通の新着メールがありますと陽気に私に教えてくれます。私は「おおっ、わーい! 僕って幸せ!」と言ってメールグループを選択します。しかしそこには、二つの

ネズミ講と、七つの広告(「最新! 奇跡の育毛トニック、ふさふさでつやつやの髪をあなたのつま先(※)に!」) と、悔い改め神を信じよ、という一つのメールがあるだけなのです。

これは迷惑千万です。あなたがそれに関してできることがあります。

訳注※: ホビット族用の育毛トニック。たぶん。

10.18.1 Spam の問題

Spam は種々さまざまな出どころからやって来ます。有用なメッセージを捨てずにすべての spam を単に始末することは不可能です。

Spam の除去(filtering) への最も単純な取り組みは、メールサーバーで、あるいは入ってきたメールを分類するときに濾過すること(filtering) です。毎日'`random-address@example.org`' から 200 通の spam メッセージを受け取るのならば、'`example.org`' を阻止すれば良いでしょう。「バイアグラ」に関するメッセージを 200 通受け取るのならば、「バイアグラ」を含むすべてのメッセージを捨ててしまえば良いでしょう。例えばブルガリアからたくさんの spam がやって来るのならば、ブルガリアの IP から来るすべてのメールを濾過すれば良いでしょう。

これは、残念ながら正当な電子メールを捨てるためのすぐれた方法です。あなたに連絡しようとする国(ブルガリア、ノルウェー、ナイジェリア、中国、等) 全体、または大陸(アジア、アフリカ、ヨーロッパ、等) さえも封じ込めてしまう危険は明らかなので、あなたに選択権があるのならば、そんなことはしないでください。

もう一つの例として、とても示唆に富んで有益な RISKS ダイジェストは、それが spam メッセージと共通の語を含んで いるために、熱心すぎるメール濾過器によって阻止されてしまいます。それでもなお孤立した環境では、注意深く使うことによって直接の濾過は有益になり得ます。

もう一つの電子メール濾過への取り組みは分散型 spam 処理で、DCC (訳注: Distributed Checksum Clearinghouse—`http://www.rhyolite.com/anti-spam/dcc/`) がそのようなシステムを導入しています。本質的には、世界中のN 個のシステムが、ガーナ、エストニアあるいはカリフォルニアにあるマシンX が spam 電子メールを送出していることを認めたら、それらN 個のシステムはX またはX からやって来た spam メールをデータベースに記入します。Spam 検出の基準は一様ではありません。それは送られたメッセージの数やメッセージの内容などであるかもしれません。メッセージが spam かどうかを分散処理システムの利用者が知りたい場合、彼はそれらのN 個のシステムのうちの一つを調べます。

分散型 spam 処理は一度にたくさんのメッセージを送る spammers と非常によく戦ってくれますが、それには利用者がかなり複雑なチェックを設定することが必要です。商用とフリーな分散型 spam 処理システムがあります。分散型 spam 処理は、それ自体の危険もはらんでいます。例えば、正当な送信者が spam を送ったかどで非難され、彼らのウェブサイトやメーリングリストがその事件のために暫くの間閉鎖されてしまう、とか。

Spam の濾過への統計的な取り組みもまた普及しています。それは過去の spam メッセージの統計的な分析に基づいています。通常その分析は、おそらく単語の対か三つの単語の組合せの合成による、単語の出現頻度の単純な計数です。Spam の統計分析はほとんどの場合にとてもよく働くのですが、時として正当な電子メールを spam として分類してしまうことがあります。分析には時間がかかります。すべてのメッセージを分析しなければなりません。そして利用者は spam を分析するためのデータベースを蓄えなければなりません。サーバーでの統計分析は人気を得ています。これには、利用者は単にメールを読めば良いという長所と、しかしサーバーにそれが過ってメールを分類したことを伝えるのが困難だという短所があります。

余人の言を待たずとも、spam との戦いは楽ではありません。ママからの電子メールとバイアグラ広告を区別する魔法のスイッチはありません。人々は非-spam と spam を区別するのに手を焼いているというのに。それは、spammers が懸命にそれらをママだと思わせようとしているのが本質だからです。Spamming は、世界が彼らに恩義があると思っている人々の一団からの、腹立たしく、無責任で、ばかげた行為です。以下の各章が spam なる疫病との戦いの助けになることを望みます。

10.18.2 Spam 退治の基礎

Spam に対処する一つの方法は、Gnus にすべての spam を‘spam’ メールグループに分離させてしまうことです(see Section 7.4.3 [Splitting Mail], p. 164)。

最初に、あなたに連絡することができる正しいメールアドレスを一つ選び、それをすべてのあなたのニュース記事のFrom ヘッダーに入れましょう。(ここでは‘larsi@trym.ifi.uio.no’を選びましたが、‘larsi+usenet@ifi.uio.no’ の形式のたくさんのアドレスの方が良い選択です。あなたのサイトの sendmail の設定がメールアドレスのローカル部としてどんなキーワードを受け付けるかは、あなたのサイトのシステム管理者に聞いてください。)

```
(setq message-default-news-headers
      "From: Lars Magne Ingebrigtsen <larsi@trym.ifi.uio.no>\n")
```

そしてnnmail-split-fancy に以下の分割規則を入てください(see Section 7.4.6 [Fancy Mail Splitting], p. 175)。

```
(...
 (to "larsi@trym.ifi.uio.no"
     (| ("subject" "re:.*" "misc")
        ("references" ".*@.*" "misc")
        "spam"))
 ...)
```

この意味は、このアドレスに届いたすべてのメールが疑わしいが、‘Re:’ で始まるSubject が付いているか、References ヘッダーが付いていればおそらく OK だろう、ということです。残りはすべて‘spam’ グループに行きます。(このアイデアはおそらく Tim Pierce 氏によるものです。)

これに加えて、多くのメール spam 屋は、あなたのところのSMTP サーバーと直接話して、To ヘッダーにあなたのメールアドレスが明示されないようにします。なんでそんなことをするのかはわかりませんが---もしかしたら、この裏をかく機構の裏をかくためかな? どちらにしても、対処は簡単なことです---特級分割規則を以下のように終端させることによって、あなた宛てでないものを全部‘spam’ グループに入れるだけです:

```
(
 ...
 (to "larsi" "misc")
 "spam")
```

私の経験では、これで実質的にはすべてが正しいグループに分類されます。まあ、それでもときどき‘spam’ グループをチェックして、正しいメールがあるかチェックしなくてはいけませんけどね。あなたが自分が良いネットワーク市民であると思っているなら、それぞれの spam の関係当局に苦情を送り付けることさえもできます---暇なときにでもね。

これで私のところでは動いています。これでみんなは簡単な方法で私に連絡を取ることができ(普通にr を押すだけでできる)、私は spam に煩わされることはまったくありませ

ん。お互いに有利な状況です。私に言わせれば、`From` ヘッダーを偽造して存在しないドメインに送らせるのはすごく良くないです。

訳注: 以上の文章は 1997 年 4 月に書かれました。

この手法には注意してください。Spammers はそれに気付いています。

10.18.3 SpamAssassin, Vipul's Razor, DCC, etc

Spam を避けるための前章のヒントが十分だった日々は過ぎ去りました。今では受け取ったたくさんの spam を減らすという触れ込みの多くの道具があります。この章は、新しい道具が古いものに取って代わって行くにつれてすぐに時代遅れになってしまうでしょうが、幸いなことにこれらのほとんどの道具は類似のインターフェースを持っているようです。この章は例として SpamAssassin を使っていますが、他のほとんどの道具にも簡単に適合するはずです。

この章は`spam.el` パッケージとは関係無いことに注意してください。それは次の章で論じられます。すべての`spam.el` の機能に関心が無いのならば、これらの単純なレシピで間に合わせることができます。

もしあなたが使う道具がメールサーバーにインストールされていないならば、あなた自身がそれを呼び出す必要があります。以下に`:postscript` メールソース指示子(see Section 7.4.4.1 [Mail Source Specifiers], p. 166) を使う場合の考え方を示します。

```
(setq mail-sources
      '((file :prescript "formail -bs spamassassin < /var/mail/%u")
        (pop :user "jrl"
             :server "pophost"
             :postscript
             "mv %t /tmp/foo; formail -bs spamc < /tmp/foo > %t")))
```

いったんメールを受けるスプールをどうにかして処理する、例えばそのメールに spam であることを表示するヘッダーを含めるようにすれば、それをふるい落とす準備は完了です。使うのは普通の分割方式(see Section 7.4.3 [Splitting Mail], p. 164) です:

```
(setq nnmail-split-methods '(("spam"  "^X-Spam-Flag: YES")
                             ...))
```

または特級分割方式(see Section 7.4.6 [Fancy Mail Splitting], p. 175) です:

```
(setq nnmail-split-methods 'nnmail-split-fancy
      nnmail-split-fancy '(| ("X-Spam-Flag" "YES" "spam")
                             ...))
```

いくらかの人たちは`:prescript` を使ってメールをいろんなプログラムにパイプすることを嫌うかもしれません(もし何かのプログラムにバグがあったら、すべてのメールを失ってしまうかもしれません)。あなたがそれらの一人ならば、別の解は分割するときに外部の道具を呼ぶことです。特級分割方式の例です:

```
(setq nnmail-split-fancy '(| (: kevin-spamassassin)
                             ...))
(defun kevin-spamassassin ()
  (save-excursion
    (widen)
    (if (eq 1 (call-process-region (point-min) (point-max)
                                   "spamc" nil nil nil "-c"))
```

```
        "spam")))
```

Nnimap バックエンドの場合、ディフォルトでは記事のボディーがダウンロードされないことに注意してください。それをするためには`nnimap-split-download-body` を`t` に設定する必要があります(see Section 7.3.3 [Client-Side IMAP Splitting], p. 161)。

以上がこれに関することです。ある種の spam はどうしても素通りしてしまいがちなので、spam を読むはめになったときに呼ぶための気の利いた関数が必要でしょう。これがその気の利いた関数です:

```
(defun my-gnus-raze-spam ()
  "SPAM の処理を Vipul の Razor に委ねてから、
それに期限切れ消去可能の印を付けます。"
  (interactive)
  (gnus-summary-save-in-pipe "razor-report -f -d" t)
  (gnus-summary-mark-as-expirable 1))
```

10.19 Spam パッケージ

Spam パッケージは spam を検出して濾過するために集結された機構を Gnus に提供します。それは新着メールを濾過し、spam か ham かに応じてメッセージを処理します。(*Ham* は spam ではないメッセージを示すために、このマニュアルを通して使われる名前です。)

10.19.1 Spam パッケージ序説

Spam パッケージがどのように働くかを理解するために、必ずこの章を読んでください。読み飛ばし、速読、または斜め読みしてはいけません。

`spam.el` シーケンスのイベントの章をちゃんと読みましょう。See Section 10.19.7 [Extending the Spam package], p. 312.

Spam パッケージを使うには、**必ず** 最初に`spam-initialize` 関数を実行させてください:

```
(spam-initialize)
```

これは`spam.el` を自動読み込み(autoload) して、Spam パッケージにその仕事をさせるために必要な諸機能が使えるようにします。Spam パッケージを利用するために、いくつかのグループパラメーターと変数を設定しなければなりません。それらは以下で説明します。Spam パッケージを制御するすべての変数は、‘`spam`’ カスタマイズグループで見つかるでしょう。

Spam パッケージと Gnus には二つの「接点」があります。それは新着メールが spam かどうかを検査するときと、グループを抜け出るときです。

新着メールが spam かどうかの検査は、やって来たメールを分割するときか、グループに入るときのどちらかで行なわれます。

最初のやり方、つまりやって来たメールを分割するときに検査をするのは、新着メールが単一のスプールファイルに入れられる`nnml` や`nnimap` のようなメールバックエンドに適しています。Spam パッケージはやって来たメールを処理し、spam と見なすメールを “spam” 用に指定したグループに送ります。See Section 10.19.2 [Filtering Incoming Mail], p. 297.

二番目のやり方は、`nntp` のような(やって来たメールのためのスプールがない) バックエンドや、やって来たメールの分割をサーバーが担当するバックエンドに適しています。この場合 Gnus のグループに入ると、そのグループにあるまだ読まれたことが無い、または未読になっているメッセージに対して spam かどうかの検査が行なわれます。検出され

た spam メッセージには spam 印が付けられます。See Section 10.19.3 [Detecting Spam in Groups], p. 298.

どちらの場合でも、spam メッセージの検出にどの方法を使うかを Spam パッケージに指示しなければなりません。選択肢として複数の方法、と言うか「spam バックエンド」(Gnus のバックエンドと混同しないでください) があります。spam の「ブラックリスト」と「ホワイトリスト」、辞書に基づいた濾過器、などから選んでください。See Section 10.19.6 [Spam Back Ends], p. 304.

Gnus の概略バッファーで spam だと同定されたメッセージには、常に'`$`' 印が付きます。

Spam パッケージは Gnus のグループを三つに分類します: ham グループ、spam グループ、および分類されないグループです。購読している各グループが ham グループと spam グループのどちらなのかを、`spam-contents` グループパラメーターを使って指定してください(see Section 3.10 [Group Parameters], p. 23)。Spam グループには特別な属性があり、spam グループに入ると、まだ読まれたことが無いすべてのメッセージに spam 印が付きます。そのため、spam グループに分割されたメールには自動的に spam 印が付きます。

Spam メッセージを同定することは、Spam パッケージの仕事の半分に過ぎません。もう半分は、グループを抜け出るときに実行します。このとき Spam パッケージは複数のことを行ないます。

最初に spam か ham かに応じて記事を処理するために*spam and ham processors* を呼び出します。各々の spam バックエンドと連係している spam と ham のプロセッサーの対があって、プロセッサーが行なうことはバックエンドに依存しています。現在のところ spam と ham プロセッサーの主な役割は、辞書に基づいた spam 濾過のためのものです: それらは将来の spam を検出する性能を改良するために、グループにあるメッセージの内容を濾過器の辞書に追加します。`spam-process` グループパラメーターで、どの spam プロセッサーを使うかを指定します。See Section 10.19.4 [Spam and Ham Processors], p. 299.

Spam 濾過器が spam メッセージに印を付けそこなったら、グループを抜け出るときにそのメッセージが spam として処理されるようにするために、あなた自身がそれに印を付けても良いでしょう。

`$`
`M-d`
`M s x`
`S x`
: 現在の記事に spam 印を付けて、'`$`' 印を表示します(`gnus-summary-mark-as-spam`)。

同様に、記事に誤って付けられた spam 印を消すこともできます。See Section 4.7.4 [Setting Marks], p. 64.

普通 ham ではないグループで見つかった ham メッセージは ham として処理されません。つまり、さらに処理されるために、それは ham グループに移動させるべきであるということです(以下を見てください)。しかし`spam-process-ham-in-spam-groups` および`spam-process-ham-in-nonham-groups` を設定することによって、それらの記事を ham として処理することを強制することもできます。

グループを抜け出るときに、二番目に Spam パッケージが行なうことは、ham 記事を spam グループの外へ、spam 記事を ham グループの外へ移動させることです。Spam グループの ham 記事は、変数`gnus-ham-process-destinations` またはグループパラメーター`ham-process-destination` で指定されたグループに移動させられます。Ham グループの spam 記事は、変数`gnus-spam-process-destinations` またはグループパラメータ

ーspam-process-destination で指定されたグループに移動させられます。これらの変数が設定されていない場合、記事はそれらの現在のグループに残されます。記事を移動させることができない場合(例えばNNTP のような読み出し専用のバックエンドでは)、代わりに記事がコピーされます。

記事が別のグループに移動させられると、その新しいグループを訪れたときに、記事は再び処理されます。普通これは問題になりませんが、それぞれの記事が一回だけ処理されるようにしたいならば、gnus-registry.el パッケージを読み込んで、変数spam-log-to-registry をt に設定してください。See Section 10.19.5 [Spam Package Configuration Examples], p. 301.

普通 spam グループはgnus-spam-process-destinations を無視します。しかしspam-move-spam-nonspam-groups-only をnil に設定すると、spam-process-destination パラメーターに従って spam は spam グループの外へ移動させられます。

最後に Spam パッケージが行なうことは、spam 記事に期限切れ消去の印を付けることです。普通それは正しい行ないです。

これらのすべてがわけがわからなくても、心配は要りません(訳注: でも訳文が正確ではないかもしれないので、変だと思ったら原文を見てね:-p)。すぐにそれは神経インターフェース上に Lisp で小話を書くように自然なことになります... え゛ごめん、それにはまだ 50 年早いですね。ただ私たちを信頼してください。それは捨てたものではありません。

10.19.2 やって来るメールの濾過

やって来るメールを濾過するために Spam パッケージを使うには、最初に特級メール分割を使うための設定を行なってください。See Section 7.4.6 [Fancy Mail Splitting], p. 175. Spam パッケージは、特級分割のための変数(メールバックエンドによるがnnmail-split-fancy またはnnimap-split-fancy) に追加することができる、特別な分割関数を定義します:

```
(: spam-split)
```

spam-split 関数は、あなたが選んだ spam バックエンド(一つまたは複数) に応じて、やって来たメールを走査します。ディフォルトでは spam グループは‘spam’ という名前のグループですが、spam-split-group をカスタマイズすることによって変更することができます。spam-split-group の値に Gnus のバックエンド名やサーバー名を含めないでください。例えば‘your-server’ というnnimap のサーバーでは、‘spam’ という名前は‘nnimap+your-server:spam’ を意味します。したがって‘nnimap+server:spam’ という値は誤りで、それは‘nnimap+your-server:nnimap+server:spam’ というグループを指すことになってしまいます。

spam-split はいかなる方法によってもメッセージの内容を変更しません。

IMAP の利用者への注意: spam バックエンドとしてspam-check-bogofilter、spam-check-ifile およびspam-check-stat を使う場合は、さらに変数nnimap-split-download-body をt に設定しなければなりません。See Section 7.3.3 [Client-Side IMAP Splitting], p. 161.

spam-use-* 変数を使って、spam-split が使う一つ以上の spam バックエンドを設定しなければなりません。See Section 10.19.6 [Spam Back Ends], p. 304. 通常spam-use-* は、あなたがこのようにして設定したすべての spam バックエンドを、単に使います。しかし、それらのいくつかだけを使うように、spam-split に指示することもできます。それがなぜ役に立つのかって? Spam バックエンドとしてspam-use-regex-headers とspam-use-blackholes を使っていて、かつ以下の分割規則を使っているとすると:

```
nnimap-split-fancy '(|
```

```
                (any "ding" "ding")
                (: spam-split)
                ;; ディフォルトのメールボックス。
                "mail")
```

問題は ding 宛てのメッセージをすべて ding フォルダーに入れようとしていることです。でもそれは、ding メーリングリスト宛てに送られた明らかな spam (例えば SpamAssassin と`spam-use-regex-headers` によって検出される spam) を許してしまうでしょう。一方、いくつかの ding 宛てのメッセージはブラックホールリストに載っているメールサーバーからやって来るので、ding の規則より前に`spam-split` を発動することができません。

解決策は SpamAssassin ヘッダーに ding の規則を置き換えさせ、ding の規則の後で別の`spam-split` の規則(二つ目の正規表現によるヘッダーの検査を含む) を作動させることです。これはパラメーターを`spam-split` に渡すことによって行なわれます:

```
nnimap-split-fancy
      '(|
        ;; spam-use-regex-headers で検出された
        ;; spam は 'regex-spam' へ。
        (: spam-split "regex-spam" 'spam-use-regex-headers)
        (any "ding" "ding")
        ;; spam-split で検出された他のすべての spam は
        ;; spam-split-group へ。
        (: spam-split)
        ;; ディフォルトのメールボックス。
        "mail")
```

これは、あなたの特別な必要に応じた特定の`spam-split` 検査を起動し、それらの検査の結果で特定の spam グループを指し示します。すべてのメールに対して、すべての spam 検査を行なう必要はありません。これが良いもう一つの理由は、分割規則を設定してあるメーリングリスト宛てのメッセージに対して、資源集約的なブラックホール・チェックを実行する必要がないということです。さらに、nnmail の分割のために nnimap のものとは異なる spam 検査のやり方を設定することもできるでしょう。気が狂うー。

使用するどんな spam バックエンドにも`spam-use-*` 変数を設定するべきです。そのわけは、`spam.el` を読み込むときに、どんな`spam-use-xyz` 変数を設定したかによって、何らかの条件付きの読み込みが行なわれるからです。See Section 10.19.6 [Spam Back Ends], p. 304.

10.19.3 グループにおける spam の検出

グループに入ったときに spam を検出するためには、そのグループの`spam-autodetect` と`spam-autodetect-methods` グループパラメーターを設定してください。これらは通常とおり*G c* か*G p* で行なうことができます(see Section 3.10 [Group Parameters], p. 23)。

使用するどんな spam バックエンドにも`spam-use-*` 変数を設定するべきです。そのわけは、`spam.el` を読み込むときに、どんな`spam-use-xyz` 変数を設定したかによって、何らかの条件付きの読み込みが行なわれるからです。

ディフォルトでは、まだ読まれたことがない記事だけが spam かどうかを検査されます。`spam-autodetect-recheck-messages` を`t` に設定することによって、グループにあるすべての記事の再検査を Gnus に強制することができます。

Spam の検査にspam-autodetect の手段を使う場合は、異なるグループで違う spam 検出手段を指定することができます。例えば‘ding’ グループは自動検出の手段としてspam-use-BBDB を持つことができる一方で、‘suspect’ グループではspam-use-blacklist およびspam-use-bogofilter の手段を使うことができます。spam-split と違って検査の順序を制御できませんが、これはたぶん重要ではありません。

10.19.4 Spam と Ham プロセッサー

Spam と ham プロセッサーには、グループバッファーを抜け出るときに行なう動作に関して特別な性質があります。Spam プロセッサーは spam メッセージに作用し、ham プロセッサーは ham メッセージに作用するということです。現在のところ、これらのプロセッサーの主な役割は、Bogofilter (see Section 10.19.6.6 [Bogofilter], p. 308) や Spam 統計パッケージのような辞書に基づいた spam バックエンドの辞書を更新することです(see Section 10.19.6.9 [Spam Statistics Filtering], p. 310)。

それぞれのグループに適用される spam と ham プロセッサーは、そのグループのspam-process グループパラメーターで決定されます。このグループパラメーターが定義されていないと、それらは変数gnus-spam-process-newsgroups によって決められます。

Gnus はあなたが受け取った spam から学びます。あなたは一つ以上の spam グループに spam 記事を集めて、変数spam-junk-mailgroups を適切に設定もしくはカスタマイズしなければなりません。また、spam を含めるグループを、そのグループパラメーターspam-contents をgnus-group-spam-classification-spam に設定するか、またはそれに対応する変数gnus-spam-newsgroup-contents をカスタマイズすることによって宣言することができます。spam-contents グループパラメーターとgnus-spam-newsgroup-contents 変数は、それらの種別をgnus-group-spam-classification-ham に設定することによって、*ham* グループであることを宣言するために使うこともできます。グループがspam-junk-mailgroups, spam-contents またはgnus-spam-newsgroup-contents であることを示す分類が行なわれていないと、それらは未分類 であると解釈されます。すべてのグループはディフォルトでは未分類です。

Spam グループでは、ディフォルトですべてのメッセージが spam であると解釈されます: そのグループに入ると、それらに‘$’ 印(gnus-spam-mark) が付きます。あるメッセージを見て、いったんそれに spam の印を付けても、後で取り消せば、その後そのグループに入ったときに、それには spam 印は付きません。spam-mark-only-unseen-as-spam パラメーターをnil にすれば、そういう動作をやめさせる、つまりすべての未読メッセージに spam 印が付くようにすることができます。そのグループの概略バッファーにいるときに、やっぱり spam ではなかったとわかった記事があったら、それらのすべてから‘$’ 印を消さなければなりません。‘$’ 印を消すには*M-u* でその記事を「未読」にするか、あるいは*d* を使って spam ではないものとして読んだことを宣言すれば良いでしょう。グループを抜けるとき、すべての spam 印(‘$’) が付いた記事は spam プロセッサーに送られ、それらを spam の標本として学習します。

メッセージは他のいろいろな方法によっても消去されるかもしれないし、ham-marks グループパラメーターが無効にされなければ、‘R’ 印と‘r’ 印、および‘X’ 印と‘K’ は、‘Y’ 印と同様に、すべて spam では無い記事に関連付けられるものと解釈されます(それぞれ‘R’ はディフォルトの既読の印、‘r’ 明示的な消去の印、‘X’ は自動的な削除の印、‘K’ は明示的な削除の印、そして‘Y’ は低いスコアのため印です)。この仮定は、特に真性の spam を検出するために消去(kill) ファイルかスコアファイルを使っている場合は、間違いかもしれません。そうであればham-marks グループパラメーターを調整するべきです。

`ham-marks` [変数]
: このグループまたはトピックパラメーターを ham であると解釈したい印のリストに設定することができます。ディフォルトでは、消去(deleted)、既読(read)、削除(killed)、kill ファイルにあるもの(kill-filed) および低いスコア(low-score、既読だけれども spam ではないと考える) 印のリストです。Ham 印のリストに可視(tick) 印を含めることが役立つこともあります。未読印を ham 印にすることは、通常それが分類されていないことを表すので、勧められません。しかし、あなたがそれを行なうことはできるし、私たちに不満はありません。

`spam-marks` [変数]
: このグループまたはトピックパラメーターを spam であると解釈したい印のリストに設定することができます。ディフォルトでは spam 印だけを持つリストです。それを変更することは勧めませんが、本当にそうしたいのならご勝手に。

グループを抜けるときに(そのグループが何 であっても)、その`spam-contents` の分類にかかわらず、spam 印が付いているすべての記事は spam プロセッサーに送られ、それらを spam の標本として学習します。意図的にたくさんの消去を行なうと、たまにそれは見ていない‘`K`’ 印が付いた記事群で終わるかもしれません。そしてそれらは偶然に spam を含んでいるかもしれません。最も良いのは、本当の spam に‘`$`’ が付いていて、他に何も印が無いことを確かめることです。

Spam グループを抜けるときに、spam 印が付いているすべての記事には spam プロセッサーで処理した後で期限切れ消去の印が付けられます。これは未分類 または*ham* グループに対しては行なわれません。さらに spam グループにあるどの**ham** 記事も、`ham-process-destination` グループパラメーターが示す場所か`gnus-ham-process-destinations` 変数の中で合致する場所のどちらかに移されます。後者はグループ名に合致する正規表現のリストです(*`M-x customize-variable`* `RET` *`gnus-ham-process-destinations`* によってこの変数をカスタマイズするのが最も簡単です)。変数を手でカスタマイズする方が好きな人のために言っておくと、それぞれのグループ名のリストは普通の Lisp の list です。`ham-process-destination` パラメーターが設定されていないと、ham 記事は移動させられません。`spam-mark-ham-unread-before-move-from-spam-group` パラメーターが設定されていると、ham 記事には移動させられる前に未読の印が付けられます。

例えばNNTP のような読み込み専用バックエンドであるために ham が移動できない場合、それはコピーされます。

グループごとに、または正規表現に合致するグループごとに、複数の移動先を指定できることに注目してください! これによって ham 記事を正規のメールグループと*ham* トレーニング グループに送ることができます。

Ham グループを抜けるときに、ham 印が付いているすべての記事は ham プロセッサーに送られ、それらを spam ではない標本として学習します。

変数`spam-process-ham-in-spam-groups` はディフォルトでは`nil` です。Spam グループで見つかった ham がプロセッサーに送られるようにしたい場合は`t` にしてください。通常これは行なわれません。その代わり、あなたが自分で ham 記事を ham グループに送って、そこで処理することが想定されています。

変数`spam-process-ham-in-nonham-groups` はディフォルトでは`nil` です。Ham ではない(spam または未分類の) グループで見つかった ham がプロセッサーに送られるようにしたい場合は`t` にしてください。通常これは行なわれません。その代わり、あなたが自分で ham 記事を ham グループに送って、そこで処理することが想定されています。

Ham または未分類 グループを抜けるときに、すべての**spam** 記事は、`spam-process-destination` グループパラメーターが示す場所か`gnus-spam-process-destinations` 変数の中で合致する場所のどちらかに移されます。後者はグループ名に合致する正規表現のリストです(*M-x customize-variable* RET *gnus-spam-process-destinations* によってこの変数をカスタマイズするのが最も簡単です)。変数を手でカスタマイズする方が好きな人のために言っておくと、それぞれのグループ名のリストは普通の Lisp の list です。`spam-process-destination` パラメーターが設定されていないと、spam 記事は単に期限切れ消去されます。グループ名は完全形であること、すなわちグループバッファーでグループ名の前に‘`nntp:servername`’ のようなものが見える場合は、ここでもそれを使う必要があります。

例えばNNTP のような読み込み専用バックエンドであるために spam が移動できない場合、それはコピーされます。

グループごとに、または正規表現に合致するグループごとに、複数の移動先を指定できることに注目してください! これによって spam 記事を正規のメールグループと*spam* トレーニング グループに送ることができます。

Ham と spam に関する問題は、Gnus がディフォルトではこの処理を追跡してくれないことです。複数回にわたって処理することを回避するために、`spam.el` が`gnus-registry.el` を使って処理された記事を追跡するように、`spam-log-to-registry` 変数を有効にしてください。`gnus-registry.el` が登録する数を制限してしまうと、制限が無い場合のように動作しないことを覚えておいてください。

Spam グループにある、まだ読まれたことが無い記事だけに spam の印を付けたい場合は、この変数をセットしてください。ディフォルトではセットされています。これを`nil` にすると、未読の記事にも spam 印が付けられます。

Ham が spam グループから移動される前に印を消したい場合は、この変数をセットしてください。これは ham に印を付けるために可視(tick) 印(‘`!`’) のようなものを使う場合に、とても役に立ちます。記事はあたかもそれがメールサーバーから来たばかりのように、無印で`ham-process-destination` に置かれるでしょう。

この変数は`spam.el` が spam の自動検出を行なう場合に、まだ読まれたことが無い記事だけか、またはすべての未読記事のどちらに対して spam 検査を行なうかを指示します。これはそのままにしておくことを勧めます。

10.19.5 Spam パッケージの設定例

Ted の設定

From Ted Zlatanov <tzz@lifelogs.com>.

```
;; gnus-registry-split-fancy-with-parent と spam の自動検出のため。
;; 詳細は gnus-registry.el を参照。
(gnus-registry-initialize)
(spam-initialize)

(setq
 spam-log-to-registry t ;; Spam の自動検出のため。
 spam-use-BBDB t
 spam-use-regex-headers t ;; X-Spam-Flag (SpamAssassin) を捕まえる。
 ;; 名前に ‘spam’ を含むすべてのグループには spam 記事がある。
```

```
 gnus-spam-newsgroup-contents '(("spam"
                                 gnus-group-spam-classification-spam))
 ;; これらの docstring を参照。
 spam-move-spam-nonspam-groups-only nil
 spam-mark-only-unseen-as-spam t
 spam-mark-ham-unread-before-move-from-spam-group t
 ;; あなたの設定に追加する前に、これが何をするか理解せよ!
 ;; nnimap 用にたぶん nnimap-split-methods を設定する必要あり。
 ;; マニュアル参照。
 nnimap-split-fancy '(|
                      ;; References を親まで辿ってそれらのグループ
                      ;; を入れる。
                      (: gnus-registry-split-fancy-with-parent)
                      ;; これはサーバー側の SpamAssassin タグを捕ま
                      ;; える。
                      (: spam-split 'spam-use-regex-headers)
                      (any "ding" "ding")
                      ;; Spam 記事はディフォルトで 'spam' に行く
                      ;; ことに注意。
                      (: spam-split)
                      ;; ディフォルトのメールボックス。
                      "mail"))

;; G p で設定した私のパラメーター。

;; すべての nnml グループと、'nnimap+mail.lifelogs.com:train' と
;; 'nnimap+mail.lifelogs.com:spam' を除いたすべての nnimap グループ
;; のためのパラメーター:
;; それは手動で検出したはずなので、どの spam も nnimap のトレーニン
;; ググループに送り込む。

((spam-process-destination . "nnimap+mail.lifelogs.com:train"))

;; すべての NNTP グループのためのパラメーター:
;; Spam を blacklist で、ham を BBDB で自動検出。
((spam-autodetect-methods spam-use-blacklist spam-use-BBDB)
;; すべての spam をトレーニンググループに送る。
 (spam-process-destination . "nnimap+mail.lifelogs.com:train"))

;; 私が spam を自動検出させたい、ほんのいくつかの NNTP グループ
;; のためのパラメーター:
((spam-autodetect . t))

;; 私の nnimap 'nnimap+mail.lifelogs.com:spam' グループ (これは
;; spam グループ) のためのパラメーター:

((spam-contents gnus-group-spam-classification-spam)
```

```
;; どんな spam も私が ham 印を付けなければ
;; 'nnimap+mail.lifelogs.com:train' に送り込まれる。(前述の
;; gnus-spam-newsgroup-contents の設定により、すべての
;; まだ読まれたことが無いメッセージを読むとそうなる。)

(spam-process-destination "nnimap+mail.lifelogs.com:train")

;; どんな ham も私の 'nnimap+mail.lifelogs.com:mail' フォルダー
;; に送り込まれるが、私の 'nnimap+mail.lifelogs.com:trainham'
;; フォルダーにもトレーニングのために送り込む。

(ham-process-destination "nnimap+mail.lifelogs.com:mail"
                         "nnimap+mail.lifelogs.com:trainham")
;; このグループでは '!' 印が付いているものだけが ham。
(ham-marks
 (gnus-ticked-mark))
;; グループを抜けるときに blacklist に送信者を覚えさせる---これは
;; 明らかに不要で、単に私の鬱憤を晴らすためにある。
(spam-process (gnus-group-spam-exit-processor-blacklist)))

;; その後 IMAP サーバー上で、私は SpamAssassin が spam を
;; 認識するトレーニングのために 'train' グループを、ham を
;; 認識するトレーニングのために 'trainham' グループを使う。
;; でも Gnus はそういうことはやってくれない。
```

サーバー上の IMAP サーバーで、統計的な濾過器とspam.el を使う

From Reiner Steib <reiner.steib@gmx.de>.

私のプロバイダーは(IMAP) メールサーバー上で(DCC と連係した) bogofilter を立ち上げました。認識された spam は'spam.detected' へ行き、残りには通常の濾過規則が適用される、すなわち'some.folder' か'INBOX' に行きます。誤検出と見逃しのトレーニングは、それぞれ'training.ham' または'training.spam' に記事をコピーするか移動させることによって行なわれます。サーバー上の cron ジョブが、それらを適切な ham または spam オプションとともに bogofilter に与え、'training.ham' および'training.spam' フォルダーからそれらを削除します。

以下のgnus-parameters の要素群によって、spam.el はほとんどの仕事を私のためにこなします:

```
("nnimap:spam\\.detected"
 (gnus-article-sort-functions '(gnus-article-sort-by-chars))
 (ham-process-destination "nnimap:INBOX" "nnimap:training.ham")
 (spam-contents gnus-group-spam-classification-spam))
("nnimap:\\(INBOX\\|other-folders\\)"
 (spam-process-destination . "nnimap:training.spam")
 (spam-contents gnus-group-spam-classification-ham))
```

- **The Spam folder:** 'spam.detected' フォルダーにおいて、私は誤検出(すなわち bogofilter

か DCC が spam であると誤って判定した正当なメール) のチェックをしなければなりません。

`gnus-group-spam-classification-spam` の項のために、すべてのメッセージには spam の印(`$`) が付けられます。誤検出を見つけたら、私は記事にいくつかの他の ham 印(`ham-marks`, Section 10.19.4 [Spam and Ham Processors], p. 299) を付けます。グループを出るとき、それらの記事は‘`INBOX`’ (私が記事を置いておきたいところ) と‘`training.ham`’ (bogofilter のトレーニング用) の両方のグループにコピーされ、‘`spam.detected`’ フォルダーから削除されます。

`gnus-article-sort-by-chars` の項は、私の誤検出の発見を簡単にします。私は、すべて似たサイズの、たくさんのワーム(sweN, ...) を受け取ります。それらをサイズ(つまり文字数) でまとめると、他の誤検出を見つけやすくなるのです。(もちろん厳密にはワームは*spam* ではありません。ともあれ、それらの要らないメールを濾過するのに bogofilter は私にとって優秀な道具です。)

- **Ham folders:** 私の ham フォルダーで、認識されなかった spam メール(見逃し) を見つけたときはいつでも、私は単に*S x* (`gnus-summary-mark-as-spam`) を叩きます。グループを出るとき、それらのメッセージは‘`training.spam`’ に移されます。

`spam-report.el` で **Gmane** グループの **spam** を報告する

From Reiner Steib <reiner.steib@gmx.de>.

以下の`gnus-parameters` に納めた要素によって、*S x* (`gnus-summary-mark-as-spam`) で`gmane.*` グループの spam 記事に印を付け、グループを出るときに Gmane に報告します:

```
("^gmane\\."
 (spam-process (gnus-group-spam-exit-processor-report-gmane)))
```

加えて、私は news.gmane.io からではなくローカルニュースサーバー(leafnode) を通して記事を読んでいるので、`(setq spam-report-gmane-use-article-number nil)` を使っています。つまり、記事番号が news.gmane.io におけるものと異なるので、正しい記事番号を見つけるために`spam-report.el` に`X-Report-Spam` ヘッダーを検査させなければなりません。

10.19.6 Spam バックエンド

Spam パッケージは spam を検出するための様々なバックエンドを提供します。それぞれのバックエンドでは、spam を検出する手段の組(see Section 10.19.2 [Filtering Incoming Mail], p. 297, see Section 10.19.3 [Detecting Spam in Groups], p. 298) と spam および ham プロセッサーの対が定義されています(see Section 10.19.4 [Spam and Ham Processors], p. 299)。

10.19.6.1 ブラックリストとホワイトリスト

`spam-use-blacklist` [変数]
: 入ってくるメールを分割するときにブラックリストを使いたい場合は、この変数を`t` に設定してください。送信者がブラックリストに載っているメッセージは`spam-split-group` に送られます。これは、送信者が spammer であることが定義されているメールに対してだけ動作する、明示的な濾過器です。

`spam-use-whitelist` [変数]
: 入ってくるメールを分割するときにホワイトリストを使いたい場合は、この変数を`t` に設定してください。送信者がホワイトリストに載っていないメッセージは、次の

spam-split 規則(による検査工程) に送られます。これは、ホワイトリストに載っていない誰かのメッセージは spam とも ham とも見なされないことを意味する、明示的な濾過器です。

`spam-use-whitelist-exclusive` [変数]
送信者がホワイトリストに載っていないすべてのメッセージが spam だと見なされることを意味する暗黙の濾過器としてホワイトリストを使いたい場合は、この変数を`t`にしてください。注意して使ってください。

`gnus-group-spam-exit-processor-blacklist` [変数]
このシンボルを、グループパラメーターのカスタマイズによってグループの`spam-process` パラメーターに加えるか、または`gnus-spam-process-newsgroups` 変数に加えてください。このシンボルがグループの`spam-process` パラメーターに加えられると、spam 印が付いた記事の送信者がブラックリストに追加されます。

警告

旧式の`gnus-group-spam-exit-processor-blacklist` の代わりに`(spam spam-use-blacklist)` を使うことを推奨します。すべて同等に動作することは保証されます。

`gnus-group-ham-exit-processor-whitelist` [変数]
このシンボルを、グループパラメーターのカスタマイズによってグループの`spam-process` パラメーターに加えるか、または`gnus-spam-process-newsgroups` 変数に加えてください。このシンボルがグループの`spam-process` パラメーターに加えられると、ham 印が付いた*ham* グループの記事の送信者がホワイトリストに追加されます。

警告

旧式の`gnus-group-ham-exit-processor-whitelist` の代わりに`(ham spam-use-whitelist)` を使うことを推奨します。すべて同等に動作することは保証されます。

ブラックリストは、あなたが spam の送信者だと考えるアドレスに合致する正規表現のリストです。例えば‘`vmadmin.com`’ の誰からでも来るメールを阻止するには、あなたのブラックリストに‘`vmadmin.com`’ を入れてください。空のブラックリストで始めましょう。ブラックリストの各項目は Emacs の正規表現の構文を使います。

逆に、ホワイトリストは何のアドレスが正当だと考えられるかを告げます。ホワイトリストにあるアドレスからやって来たすべてのメッセージは、非-spam だと見なされます。Section 10.19.6.2 [BBDB Whitelists], p. 306, も見てください。ホワイトリストの各項目は Emacs の正規表現の構文を使います。

ブラックリストとホワイトリストのファイルの所在は、`spam-directory` 変数(ディフォルトは`~/News/spam`) または直接`spam-whitelist` と`spam-blacklist` 変数でカスタマイズすることができます。ホワイトリストとブラックリストのファイルは、ディフォルトでは`spam-directory` のディレクトリーにあり、それぞれ`whitelist` と`blacklist` という名前が付けられます。

10.19.6.2 BBDB ホワイトリスト

`spam-use-BBDB` [変数]

`spam-use-whitelist` (see Section 10.19.6.1 [Blacklists and Whitelists], p. 304) に似ていますが、ホワイトリストのアドレスの源として BBDB を使います。正規表現はありません。`spam-use-BBDB` をちゃんと動作させるには BBDB を読み込まなければ(load しなければ) なりません。その送信者が BBDB に載っていないメッセージは、次の spam-split 規則(による検査工程) に送られます。これは、BBDB に載っていない誰かのメッセージは spam とも ham とも見なされないことを意味する、明示的な濾過器です。

`spam-use-BBDB-exclusive` [変数]

送信者が BBDB に載っていないすべてのメッセージが spam だと見なされることを意味する暗黙の濾過器として BBDB を使いたい場合は、この変数を`t` にしてください。注意して使ってください。BBDB に載っている送信者だけが通行を許され、他のすべては spammers として分類されます。

`spam.el` に関する限りは、`spam-use-BBDB` の別名として`spam-use-BBDB-exclusive` を使うことができますが、それは別のバックエンドではありません。`spam-use-BBDB-exclusive` を`t` に設定すれば、BBDB による分割はすべて排他的になります。

`gnus-group-ham-exit-processor-BBDB` [変数]

このシンボルを、グループパラメーターのカスタマイズによってグループの`spam-process` パラメーターに加えるか、または`gnus-spam-process-newsgroups` 変数に加えてください。このシンボルがグループの`spam-process` パラメーターに加えられると、ham 印が付いた*ham* グループの記事の送信者が BBDB に追加されます。

警告

旧式の`gnus-group-ham-exit-processor-BBDB` の代わりに、`(ham spam-use-BBDB)` を使うことを推奨します。すべて同等に動作することは保証されます。

10.19.6.3 Gmane Spam 報告

`gnus-group-spam-exit-processor-report-gmane` [変数]

グループパラメーターか変数`gnus-spam-process-newsgroups` をカスタマイズして、このシンボルをグループの`spam-process` パラメーターに加えてください。これが加えられると、spam 印が付いた記事のグループが HTTP 経由で Gmane の管理者に報告されます。

Gmane はかつて`http://gmane.org` にありました。

警告

旧式の`gnus-group-spam-exit-processor-report-gmane` の代わりに`(spam spam-use-gmane)` を使うことを推奨します。すべて同等に動作することは保証されます。

`spam-report-gmane-use-article-number` [変数]

この変数はディフォルトで`t` です。例えばあなた自身がニュースサーバーを運営しているなどの理由によって、ローカルな記事番号が Gmane の記事番号と合わない場合は、`nil` に設定してください。`spam-report-gmane-use-article-number` が`nil` であると、`spam-report.el` はその番号を記事のヘッダーから取得します。

`spam-report-user-mail-address` [変数]
Gmane への spam の報告に付加される User-Agent に現れるメールアドレスです。これは、誤った報告が行なわれたときに、Gmane の管理者があなたに連絡できるようにするためのものです。ディフォルトは`user-mail-address` です。

10.19.6.4 ブラックホール

`spam-use-blackholes` [変数]
このオプションはディフォルトで無効になっています。このオプションをセットすると、Gnus にブラックホール型の分散 spam 処理システム(例えば DCC) を調べさせることができます。変数`spam-blackhole-servers` は、Gnus が意見を求めるブラックホール・サーバーのリストを持ちます。現在のリストはかなり広範囲に渡っていますが、もし時代遅れなサーバーを含んでいたら必ず私たちに知らせるようにしてください。
ブラックホール・チェックは`dig.el` パッケージを使います。しかし`spam-use-dig` を`nil` に設定すれば、より良い性能のために`dns.el` を代わりに使うことを`spam.el` に指示することができます。現状では`spam-use-dig` を`nil` に設定することは、いく人かの利用者が使えないかもしれないので、それが可能な性能改善であるにもかかわらず推奨されません。しかし、それが動くかどうかを確かめることはできます。

`spam-blackhole-servers` [変数]
ブラックホール・チェックのために意見を求めるサーバーのリストです。

`spam-blackhole-good-server-regex` [変数]
ブラックホール・サーバーのリストと照合されてはならない IP の正規表現です。`nil` に設定されると無効になります。

`spam-use-dig` [変数]
`dns.el` パッケージの代わりに`dig.el` パッケージを使います。ディフォルトの設定である`t` が推奨されます。

ブラックホール・チェックは入って来るメールに対してだけ行なわれます。ブラックホールに spam または ham プロセッサーはありません。

10.19.6.5 正規表現によるヘッダーの合致検査

`spam-use-regex-headers` [変数]
このオプションはディフォルトで無効になっています。このオプションをセットすると、Gnus に正規表現のリストとメッセージヘッダーを照合させることができます。変数`spam-regex-headers-spam` および`spam-regex-headers-ham` が正規表現のリストを持ちます。メッセージが spam か ham かどうかをそれぞれの変数を使って決めるために、Gnus はメッセージヘッダーを検査します。

`spam-regex-headers-spam` [変数]
メッセージヘッダーの中で一致した時に、それが spam であることを断定するための正規表現のリストです。

`spam-regex-headers-ham` [変数]
メッセージヘッダーの中で一致した時に、それが ham であることを断定するための正規表現のリストです。

正規表現によるヘッダーの検査は、入ってきたメールに対してだけ行なわれます。正規表現のために特有な spam または ham プロセッサーはありません。

10.19.6.6 Bogofilter

`spam-use-bogofilter` [変数]
: Eric Raymond の迅速な Bogofilter を`spam-split` に使用したい場合は、この変数をセットしてください。

  Spam 記事に'`$`' 印を関連付ける最小限度の世話だけで、Bogofilter トレーニングはすべてかなり自動的になります。Spam とそうでないものそれぞれの種類について数百通ずつの記事を入手するまで、これをやらなければなりません。デバッグまたは好奇心のどちらかのために概略モードで`S t` コマンドを使うことによって、現在の記事の*spam* 度 (spamicity) スコア(0.0〜1.0) を表示させることができます。

  Bogofilter はメッセージが spam かどうかを、ある明確な閾値に基づいて見極めます。閾値はカスタマイズできます。Bogofilter のドキュメントを調べてください。

  Path に`bogofilter` の実行ファイルが無い場合、Bogofilter の処理は取り消されます。

  `spam-use-bogofilter-headers` を使う場合は、これを有効にしてはいけません。

`M s t`
`S t`
: Bogofilter の spam 度スコアを得ます(`spam-bogofilter-score`)。

`spam-use-bogofilter-headers` [変数]
: メッセージヘッダーだけを調べるために Eric Raymond の迅速な Bogofilter を`spam-split` に使用したい場合は、この変数をセットしてください。これは`spam-use-bogofilter` と同じように動作しますが、あらかじめ`X-Bogosity` ヘッダーがメッセージに存在しなければなりません。通常これは procmail の技法か、何かそれに似たもので行なうことになるでしょう。Bogofilter のインストールに関する文書を調べてください。

  `spam-use-bogofilter` を使う場合は、これを有効にしてはいけません。

`gnus-group-spam-exit-processor-bogofilter` [変数]
: このシンボルを、グループパラメーターのカスタマイズによってグループの`spam-process` パラメーターに加えるか、または`gnus-spam-process-newsgroups` 変数に加えてください。このシンボルがグループの`spam-process` パラメーターに加えられると、spam 印が付いた記事が bogofilter の spam データベースに加えらます。

  警告

  旧式の`gnus-group-spam-exit-processor-bogofilter` の代わりに`(spam spam-use-bogofilter)` を使うことを推奨します。すべて同等に動作することは保証されます。

`gnus-group-ham-exit-processor-bogofilter` [変数]
: このシンボルを、グループパラメーターのカスタマイズによってグループの`spam-process` パラメーターに加えるか、または`gnus-spam-process-newsgroups` 変数に加えてください。このシンボルがグループの`spam-process` パラメーターに加えられると、ham 印が付いた*ham* グループの記事が非-spam 記事用の Bogofilter データベースに追加されます。

警告

旧式のgnus-group-ham-exit-processor-bogofilter の代わりに(ham spam-use-bogofilter) を使うことを推奨します。すべて同等に動作することは保証されます。

spam-bogofilter-database-directory [変数]
これは Bogofilter がそのデータベースを格納するディレクトリーです。ディフォルトでは設定されていないので、Bogofilter はそれ自身のディフォルトのデータベース・ディレクトリーを使います。

Bogofilter のメール分類器は、意図と目的の点でifile に似ています。Ham および spam のプロセッサーが提供され、記事で Bogofilter が使われるべきか、または既に使われたかを spam-split に示すためのspam-use-bogofilter 変数とspam-use-bogofilter-headers があります。この機能を検査するために Bogofilter のバージョン 0.9.2.1 が使われました。

10.19.6.7 SpamAssassin back end

spam-use-spamassassin [変数]
spam-split に SpamAssassin を使いたい場合は、この変数をセットしてください。

SpamAssassin は、規則と分析のセット(ベイジアンフィルタを含む) に基づいて、それぞれの記事のスコアを裁定します。ベイジアンフィルタは、spam 記事に'$' 印を関連させることによって訓練することができます。Spam のスコアは、概略モードで*S t* コマンドを使うことによって見ることができます。

この変数をセットすると、それぞれの記事はspam-split が呼ばれるときに SpamAssassin によって処理されます。メールが SpamAssassin で処理されるようになっていて、SpamAssassin ヘッダーだけを使いたいのならば、代わりにspam-use-spamassassin-headers をセットしてください。

spam-use-spamassassin-headers を使う場合、これを有効にしてはいけません。

spam-use-spamassassin-headers [変数]
メールが SpamAssassin で処理されるようになっている場合に、SpamAssassin ヘッダーに基づいてspam-split に分割を行なわせたいのならば、この変数をセットしてください。

spam-use-spamassassin を使う場合、これを有効にしてはいけません。

spam-spamassassin-program [変数]
この変数は SpamAssassin の実行形式を指します。spamd を稼働しているならば、より速い処理のために、この変数にspamc の実行形式を設定することができます。spamd/spamc の更なる情報は、SpamAssassin のドキュメントを見てください。

SpamAssassin は、spam を同定するために広範な分析を行なう、強力で融通性のある spam 濾過器です。Ham および spam のプロセッサーが提供され、記事で SpamAssassin が使われるべきか、または既に使われたかを spam-split に示すためのspam-use-spamassassin 変数とspam-use-spamassassin-headers 変数があります。この機能を検査するために SpamAssassin のバージョン 2.63 が使われました。

10.19.6.8 ifile による spam の濾過

`spam-use-ifile` [変数]
Bogofilter に似た統計分析器である`ifile` を`spam-split` に使いたい場合は、この変数を有効にしてください。

`spam-ifile-all-categories` [変数]
`spam-use-ifile` に、単なる spam/非-spam ではなくて ifile のすべての区分(カテゴリー) を与えてもらいたいならば、この変数を有効にしてください。これを使う場合は、その文献に書かれているように ifile をトレーニングしておかなければなりません。

`spam-ifile-spam-category` [変数]
ifile に関する限り、これは spam メッセージのカテゴリーです。実際に使われる文字列は無関係ですが、たぶんあなたは‘`spam`’ のディフォルト値を残しておきたいでしょう。

`spam-ifile-database` [変数]
これは ifile データベースのファイル名です。ディフォルトでは定義されていないので、ifile はそれ自身のディフォルトのデータベース名を使います。

ifile のメール分類器は、意図と目的の点で Bogofilter に似ています。Spam と ham のプロセッサーが提供され、ifile が使われるべきであることを spam-split に示すための`spam-use-ifile` 変数があります。この機能を検査するために ifile のバージョン 1.2.1 が使われました。

10.19.6.9 Spam 統計濾過

このバックエンドは、統計に基づいた濾過を行なう Spam 統計 Emacs Lisp パッケージを使います(see Section 10.19.8 [Spam Statistics Package], p. 314)。これを使う前に、あなたの Spam 統計辞書を初期化するための、いくつかの追加の処理を行なう必要があるでしょう。See Section 10.19.8.1 [Creating a spam-stat dictionary], p. 315.

`spam-use-stat` [変数]

`gnus-group-spam-exit-processor-stat` [変数]
このシンボルを、グループパラメーターのカスタマイズによってグループの`spam-process` パラメーターに加えるか、または`gnus-spam-process-newsgroups` 変数に加えてください。このシンボルがグループの`spam-process` パラメーターに加えられると、spam 印が付いた記事が spam 記事用の spam-stat データベースに追加されます。

警告

旧式の`gnus-group-spam-exit-processor-stat` の代わりに`(spam spam-use-stat)` を使うことを推奨します。すべて同等に動作することは保証されます。

`gnus-group-ham-exit-processor-stat` [変数]
このシンボルを、グループパラメーターのカスタマイズによってグループの`spam-process` パラメーターに加えるか、または`gnus-spam-process-newsgroups` 変数に加えてください。このシンボルがグループの`spam-process` パラメーターに加えられ

ると、ham 印が付いた*ham* グループの記事が非-spam 記事用の spam-stat データベースに追加されます。

警告

旧式の`gnus-group-ham-exit-processor-stat` の代わりに`(ham spam-use-stat)` を使うことを推奨します。すべて同等に動作することは保証されます。

これは`spam.el` が`spam-stat.el` と働き合うことを可能にします。`spam-stat.el` は(Lisp だけの) spam 内部データベースを提供しますが、それは ifile や Bogofilter と違って外部プログラムを必要としません。Spam と ham のプロセッサー、および`spam-split` のための`spam-use-stat` 変数が提供されます。

10.19.6.10 Gnus で SpamOracle を使うには

気軽に spam を濾過する一つのやり方は SpamOracle を使うことです。統計的にメールを濾過するための道具である SpamOracle は、Xavier Leroy によって書かれました。これは別にインストールする必要があります。

Gnus で SpamOracle を使うには、複数のやり方があります。すべての場合に、メールは*mark* モードで動作している SpamOracle にパイプされます。すると SpamOracle は、そのメールを spam だと見なしたかどうかを示す‘`X-Spam`’ ヘッダーを記入します。

実現可能な一つは、SpamOracle を`:prescript` としてSection 7.4.4.1 [Mail Source Specifiers], p. 166, から走らせることです。この方法には、利用者が*X-Spam* ヘッダーを見ることができるという利点があります。

もっとも手軽な方法は、`spam.el` (see Section 10.19 [Spam Package], p. 295) が SpamOracle を呼ぶようにすることです。

`spam.el` で SpamOracle を利用できるようにするためには、変数`spam-use-spamoracle` を`t` にして、`nnmail-split-fancy` または`nnimap-split-fancy` を設定してください。See Section 10.19 [Spam Package], p. 295. この例では nnimap サーバーの‘`INBOX`’ が SpamOracle を使って濾過されます。Spam であると認定されたメールは、`spam-split-group` (この場合は‘`Junk`’) に移動させられます。Ham なメッセージは‘`INBOX`’ に残ります:

```
(setq spam-use-spamoracle t
      spam-split-group "Junk"
      ;; nnimap 用にたぶん nnimap-split-methods を設定する必要あり。
      ;; マニュアル参照。
      nnimap-split-inbox '("INBOX")
      nnimap-split-fancy '(| (: spam-split) "INBOX"))
```

`spam-use-spamoracle` [変数]
Gnus に SpamOracle を使って spam の濾過をさせたい場合に`t` にしてください。

`spam-spamoracle-binary` [変数]
Gnus は利用者の PATH で見つかった`spamoracle` という SpamOracle のバイナリーを使います。これにはカスタマイズ可能な変数`spam-spamoracle-binary` を使います。

`spam-spamoracle-database` [変数]
SpamOracle はその解析結果をデータベースとして格納するために、ディフォルトで`~/.spamoracle.db` ファイルを使います。これは変数`spam-spamoracle-database`

で制御され、ディフォルトは`nil` です。それは、ディフォルトの SpamOracle データベースが使われることを意味します。データベースがどこか特別な場所に置きたい場合は、`spam-spamoracle-database` をそのパスに設定してください。

SpamOracle はメッセージが spam か ham かを見極めるために統計的な手法を使います。間違いや見逃しの少ない良い結果を得るために、SpamOracle はトレーニングを必要とします。SpamOracle は spam メールの特徴を学びます。*add* モード(トレーニング・モード) を使って、良いメール(ham) と spam を SpamOracle に与えなければなりません。これは、概略バッファーで| を押すことによってメールを SpamOracle のプロセスにパイプするか、または`spam.el` の spam および ham プロセッサーを使うことによって行なうことができます。See Section 10.19 [Spam Package], p. 295.

`gnus-group-spam-exit-processor-spamoracle` [変数]
このシンボルを、グループパラメーターのカスタマイズによってグループの`spam-process` パラメーターに加えるか、または`gnus-spam-process-newsgroups` 変数に加えてください。このシンボルがグループの`spam-process` パラメーターに加えられると、spam 印が付いた記事が spam のサンプルとして SpamOracle に送られます。

警告

旧式の`gnus-group-spam-exit-processor-spamoracle` の代わりに`(spam spam-use-spamoracle)` を使うことを推奨します。すべて同等に動作することは保証されます。

`gnus-group-ham-exit-processor-spamoracle` [変数]
このシンボルを、グループパラメーターのカスタマイズによってグループの`spam-process` パラメーターに加えるか、または`gnus-spam-process-newsgroups` 変数に加えてください。このシンボルがグループの`spam-process` パラメーターに加えられると、*ham* グループにある ham 印が付いた記事が ham の記事のサンプルとして SpamOracle に送られます。

警告

旧式の`gnus-group-ham-exit-processor-spamoracle` の代わりに`(ham spam-use-spamoracle)` を使うことを推奨します。すべて同等に動作することは保証されます。

例: これらは ham グループとして分類された、つまり ham の記事しかないグループのためのグループパラメーターです。

```
((spam-contents gnus-group-spam-classification-ham)
 (spam-process ((ham spam-use-spamoracle)
                (spam spam-use-spamoracle))))
```

このグループでは`spam-use-spamoracle` が ham と spam の両方の処理を行ないます。このグループに spam 記事があって(SpamOracle が十分なサンプルを食べさせてもらっていなければ、そうなりますね)、かつ利用者がいくつかの記事に spam の印を付けたならば、それらの記事は SpamOracle によって処理されます。そのプロセッサーは、新しい spam のサンプルとして SpamOracle に記事を送ります。

10.19.7 Spam パッケージの拡張

Blackbox という新しいバックエンドを追加したいとしましょう。入ってくるメールを濾過するために以下のものを用意してください:

1. コード

```
(defvar spam-use-blackbox nil
  "Blackbox を使うときは t にする。")
```

Blackbox が入ってくるメールを検査できるのであれば、spam-check-blackbox を書いてください。

Blackbox が spam と ham を登録または登録解除できるのであれば、手始めに bogofilter の登録/解除ルーチンを使って、またはもっと Blackbox にふさわしい他の登録/解除ルーチンを使ってspam-blackbox-register-routine およびspam-blackbox-unregister-routine を書いてください。

2. 関数

spam-check-blackbox 関数は、他の習慣に倣って‘nil’ かspam-split-group を返さなければなりません。あなたに何ができるかの例として、既存のspam-check-* 関数を参照してください。また、あなたがそうでない理由を完全に理解していないならば、テンプレートに齧り付いてください。

Spam と ham メッセージを処理するために、以下のものを用意してください:

1. コード

Spam または ham のプロセッサーを用意する必要はありません。Blackbox が spam または ham の処理をサポートする場合だけ、それらを用意してください。

さらに ham と spam のプロセッサーは単一の変数ではなくされつつあり、代わりに(spam spam-use-blackbox) または(ham spam-use-blackbox) の形式が推奨されます。今のところ spam/ham プロセッサー変数はまだあちこちにありますが、長く存続することはないでしょう。

```
(defvar gnus-group-spam-exit-processor-blackbox "blackbox-spam"
  "概略を出るときに呼ばれる blackbox の spam プロセッサー。
Spam グループだけに適用される。")

(defvar gnus-group-ham-exit-processor-blackbox "blackbox-ham"
  "概略を出るときに呼ばれる blackbox の ham プロセッサー。
Spam ではない（未分類または ham）グループだけに適用される。")
```

2. Gnus のパラメーター

gnus.el にあるグループパラメーターspam-process に

```
(const :tag "Spam: Blackbox" (spam spam-use-blackbox))
(const :tag "Ham: Blackbox"  (ham spam-use-blackbox))
```

を加えてください。それを必ずパラメーターと変数のカスタマイズの二回について行なうようにしてください。

Blackbox が入ってくるメールが spam かどうかを検査できるのであれば、

```
(variable-item spam-use-blackbox)
```

をgnus.el のグループパラメーターspam-autodetect-methods に加えてください。

最後に、spam.el にある適切なspam-install-*-backend 関数を使ってください。利用できる関数は次の通りです。

1. spam-install-backend-alias

この関数は、オリジナルのバックエンドのようにすべてを行なうバックエンドのために、別名を使うことができるようにするだけです。今のところ、これ

はspam-use-BBDB-exclusive をspam-use-BBDB のように働かせるためだけに使われます。

2. spam-install-nocheck-backend

   この関数は、検査する機能は無いけれども ham または spam を登録/解除することができるバックエンドになります。spam-use-gmane がそのようなバックエンドです。

3. spam-install-checkonly-backend

   この関数は、入ってくるメールが spam かどうかの検査だけを行なうことができるバックエンドになります。それはメッセージを登録または登録解除することができません。spam-use-blackholes はそのようなバックエンドです。

4. spam-install-statistical-checkonly-backend

   この関数は、入ってくるメールの検査だけを行なうことができる、統計処理を行なうバックエンド(検査のためにメッセージの本文全体を必要とします) になります。spam-use-regex-body がそのような濾過器です。

5. spam-install-statistical-backend

   この関数は、入ってくるメールの検査と登録/解除ルーチンを持つ、統計処理を行なうバックエンドになります。spam-use-bogofilter は、そのように仕立てられています。

6. spam-install-backend

   これは最も普通なバックエンドになります。それは検査とメッセージの登録/解除を行なうことができ、統計処理の能力はありません。spam-use-BBDB がそのようなバックエンドです。

7. spam-install-mover-backend

   移動させる(mover) バックエンドはspam.el の内部で、概略バッファーを出るときにある明確なやり方で記事を移動させます。おそらくそのようなバックエンドを使うことは無いでしょう。

10.19.8 Spam 統計パッケージ

Paul Graham は統計を使った spam の濾過に関する優れたエッセイを書きました: A Plan for Spam (https://www.paulgraham.com/spam.html)。そこで彼は SpamAssassin によって使われているような規則ベースの濾過に固有な欠陥について述べています。例えば: 誰かが規則を書かなければならないし、他のすべての人はこれらの規則をインストールしなければなりません。あなたはいつも遅れをとってしまいます。それよりも、それが spam または非-spam に何となく似ているかどうかに基づいてメールを濾過する方が遥かに良いだろうと彼は主張しています。これを測定する一つの手段は単語の分布です。その後で彼は、新着メールがあなたの他の spam メールに似ているかどうかをチェックする方法を述べています。

基本的な考えはこうです: あなたのメールの二つの集合を作ります。一方は spam を、もう一方は spam ではないメールを集めたものです。両方の集合における各単語の出現頻度を数えて、集合のメールの総数で重み付けを行ない、この情報を辞書に格納します。新着メールのすべての単語について、spam か spam ではないメールに属する確率を判定します。15 の最も顕著な単語を使って、そのメールが spam であることの確率の総計を計算します。この確率がある閾値より高かったら、そのメールは spam であると見なされます。

Spam 統計パッケージは、この種の濾過のためのサポートを Gnus に追加します。Spam パッケージ(see Section 10.19 [Spam Package], p. 295) のバックエンドの一つとして、またはそれ自体を使うことができます。

Spam 統計パッケージを使う前に、それを使うための準備を行なう必要があります。第一に、あなたのメールの二つの集合が要ります。一方は spam を、もう一方は spam ではないメールを集めたものです。そして、それらの二つの集合を使って辞書を作り、それをセーブしてください。そして特に最後に、あなたの特級分割の規則でこの辞書を使ってください。

10.19.8.1 spam-統計(spam-stat) 辞書を作る

統計に基づいた spam 濾過を始めるには、前もって二つのメールの集合(一方は spam を、もう一方は spam ではないメールを集めたもの) に基づいた統計を作らなければなりません。そしてそれらの統計は、後で利用するために辞書に格納されます。それらの統計を意味のあるものにするために、両方の集合につき数百通のメールが必要です。

今のところ Gnus は nnml バックエンドでだけ辞書の自動生成をサポートします。nnml バックエンドは一通のメールにつき一つのファイルで、すべてのメールを一つのディレクトリーに格納します。以下を使ってください:

`spam-stat-process-spam-directory` [関数]
: このディレクトリーにあるすべてのファイルについて spam の統計を生成します。すべてのファイルは一つの spam メールとして扱われます。

`spam-stat-process-non-spam-directory` [関数]
: このディレクトリーにあるすべてのファイルについて非-spam の統計を生成します。すべてのファイルは一つの spam ではないメールとして扱われます。

`spam-stat-process-directory-age` [変数]
: 処理されるファイルの最大経過日数。このフィルターがないと、数千のメッセージを含む spam-stat の再トレーニングに長い時間がかかる可能性があります。ディフォルトは 90 ですが、最初のトレーニングではより大きな値に設定する方が良いでしょう。

普通は`~/Mail/mail/spam` のようなディレクトリー(通常‘`nnml:mail.spam`’ グループに対応) について`spam-stat-process-spam-directory` を呼ぶことになるでしょう。また`~/Mail/mail/misc` のようなディレクトリー(通常‘`nnml:mail.misc`’ グループに対応) について`spam-stat-process-non-spam-directory` を呼びましょう。

IMAP を使っている場合はメールをローカルには持っていないので、それは働きません。一つの解決策は、Gnus エージェントで記事をキャッシュすることです。そうすれば`spam-stat-process-spam-directory` として`"~/News/agent/nnimap/mail.yourisp.com/personal_spam"` のようなものを使うことができます。See Section 7.9.5 [Agent as Cache], p. 226.

`spam-stat` [変数]
: この変数はすべての統計のハッシュテーブル---辞書と言っているもの---を保持します。このハッシュテーブルは、双方の集合のすべての単語について spam および spam ではないメールにおける出現頻度を表すベクトルを格納します。

統計を最初から作り直したいときは、辞書をリセットする必要があります。

`spam-stat-reset` [関数]
: すべての統計を削除して`spam-stat` ハッシュテーブルをリセットします。

行なったら辞書をセーブしなければなりません。辞書はかなり大きくなるかもしれません。辞書を追加更新しない場合(言い換えると、例えば毎月一回作り直す場合)、頻繁に現れないか、または spam か spam ではないメールのどちらに属するかがはっきりしないすべての単語を削除することによって、辞書のサイズを小さくすることができます。

`spam-stat-reduce-size` [関数]
辞書のサイズを小さくします。これは辞書を追加更新したくない場合だけ使ってください。

`spam-stat-save` [関数]
辞書をセーブします。

`spam-stat-file` [変数]
辞書の格納に使うファイル名です。ディフォルトは`~/.spam-stat.el` です。

10.19.8.2 spam-統計(spam-stat) を使ってメールを分割する

この章は Spam 統計パッケージを Spam パッケージ(see Section 10.19 [Spam Package], p. 295) とは独立して 使う方法について説明します。

最初に以下を`~/.gnus.el` ファイルに追加してください:

```
(require 'spam-stat)
(spam-stat-load)
```

これは必要な Gnus のコードとあなたが作った辞書を読み込みます。

次に、特級分割の規則を改造する必要があります: どうやって`spam-stat` を使うかを決めてください。以下の例は nnml バックエンド用です。nnimap バックエンドでもまったく同様に動作します。単に`nnmail-split-fancy` の代わりに`nnimap-split-fancy` を使ってください。

‘`mail.misc`’ と‘`mail.spam`’ の二つのグループだけがある、最も単純な事例を想定しましょう。以下の式は、メールが spam でなかったら‘`mail.misc`’ に行くことを示します。もし spam だったら、`spam-stat-split-fancy` は‘`mail.spam`’ を返します。

```
(setq nnmail-split-fancy
      `(| (: spam-stat-split-fancy)
          "mail.misc"))
```

`spam-stat-split-fancy-spam-group` [変数]
Spam 用のグループです。ディフォルトは‘`mail.spam`’ です。

特定の表題を持つメールを他のグループに入れる濾過をも行ないたいならば、以下の式を使ってください。正規表現に合致しないメールだけに spam の可能性があると考えます。

```
(setq nnmail-split-fancy
      `(| ("Subject" "\\bspam-stat\\b" "mail.emacs")
          (: spam-stat-split-fancy)
          "mail.misc"))
```

最初に spam の濾過をしたい場合、辞書を作るときに十分に注意しなければなりません。`spam-stat-split-fancy` は‘`mail.emacs`’ と‘`mail.misc`’ のどちらのメールも spam ではないと解釈しなければならないので、辞書を作るときに spam ではない集合に両方とも入っていなければならないことに注意してください。

```
(setq nnmail-split-fancy
```

```
        `(| (: spam-stat-split-fancy)
            ("Subject" "\\bspam-stat\\b" "mail.emacs")
            "mail.misc"))
```

これを伝統的な濾過と組み合わせることもできます。ここではすべての HTML だけのメールを‘mail.spam.filtered’ グループに入れるものとしましょう。spam-stat-split-fancy はそれらのメールを見ないので、辞書を作るときに、‘mail.spam.filtered’ のメールが spam の集合または spam ではない集合のどちらにも入るべきではないことに注意してください!

```
(setq nnmail-split-fancy
      `(| ("Content-Type" "text/html" "mail.spam.filtered")
          (: spam-stat-split-fancy)
          ("Subject" "\\bspam-stat\\b" "mail.emacs")
          "mail.misc"))
```

10.19.8.3 spam-統計(spam-stat) 辞書への低階層インターフェース

spam-stat を使うための主なインターフェースは以下の関数群です:

spam-stat-buffer-is-spam [関数]
: Spam であると考えられる新着メールがあるバッファーで呼ばれます。まだ処理されていない新着メールに対して使ってください。

spam-stat-buffer-is-no-spam [関数]
: Spam ではないと考えられる新着メールがあるバッファーで呼ばれます。まだ処理されていない新着メールに対して使ってください。

spam-stat-buffer-change-to-spam [関数]
: それが spam ではなくて通常のものだとはもはや考えられないメールがあるバッファーで呼ばれます。すでに spam ではないものとして処理されてしまったメールの地位の変更に使ってください。

spam-stat-buffer-change-to-non-spam [関数]
: それが通常のものではなくて spam だとはもはや考えられないメールがあるバッファーで呼ばれます。すでに spam であるものとして処理されてしまったメールの地位の変更に使ってください。

spam-stat-save [関数]
: ハッシュテーブルをファイルにセーブします。変数spam-stat-file で設定されたファイル名が使われます。

spam-stat-load [関数]
: ハッシュテーブルをファイルから読み込みます。変数spam-stat-file で設定されたファイル名が使われます。

spam-stat-score-word [関数]
: 単語の spam スコアを返します。

spam-stat-score-buffer [関数]
: バッファーの spam スコアを返します。

`spam-stat-split-fancy` [関数]
: 特級メール分割のためにこの関数を使ってください。`nnmail-split-fancy` に規則‘`(: spam-stat-split-fancy)`’ を追加しましょう。

それを使う前に、必ず辞書が読み込まれているようにしてください。これには`~/.gnus.el` ファイルに以下が必要です:

```
(require 'spam-stat)
(spam-stat-load)
```

以下は一般的なテストのための関数呼び出しです:

```
リセット: (setq spam-stat (make-hash-table :test 'equal))
Spam の学習: (spam-stat-process-spam-directory "~/Mail/mail/spam")
非-spam の学習: (spam-stat-process-non-spam-directory "~/Mail/mail/misc")
辞書をセーブ: (spam-stat-save)
辞書ファイルのサイズ: (file-attribute-size (file-attributes spam-stat-file))
単語数: (hash-table-count spam-stat)
Spam の検査: (spam-stat-test-directory "~/Mail/mail/spam")
非-spam の検査: (spam-stat-test-directory "~/Mail/mail/misc")
辞書のサイズを小さくする: (spam-stat-reduce-size)
辞書をセーブ: (spam-stat-save)
辞書ファイルのサイズ: (file-attribute-size (file-attributes spam-stat-file))
単語数: (hash-table-count spam-stat)
Spam の検査: (spam-stat-test-directory "~/Mail/mail/spam")
非-spam の検査: (spam-stat-test-directory "~/Mail/mail/misc")
```

以下は辞書を生成する方法です:

```
リセット: (setq spam-stat (make-hash-table :test 'equal))
Spam の学習: (spam-stat-process-spam-directory "~/Mail/mail/spam")
非-spam の学習: (spam-stat-process-non-spam-directory "~/Mail/mail/misc")
別の必要な非-spam グループに対して繰り返し...
辞書のサイズを小さくする: (spam-stat-reduce-size)
辞書をセーブ: (spam-stat-save)
```

10.20 Gnus レジストリー

Gnus レジストリーは記事の Message-ID を元にすべてのバックエンドを横断的に追跡するためのパッケージです。これが提供するいくつもある素敵な事をすれば、Gnus ユーザーは近所の人たちから羨ましがられ、床屋がただで散髪してくれるようになり、世界で起きる様々な問題の専門家になることができるでしょう。かなり素晴らしい機能が満載です。うーむ、全部がそうではないかもしれませんけれどね。

じきに詳しく説明しますが、まずはそれらの機能の早見表を掲げておきましょう。あなたの注意力が持続しないかもしれないので... いや、何でもないです。

1. 記事をそれらの親に分割する

   議論を同じグループで続けられるようにする機能です。Message-ID に加えて表題と送信者を使うことができます。これにはいくつかのやり方があります。

2. ID に基づいてメッセージを照会する

   gnus-summary-refer-parent-article のようなコマンドは、照会する記事に行くするために、そのメッセージがあるグループにかかわらず、レジストリーを利用することができます。

3. 独自のフラグとキーワードを記録する

   レジストリーは記事のための独自のフラグとキーワードを記録することができます。これで例えば記事に“To-Do” の印を付けることができ、記事が nnimap、nnml、nnmaildir などのどのバックエンドにあっても、そのフラグは立ち続けます。

4. 任意のデータを記録する

   レジストリーは記事のためのどんなデータでも、簡単な Lisp インターフェースを通して記憶しておくことができます。組み込まれている逆引き機能を有効にしておくと、特定の条件に合うすべての記事をすばやく見つけることができます。

10.20.1 Gnus レジストリーの設定

良くしたことに Gnus レジストリーの設定はとても簡単です:

```
(setq gnus-registry-max-entries 2500)

(gnus-registry-initialize)
```

これは Gnus が newsrc をセーブするときの処理にレジストリーをセーブする処理を加えます(それは Gnus を終了するときと*Group* バッファーで*s* を押したときに発動します)。さらにこれは Gnus の記事に関する動作(コピー、移動など) にレジストリーを操作する機能を加えるので、この初期化による結果を元に戻すのは簡単ではありません。その、あまり愉快ではない詳細については gnus-registry-initialize を見てください。

以下はレジストリーの持ち主が使うための他の設定です(それらを無頓着にコピーする前に、それらが何をするかを理解してください)。

```
(setq
 gnus-registry-split-strategy 'majority
 gnus-registry-ignored-groups '(("nntp" t)
                                ("nnrss" t)
                                ("spam" t)
                                ("train" t))
 gnus-registry-max-entries 500000
 ;; これがディフォルト
 gnus-registry-track-extra '(sender subject))
```

これらが意味するのは、あちこちにあるたくさんの記事を保持し、送信者と表題で(単なる親の Message-ID でではなく) 記事を追跡し、入ってきた記事をレジストリーが分割するときに記事の行き先として一つ以上の候補があったら多数決で決める、ということです。加えて“nntp”、“nnrss”、“spam” または“train” に合致するグループの記事をレジストリーに無視させます。

あなたがこのすべてに感銘を受けることは疑いありませんが、こう要求しもするでしょう。「私は Gnus ユーザーでカスタマイズすることが生きがいです。もっと下さい」。では諸設定の一般的な説明に参りましょう。

`gnus-registry-unfollowed-groups` [変数]
`gnus-registry-split-fancy-with-parent` が追跡しないグループです。それらは、でもレジストリーに記録されます。これは正規表現のリストです。ディフォルトでは、名前が“delayed”、“drafts”、“queue” または“INBOX” で終わっている、`nnmairix` バックエンドに属している、あるいは“archive” という語を含んでいるどんなグループも追跡しません。

`gnus-registry-max-entries` [変数]
レジストリーが保持する項目の数(整数または無制限を意味する`nil`) です。レジストリーがこのサイズに達してしまう、または越えてしまうと、新しいエントリーの追加を受け付けなくなります。

`gnus-registry-register-all` [変数]
もしこのオプションが`nil` でなかったら、レジストリーはすべてのメッセージを表示されている通りに登録します。これは、レジストリーはすべてのメッセージがどこにあるかを知っている必要があるので、親への分割と Message-ID の参照を正しく機能させるために重要ですが、Gnus が行なうグループを開くことと保存が遅くなる可能性があります。もしこのオプションが`nil` である場合、エントリーは手動で作成、つまり例えばメッセージのカスタムフラグまたはキーワードを保存しなければなりません。

`gnus-registry-prune-factor` [変数]
除去に際してレジストリーをどのくらい減らすかを、このオプション(0 から 1 までの浮動小数) で制御します。一定数のエントリーを削除する代わりに、レジストリーの数が`gnus-registry-max-entries` より小さくなるように除去を行ないます。このオプションは厳密にどのくらい小さくするかを制御し、目標値はエントリーの最大数× (1 - この係数) になります。ディフォルトは 0.1 です。つまり、もしあなたのレジストリーが 50000 エントリーを上限としているなら、除去の際に 45000 エントリーに削減しようとします。貴重(precious) であると記されたキーを持っているエントリーは除去されません。

`gnus-registry-default-sort-function` [変数]
このオプションは除去に際してレジストリーをどのようにソートするかを指示します。関数を与える場合、それはリストの先頭から除去を始めて、最も価値の低いエントリーを先頭に置くものでなければなりません。ディフォルト値は最も古いエントリーから除去する`gnus-registry-sort-by-creation-time` です。`nil` にするとソートを行なわず、除去の処理速度が上がります。

`gnus-registry-cache-file` [変数]
Gnus の操作を通じてレジストリーが記録されるファイルです。ディフォルトでファイル名は`.newsrc.eld` と同じディレクトリーに置かれる`.gnus.registry.eieio` です。

10.20.2 Message-ID に基づいてレジストリーで記事を取得する

レジストリーは、それぞれの`Message-ID` の記事が存在するグループを知っています。これは「記事を参照する方法」すなわち「ある Message-ID の記事を参照する方法を Gnus に指示するもの」を増強するためにてこ入れすることができます。See Section 4.23 [Finding the Parent], p. 109.

nnregistry 参照方法はまさにそれをします。それには、記事がそれがあるグループにかかわらず見つかるかもしれないという特長があります---もしそのMessage-ID をレジストリーが知っていれば。それは、以下の方針に沿って、あるもので起動ファイルを増大することにより可能になるはずです:

```
;; レジストリーを使って記事を参照する場合に良好なヒット率を得るため
;; には、十分なエントリーを維持するようにしてください。記事がどこに
;; あるかを Gnus に知らせるために、長いグループ名を使用してください。
(setq gnus-registry-max-entries 2500)

(gnus-registry-initialize)

(setq gnus-refer-article-method
      '(current
        (nnregistry)
        (nnweb "google" (nnweb-type google))))
```

上記の例は、最初は現在のグループの中で、あるいはレジストリーを使って、そしてそれらすべてが失敗したら Google を使って記事を探すことを Gnus に指示します。

10.20.3 親への特級分割

簡単に言うと、これでフォローアップの電子メールを、それがあるべき場所に置くことができます。

すべての記事は固有の Message-ID を持っていて、レジストリーはそれを記憶します。記事を移動またはコピーするとき、レジストリーはそのことに気付いて、分割方法のひとつの選択肢として新しいグループを提案します。

フォローアップするとき、言及する対象となる元の記事の Message-ID は通常ヘッダーにあります。レジストリーはそれを知り、その記載内容を使って元の記事がどこにあるかを探します。あなたが特級分割の設定に置いておく必要があるのは、このような規則だけです:

```
(setq nnimap-my-split-fancy '(|

      ;; split to parent: you need this
      (: gnus-registry-split-fancy-with-parent)

      ;; other rules, as an example
      (: spam-split)
      ;; default mailbox
      "mail")
```

gnus-registry-register-all がnil でない(ディフォルト) の場合、レジストリーはすべてのメッセージに対して分割を実行します。nil だったら分割は明示的に登録したメッセージの子に対してのみ発生します。

さらに、以下の変数をカスタマイズする必要があるでしょう。

gnus-registry-track-extra [変数]

これはシンボルのリストです。カスタマイズのインターフェースから変更するのには、それがベストです。ディフォルトは(subject sender recipient) で、それでおそらく動作するでしょう。メールの流通量が大きくて人々が同じグループにとどまっていないと、煩わしくなる場合があります。

それら枠外のデータの追跡をやめるときは、コマンドgnus-registry-remove-extra-data を使って、既存のレジストリーから取り除いてください。

gnus-registry-split-strategy [変数]
これはシンボルです。カスタマイズのインターフェースから変更するのには、それがベストです。ディフォルトはnil ですが、合致の多数決または最初に見つかったものに基づいて送信者(sender) または表題(subject) で分割するにはmajority またはfirst に設定する必要があるでしょう。私はmajority が最も良いことを見つけました。

10.20.4 独自のフラグとキーワードの記録

レジストリーを使って、独自のフラグとキーワードをメッセージごとに設定することができます。Gnus->Registry Marks メニューか*M M x* ショートカット・キーを使ってください。ここでx は印の名前の最初の文字です。

gnus-registry-marks [変数]
レジストリーが扱うことができる独自の印のリストです。もしそうしたいなら、ディフォルトのリストを変更することができます。それを行なうのならば、それらが効果を及ぼす前に Emacs を終了させる必要があります(レジストリーを抜き取って(unload して) から再読み込み(reload) するか、または必要であろう特別なマクロを実行することもできますが、たぶんそんな面倒なことは望まないでしょう)。カスタマイズのインターフェースを使って、そのリストを変更してください。

ディフォルトでこのリストにはImportant、Work、Personal、To-Do それにLater の印があります。それらすべてに、Important のための*M M i* のように、最初の文字を使うショートカット・キーが用意されています。

gnus-registry-set-article-mark [関数]
独自のレジストリーの印を記事に付加するために使う関数です。補完のために、利用できる印を提示します。

概略行にレジストリーの印を表示できる書法を作る関数を、defalias を使って設定することができます。このための関数として、:char プロパティーを使って単一の文字で印を表示するもの、および完全な文で印を表示するものの 2 つが用意されています。

```
;; 単一の文字で印を表示する
;; ('gnus-registry-marks' の :char プロパティーを参照):
;; (defalias 'gnus-user-format-function-M
;;   'gnus-registry-article-marks-to-chars)

;; 名前で印を表示する ('gnus-registry-marks' を参照):
;; (defalias 'gnus-user-format-function-M
;;   'gnus-registry-article-marks-to-names)
```

10.20.5 任意のデータの記録

レジストリーには任意のデータを記録するために、Message-ID をキーとして使う簡単なインターフェースがあります(データは保存するためにできる限り一つのリストに変換されます)。

gnus-registry-set-id-key (*id key value*) [関数]
id を持つメッセージのためにkey についてvalue を格納します。

`gnus-registry-get-id-key` (*id key*) [関数]
: `id` を持つメッセージのために`key` についてのデータを取得します。もしオプション`gnus-registry-register-all` が`nil` でなかったら、この関数は`id` のエントリーが存在しない場合は、それも作成します。

`gnus-registry-extra-entries-precious` [変数]
: 特別なエントリーが貴重(precious) であるなら、たとえその Message-ID の記事のグループが無くても、またレジストリーのサイズの制限に達しても、それらが存在しているとレジストリーはすべてのエントリーを永久に保持します。ディフォルトではこれはまさに`(mark)` なので、独自のレジストリーの印は貴重であると見なされます。

10.21 Gnus クラウド

Gnus クラウドは印や雑多なファイルとデータを複数のマシン間で同期するための一つの手段です。

すべての印(どの記事を読んだか、どれが重要か、など) を数台のマシンの間で同期させたいという需要が、非常にたびたび生じます。IMAP ではプロトコルに組み込まれているので、多くのマシンから nnimap グループを読むことができるし、それらは自動的に同期されます。しかし NNTP、nnrss、および他の多くのバックエンドは印を蓄えないので、それはローカルに行なわなければなりません。

Gnus クラウドは印とあなたが選んだファイルを IMAP サーバーの特別なフォルダーに格納します。それは DropTorrentSyncBoxOakTree(TM) のようなものです。[1]

10.21.1 Gnus クラウドの設定

Gnus クラウドの設定には 1 分もかかりません。グループバッファーから:

サーバーバッファーに行くために`^` を押してください。そこでは Gnus が知っているすべてのサーバーが見えるでしょう。See Section 7.1 [Server Buffer], p. 146.

そうしたらクラウドで同期させたいどんなサーバーでも、*i* を押して指定してください(それらの印が同期されます)。

そしてクラウドのホストになる単一のサーバーを*I* を押して指定してください(それは IMAP サーバーでなければなりません。すべての同期データを持つフォルダーを提供することになります)。これは(カスタマイズの手段を使って) 変数`gnus-cloud-method` を設定し、初めての CloudSynchronizationDataPack(TM) をアップロードすることを促します。

10.21.2 Gnus クラウドの使い方

設定した後は、グループバッファーからこれらのショートカットを使うことができます:

`~ RET`

`~ d`
: 最新の Gnus クラウドのデータをダウンロードします。

`~ u`

`~ ~`
: ローカルな Gnus クラウドのデータをアップロードします。新しい CloudSynchronizationDataPack(TM) を作成します。

[1] 「Gnus クラウド」という名前はパロディーです。クラウド・コンピューティング、すなわち`https://www.gnu.org/philosophy/words-to-avoid.html#CloudComputing` (通常は使わない方が良い誤解を招く用語) とはほとんど関係がありません。

しかし待ってください。もっとあります。もちろん、もっと。たくさん。以下のすべてをカスタマイズできます。

`gnus-cloud-synced-files` [変数]
すべての CloudSynchronizationDataPack(TM) の部分になるファイル群があります。それらはアップロードするたびに付いて行くので、大きなファイルをたくさん同期させるのはやめましょう。100Kb 未満がベストです。

`gnus-cloud-storage-method` [変数]
いくつかの格納方法から選ぶためのものです。EPG の機能を使うことを強く推奨します。もし GnuPG がインストールされていて EPG が読み込まれていれば自動で適用されます。でも Base64+gzip か Base64 を使ったり、またはエンコードしないことも可能です。

`gnus-cloud-interactive` [変数]
これがセットされていると(ディフォルトでセット)、Gnus クラウドパッケージはいろんな場面であなたに確認を求めます。不満に感じることがなければ、そのままにしておいてください。

`gnus-cloud-method` [変数]
CloudSynchronizationDataPack(TM) を格納する IMAP サーバーの名前です。サーバーバッファーから設定するのが最も簡単です(see Section 10.21.1 [Gnus Cloud Setup], p. 323)。

10.22 システムスリープの統合

ラップトップ、あるいはスリープまたは休止機能があるシステムを使っていると、長時間実行しているサーバー接続が「ハング」してしまい、システムが再開した後でユーザーが手動で接続をいったん閉じて開き直す必要が生じることがあります。システムスリープライブラリーを使うと、Gnus はシステムがスリープになる前にすべてのサーバー接続を自動的に閉じることができます。これを有効にするには`gnus-close-on-sleep` を`nil` 以外の値にしてください。

10.23 他のモードとの相互作用

10.23.1 Dired

`gnus-dired-minor-mode` は dired バッファーで使えるいくつかの便利な機能を提供します。これは次の式で有効になります:

```
(add-hook 'dired-mode-hook 'turn-on-gnus-dired-mode)
```

C-c C-m C-a
Dired で印を付けたものを添付ファイルとして送信します(`gnus-dired-attach`)。この関数はどの message バッファーに添付するかを尋ねて、ディフォルトではそのバッファーの末尾にファイルを添付します。代わりに現在の位置に添付させたいなら`gnus-dired-attach-at-end` をカスタマイズしてください。

C-c C-m C-l
適切な mailcap 項目に従ってファイルを開きます(`gnus-dired-find-file-mailcap`)。接頭引数を付けると、ファイルを新しいバッファーで(単に) 開きます。

C-c C-m C-p
: mailcap 項目に従ってファイルを印刷します(gnus-dired-print)。印刷コマンドが無い場合は PostScript 画像として印刷します。

10.24 いろいろのいろいろ

gnus-home-directory
: すべての Gnus のファイル名とディレクトリー名の変数は、これを基点にして初期値が決定されます。ディフォルトは~/ です。

gnus-directory
: Gnus がデータを格納するほとんどのファイルとディレクトリーの名前の変数は、これを基点にして初期値が決定されます。ディフォルトはSAVEDIR 環境変数の値か、その変数が設定されていない場合は~/News/ です。

  ~/.gnus.el ファイルが読まれたときは Gnus のほとんどはすでに読み込まれているということに注意してください。これはつまり、この変数を~/.gnus.el の中で設定しても、この変数によって初期化される他のディレクトリー変数は正しく設定されないだろうということです。この変数は代わりに.emacs で設定してください。

gnus-default-directory
: 上記の変数にはまったく関係ありません---この変数はすべての Gnus バッファーのディフォルトディレクトリーをどうすべきかを設定します。もし*C-x C-f* のような命令を実行すると、現在のバッファーのディフォルトディレクトリーを起点にしたプロンプトが出てくるでしょう。この変数がnil (これがディフォルト) であれば、Gnus を起動したときにあなたがいたバッファーのディフォルトディレクトリーがディフォルトディレクトリーになるでしょう。

gnus-verbose
: この変数は 0 から 10 までの間の整数です。数値が大きいほどたくさんのメッセージが表示されます。この変数が 0 であれば Gnus は何のメッセージも見せません。これが 7 (ディフォルト) であれば特に重要なメッセージが表示され、10 であれば Gnus は決してお喋りを止めず、たくさんのメッセージであなたにめまいを起こさせるでしょう。

gnus-verbose-backends
: この変数はgnus-verbose と同様の効果をもたらしますが、Gnus 本体ではなく Gnus のバックエンドに対して適用されます。

gnus-add-timestamp-to-message
: この変数はgnus-verbose とgnus-verbose-backends で制御されて出力されるメッセージに、時刻(タイムスタンプ) を付加するかどうかを制御します。ディフォルト値は時刻を付加しないことを意味するnil です。log だったら*Messages* バッファーに入るメッセージだけに時刻を付加します。もしnil でもlog でもなければ、ログに入るメッセージだけでなく、エコーエリアに表示されるものにも時刻を付加します。

nnheader-max-head-length
: バックエンドが記事の連続したヘッダー部を読んでいるときは、できる限り少ない量だけを読もうとします。この変数(ディフォルト 8192) は、バックエンド

がヘッダーと本文の間の区切り行の捜索を諦める前に読み込む絶対最大長を指定します。この変数が`nil` であれば、読み込み上限はありません。もし`t` であれば、バックエンドは記事を部分部分で読み込もうとはせず、記事全体を読み込みます。これは`ange-ftp` のあるバージョンで意味を持ちます。

`nnheader-head-chop-length`
この変数(ディフォルト 2048) は、前記の操作を行なっているときに、どれくらいの大きさの単位で各記事を読み込むかを設定します。

`nnheader-file-name-translation-alist`
これはファイル名の文字をどのように変換するかを指定する連想リストです。例えば、もし‘:’ があなたのシステムではファイル名の文字としては使えない場合(あなたは MS Windows の利用者ですね)、以下のようにすることができます。

```
(setq nnheader-file-name-translation-alist
      '((?: . ?_)))
```

実際には、これは MS Windows (ちぇっ!) システム上でのこの変数のディフォルト値(の一部) です。

`gnus-hidden-properties`
これは「不可視」テキストを隠すために使われる属性のリストです。ほとんどのシステムではディフォルトは`(invisible t intangible t)` で、これは不可視テキストを見えなくして触れないようにします。

`gnus-parse-headers-hook`
ヘッダーを解釈する前に呼び出されるフック。これは例えば、取得したヘッダーの統計情報を取るとか、あるいはある種のヘッダーを取り除くことに使うことができます。まあ、私は何でこんなものが欲しいかよくわかんないんだけどね。

`gnus-shell-command-separator`
二つのシェル命令を区切るのに使用される文字列。ディフォルトは‘;’ です。

`gnus-invalid-group-regexp`
利用者にグループ名を尋ねるときに使う、「無効な」グループ名に合致する正規表現です。ディフォルト値は Gnus の内部動作をめちゃめちゃにしてしまうかもしれない、いくつかの**本当に** 使えないグループ名を引っかけます。(通常、選択方法とグループの境界に使っている‘:’ などを許してしまうとまずい、ということです。)

IMAP の利用者はグループ名に‘/’ を使いたいかもしれませんが。

`gnus-safe-html-newsgroups`
Html 記事中のリンクがすべて安全であると見なされるグループです。値はそれらのグループに合致する正規表現、グループ名のリスト、または`nil` です。これは`mm-w3m-safe-url-regexp` より優先されます。ディフォルト値は`"\\`nnrss[+:]"` です。これは emacs-w3m で記事を描画するときだけ、つまり`mm-text-html-renderer` が`w3m` に設定されているときだけ効果があります。See Section “表示のカスタマイズ” in *The Emacs MIME Manual*.

11 終わり

はい、以上がマニュアルです---あなたはもう自分自身の人生を送ることができます。連絡をとってください。あなたの猫によろしく伝えてください。

おお、**神よ**---さよならを耐えることはできません。(すすり泣き。)

チャールズ・レズニコフはそれを非常によく表しているので、ここは彼のために譲ります:

> 賛美の歌(テデウム)
>
> 勝利ゆえにぼくは
> 歌うのではない、
> 勝利などひとつもないから、
> ありふれた日光のため、
> そよ風のため、
> 春の気前よさのために歌う。
>
> 勝利のためではなく
> 僕としては精一杯やった
> 一日の仕事のために。
> 玉座のためではなく
> みんなのテーブルの席で。

(新潮文庫「空腹の技法」著:ポールオースター、訳:柴田元幸、畔柳和代、ISBN:4102451080より引用)

12 付録

12.1 歴史

GNUS は梅田政信氏によって書かれました。1994 年の秋が忍び寄ってくるころ、退屈していたラルス・マッグヌ・イングブリグットスン(Lars Magne Ingebrigtsen) は Gnus を書き直そうと決心しました。

この非道な行為の責任者を調べてみたいのなら、あなたの(いまいましい!) ウェブブラウザーを`https://quimby.gnus.org/` に向けることができます。これは、かつては新しくて粋な版の Gnus の第一配布場所だったところで、Newsrc をぶっ壊して人々を激怒させるサイトとしても知られていましたが、近ごろでは Gnus は Emacs リポジトリーで開発されています。

最初のアルファ版の開発期間に、新しい Gnus は“(ding) Gnus”と呼ばれていました。*(ding)* はもちろん、*ding is not Gnus* の短縮形で、これはまったく完全な嘘ですが、だれがそんなことを気にするでしょうか? (ところで、この短縮形の“Gnus”はおそらく梅田さんの意図通り「ニュース」と発音されるべきで、そうするともっと適切な名前になります。そう思いませんか?)

どちらにせよ、すべてのエネルギーを新しい元気の良い名前を付けるのに使い果たした後で、その名前はあまりに 元気が良すぎるということになり、それを“Gnus”と再び命名しました。でも、今回は大文字と小文字を混ぜています。“Gnus”と“GNUS”です。新しいものと古いもの。

12.1.1 Gnus Versions

最初の「正しい」Gnus 5 のリリースは 1995 年 11 月に Emacs 19.30 の配布に含まれたときになされました(132 の(ding) Gnus のリリース足すことの Gnus 5.0 の 15 リリース)。

1996 年 5 月に次の世代の Gnus (別名“September Gnus”(99 リリースの後で)) が“Gnus 5.2”という名前でリリースされました(40 リリース)。

1996 年の 7 月 28 日に Red Gnus の作業が始まり、それは 1997 年 1 月 25 日に(84 リリースの後で)“Gnus 5.4”としてリリースされました(67 リリース)。

1997 年 9 月 13 日に、Quassia Gnus が開始され、37 リリース続きました。それは“Gnus 5.6”として 1998 年 3 月 8 日にリリースされました(46 リリース)。

1998 年 8 月 29 日に Gnus 5.6 から Pterodactyl Gnus が生まれ、1999 年 12 月 3 日に(99 リリースと CVS リポジトリでの作業の後)“Gnus 5.8”としてリリースされました。

2000 年 10 月 26 日に Oort Gnus が開始され、2003 年 5 月 1 日に Gnus 5.10 としてリリースされました(24 リリース)。

2004 年 1 月 4 日に No Gnus が始まりました。

2007 年 7 月、GNU Emacs 22.1 に Gnus 5.11 が同梱されました。

2009 年 7 月、No Gnus が GNU Emacs 23.1 とともに Gnus 5.13 としてリリースされました。

2010 年 4 月 19 日、Gnus の開発は Git に移行しました。

2012 年 1 月 31 日、Ma Gnus が始まりました。

それ以来 Gnus は Emacs と一緒にしかリリースされていません。

接頭語を持った版の Gnus—“(ding) Gnus”,“September Gnus”,“Red Gnus”,“Quassia Gnus”,“Pterodactyl Gnus”,“Oort Gnus”,“No Gnus”,“Ma Gnus”---に出会っても、混乱しないでください。あなたが恐がっていることを知られてはいけません。後ろに下がりなさい。ゆっくりと。他に何をしても、走ってはいけません。それが届かくなるまで、静かに歩き去りなさい。正しくリリースされた版の Gnus を見つけて、代わりにそれにすり寄りなさい。

12.1.2 なぜ?

Gnus の目的は何ですか?

私は、あなたの考え付くことをすべてできる「いかす」「盛り上がってる」「かっこいい」「しゃれた」ニュースリーダーを提供したいと思います。これは私の大元の動機だったのですが、Gnus の作業をしている間に、この世代のニュースリーダーは本当に石器時代に属していることが明らかになりました。ニュースリーダーは、インターネットの揺籃期からほとんど発展していませんでした。もし現在の増加率で流通量が増加しつづければ、すべての現在のニュースリーダーはまったく役に立たなくなるでしょう。毎日何千もの新しい記事がやってくるニュースグループを扱うにはどうすれば良いのでしょう? 数百万の投稿者に遅れないように付いていくにはどうすれば良いのでしょう?

Gnus はこれらの質問に真の解決を提供するわけではありませんが、私は Gnus がニュースを読み、取得するための新しい方法を実験する場として使われることを、是非とも見届けたいのです。ニュースリーダーをバックエンドから分離するという梅田さんの賢明な方針を拡張することによって、今や Gnus はメールを取得したり、異なる出所からニュースを取得するための新しいバックエンドを書きたい人のために、シンプルなインターフェースを提供しています。私は役に立ちそうなすべての場所に、カスタマイズのためのフックを加えました。それによって、探検し、発明したいすべての人を招いているのです。

おそらく Gnus は完成することはないのかもしれません。*C-u 100 M-x all-hail-emacs* です。

12.1.3 標準への準拠

理由無き反抗などと申すものはございませんよ、奥様。私たちはすべての知られている標準に準拠しています。もちろん私たちが賛成できない標準と/もしくは習慣は除きますが。

RFC 822 この標準またはその後継(現在は RFC の 2822 と 5322) への知られている違反はありません。

RFC 1036 この標準(今では廃版) も知られている違反はありません。

RFC 5536 RFC 1036 の後継です。これにはいくつかの違反があります。

X-Newsreader
User-Agent
: これらは「つまらないヘッダー」と見なされていますが、私は消費者の情報であると考えています。`tin` と`Netscape` から送られてくる非常に多くの酷い記事を見た後では、私は記事を投稿するためにはそれらを使わない方が良いということを知っています。もし`X-Newsreader` ヘッダーが無ければ、私はその情報を得ることはなかったでしょう。

USEFOR USEFOR は IETF の作業部会で、インターネット RFC の 5536 と 5537 を発行しました。Gnus タワーはこれらの基準によって具体化された変更を実装することを検討します。

MIME---RFC 2045--2049 etc
MIME 関連のすべての RFC がサポートされています。

Disposition Notifications - RFC 2298
Message Mode は受信者からの開封確認を要求することができます。

PGP---RFC 1991 and RFC 2440
RFC 1991 は最初のPGP メッセージの規格で、Informational RFC (訳注: 後述の標準化トラックではないが有用な情報) として発行されました。現在 Open PGP と呼ばれる後継の RFC 2440 が、標準化トラック(訳注: Standards Track---国際標準とすべき仕様) に乗せられました。どちらも非-MIME メッセージのためのPGP の様式を定義します。Gnus はエンコード(署名および暗号化) とデコード(認証および暗号のデコード) の両方をサポートします。

PGP/MIME---RFC 2015/3156
RFC 2015 (RFC 1991 の代わりに RFC 2440 に基づいた 3156 で置き換えられました) は、RFC 1991/2440 をMIME で囲う様式について述べています。Gnus はエンコードとデコードの両方をサポートします。

S/MIME---RFC 2633
RFC 2633 はS/MIME の形式について述べています。

IMAP---RFC 1730/2060, RFC 2195/4616, RFC 2086, RFC 2359, RFC 2595, RFC 1731
RFC 1730 はIMAP バージョン 4 で、RFC 2060 (IMAP 4 改定 1) で多少更新されています。RFC 2195 はIMAP の CRAM-MD5 認証について述べています。RFC 2086 はIMAP の使用制限一覧(ACL) について述べています。RFC 2359 はIMAP のプロトコルの拡張について述べています。RFC 2595 はIMAP における適切なTLS の統合(STARTTLS) について述べています。RFC 1731 はIMAP の GSSAPI/Kerberos4 の手法について述べています。RFC 4616 はIMAP の AUTH=PLAIN 認証について述べています。

上に書かれている文章に関することで、Gnus がそれを満たしていないような動作をしていることに気付いたら、ためらわずに Gnus タワーと私たちに知らせてください。

12.1.4 貢献者

新しい Gnus の版は(ding) メーリングリストのすべての人たちの助けが無ければできなかったでしょう。一年以上にわたって、私は毎日彼らから莫大な数の素敵なバグレポートを受け取り、そのそれぞれが私を喜びで満たしました。投げキッス。このリストの人たちは、私のリリース方針のために耐え難きを耐える試練に遭いました:「あぁ、それはすばらしい考えだ<かしゃかしゃかしゃ...> よしっ、すぐにリリースだ<えいやっ> あれれっ、まったく動かないぞ<かしゃかしゃかしゃ...> よしっ、すぐに出そう<ほらよっ> おっと、待った、ぜんぜん動作しない...」。Micro$oft---あっかんべーだ。アマチュアめ。私はもっと 悪い。(それとも「より悪い」?「もっと悪い」?「最悪」?)

私はこの機会に学会に感謝を... おおっと、違った。

- 梅田政信---元のGNUS を書いた人です。
- Shenghuo Zhu---uudecode.el, mm-uu.el, rfc1843.el, nnwarchive, それに一般的なバグ修正、新しい機能などはもとよりMIME と他の形式のエンコード/デコードに関連するほんとうに多くのもの。
- Per Abrahamsen---custom、スコア、ハイライトとSOUP コード(他の多くのことと共に)。

- Luis Fernandes---デザインとグラフィック。
- Joe Reiss---スマイリーの顔の作者。
- Justin Sheehy—FAQ のメインテイナー。
- Erik Naggum---手助け、アイデア、支援、コード他。
- Wes Hardaker—`gnus-picon.el` とマニュアルの*picon* の章(see Section 10.16.4 [Picons], p. 289)。
- Kim-Minh Kaplan---picon コードにおける更なる作業。
- Brad Miller—`gnus-gl.el` とマニュアルの Grouplens の章。
- Sudish Joseph---数え切れないほどのバグの修正。
- Ilja Weis—`gnus-topic.el`。
- Steven L. Baur---たくさんのたくさんのたくさんのバグの発見と修正。
- Vladimir Alexiev---refcard とリファレンスの小冊子。
- Felix Lee & Jamie Zawinski---私は Felix Lee と JWZ の XGnus 配布からいくつかの部分を盗みました。
- Scott Byer—`nnfolder.el` の拡張と改訂。
- Peter Mutsaers---孤児記事のスコアコード。
- Ken Raebburn—POP メールサポート。
- Hallvard B Furuseth---いろいろな小さな物や部分、特に.newsrc ファイルを扱う部分。
- Brian Edmonds—`gnus-bbdb.el`。
- David Moore—`nnvirtual.el` の改訂と多くの他のこと。
- Kevin Davidson—*ding* の名前を思い付きました。ですから、彼を責めてください。
- François Pinard---多くの、多くの興味深く完全なバグレポートと autoconf のサポート。

このマニュアル(Gnus 英語版) は Adrian Aichner と Ricardo Nassif, Mark Borges によって校正され、Jost Krieger によって一部分を校正されました。

以下の人々は多くのパッチと提案で貢献しました:

Christopher Davis, Andrew Eskilsson, Kai Grossjohann, Kevin Greiner, Jesper Harder, Paul Jarc, Simon Josefsson, David Kågedal, Richard Pieri, Fabrice Popineau, Daniel Quinlan, Michael Shields, Reiner Steib, Jason L. Tibbitts, III, Jack Vinson, 山岡克美, and Teodor Zlatanov.

それと、以下の人にもパッチやその他のものを感謝します:

Jari Aalto, Adrian Aichner, Vladimir Alexiev, Russ Allbery, Peter Arius, Matt Armstrong, Marc Auslander, Miles Bader, Frank Bennett, Alexei V. Barantsev, Robert Bihlmeyer, Chris Bone, Mark Borges, Mark Boyns, Rob Browning, Lance A. Brown, Kees de Bruin, Martin Buchholz, Joe Buehler, Kevin Buhr, Alastair Burt, Joao Cachopo, Zlatko Calusic, Massimo Campostrini, Castor, David Charlap, Dan Christensen, Kevin Christian, Jae-you Chung, James H. Cloos, Jr., Laura Conrad, Michael R. Cook, Glenn Coombs, Andrew J. Cosgriff, Neil Crellin, Frank D. Cringle, Geoffrey T. Dairiki, Andre Deparade, Ulrik Dickow, Dave Disser, Rui-Tao Dong, Joev Dubach, Michael Welsh Duggan, Dave Edmondson, Paul Eggert, Mark W. Eichin, Karl Eichwalder, 榎並嗣智, Michael Ernst, Luc Van Eycken, Sam Falkner, Nelson Jose dos Santos Ferreira, Sigbjorn Finne, Sven

Fischer, Paul Fisher, Decklin Foster, Gary D. Foster, Paul Franklin, Guy Geens, Arne Georg Gleditsch, David S. Goldberg, Michelangelo Grigni, Dale Hagglund, D. Hall, Magnus Hammerin, 半田剣一, Raja R. Harinath, 林芳樹, P. E. Jareth Hein, ひさしげけんじ, Scott Hofmann, Tassilo Horn, Marc Horowitz, Gunnar Horrigmo, Richard Hoskins, Brad Howes, Miguel de Icaza, François Felix Ingrand, 市川達哉, 石川一郎, Lee Iverson, 岩室元典, Rajappa Iyer, Andreas Jaeger, Adam P. Jenkins, Randell Jesup, Fred Johansen, Gareth Jones, Greg Klanderman, Karl Kleinpaste, Michael Klingbeil, Peter Skov Knudsen, 小林修平, Petr Konecny, 小関吉則, Thor Kristoffersen, Jens Lautenbacher, Martin Larose, Seokchan Lee, Joerg Lenneis, Carsten Leonhardt, James LewisMoss, Christian Limpach, Markus Linnala, Dave Love, Mike McEwan, Tonny Madsen, Shlomo Mahlab, Nat Makarevitch, Istvan Marko, David Martin, Jason R. Mastaler, Gordon Matzigkeit, Timo Metzemakers, Richard Mlynarik, Lantz Moore, 守岡知彦, Erik Toubro Nielsen, Hrvoje Nikšić, Andy Norman, Fred Oberhauser, C. R. Oldham, Alexandre Oliva, Ken Olstad, 大西雅晴, 小野秀貴, Ettore Perazzoli, William Perry, Stephen Peters, Jens-Ulrik Holger Petersen, Ulrich Pfeifer, Matt Pharr, Andy Piper, John McClary Prevost, Bill Pringlemeir, Mike Pullen, Jim Radford, Colin Rafferty, Lasse Rasinen, Lars Balker Rasmussen, Joe Reiss, Renaud Rioboo, Roland B. Roberts, Bart Robinson, Christian von Roques, Markus Rost, Jason Rumney, Wolfgang Rupprecht, Jay Sachs, Dewey M. Sasser, Conrad Sauerwald, Loren Schall, Dan Schmidt, Ralph Schleicher, Philippe Schnoebelen, Andreas Schwab, Randal L. Schwartz, Justin Sheehy, Danny Siu, Matt Simmons, Paul D. Smith, Jeff Sparkes, Toby Speight, Michael Sperber, Darren Stalder, Richard Stallman, Greg Stark, Sam Steingold, Paul Stevenson, Jonas Steverud, Paul Stodghill, 須藤清一, Kurt Swanson, Samuel Tardieu, Teddy, 戸沢晶彦, Chuck Thompson, Philippe Troin, James Troup, Trung Tran-Duc, Jack Twilley, Aaron M. Ucko, Aki Vehtari, Didier Verna, Vladimir Volovich, Jan Vroonhof, Stefan Waldherr, Pete Ware, Barry A. Warsaw, Christoph Wedler, Joe Wells, Lee Willis, and Lloyd Zusman.

Gnus のアルファ配布に含まれている ChangeLog は、それぞれの人たちが行なったことの完全な大要を伝える豊かな読み物です。(550KB といくらか)。(訳注: 非常に古い ChangeLog の記述が何度かばっさり捨てられましたが、それでも現在は非常に大きなサイズになっています。)

私が忘れてしまったすべての人に謝罪します。間違いなくたくさんの人を忘れてしまったことでしょう。

わぁ、こんなに人がいるとは思わなかった。これは本当に Gnus を使っている人がいるということなんでしょう。そんなことを誰が想像したでしょうか!

12.1.5 新しい機能

より最近の変更の要約については、通常の Emacs `NEWS` ファイルを参照してください。

これらのリストは、もちろんたいていの 重要な新しい機能に関する短い 要約でしかありません。いいえ、実は違います。もっともっとたくさんのものがあるのです。そう、事実上私たちは十分に用の無いもの(feeping creaturism) を持っているのです。

12.1.5.1 (ding) Gnus

Gnus 5.0/5.1 の新しい機能:

- すべてのバッファーの外観は、フォーマットのような変数(see Section 3.1 [Group Buffer Format], p. 12, and see Section 4.1 [Summary Buffer Format], p. 47) によって設定を変えることができるようになりました。

- ローカルスプールと、いくつかのNNTP サーバーを同時に使うことができるようになりました(see Chapter 7 [Select Methods], p. 146)。
- 複数のグループを仮想グループに合併できるようになりました(see Section 7.7 [Virtual Groups], p. 209)。
- 多くの異なるメール様式を読めるようになりました(see Section 7.4 [Getting Mail], p. 162)。すべてのメールバックエンドは、便利なメール期限切れ消去機構を実装しています(see Section 7.4.9 [Expiring Mail], p. 181)。
- Gnus は根っこ(root) を失ったスレッドを集めるためのいろいろな戦略(それによってまばらな副スレッドを一つのスレッドにする) を使ったり、完全なスレッドを組み上げるのに十分なヘッダーをいったん戻って取得することができます(see Section 4.9.1 [Customizing Threading], p. 70)。
- 切られたグループ(killed groups) はグループバッファーに表示することができて、それらも読むことができます(see Section 3.11 [Listing Groups], p. 31)。
- Gnus はグループを部分的に更新することができます---2,3 のグループの新しい記事を調べるために、アクティブファイル全体を取得する必要はありません(see Section 2.8 [The Active File], p. 10)。
- Gnus はグループの段階的購読度を実装しました(see Section 3.6 [Group Levels], p. 19)。
- 何種類もの基準に従って、記事にスコアを付けることができます(see Chapter 8 [Scoring], p. 233)。どのように記事にスコアを付けるかを、Gnus に見つけさせることもできます(see Section 8.6 [Adaptive Scoring], p. 244)。
- Gnus は普通の Emacs の方法で自動保存されるドリブルバッファーを維持するので、あなたが何を読んだかのデータをマシンが落ちたときでもあまり失わないでしょう(see Section 2.7 [Auto Save], p. 9)。
- Gnus は`.emacs` ファイルをぐちゃぐちゃにすることを避けるために、今では専用の起動ファイル(`~/.gnus.el`) を持つようになりました。
- グループと記事の両方にプロセス印を付けることができ、すべての印の付いた項目で処理を実行することができます(see Section 10.1 [Process/Prefix], p. 271)。
- グループの一覧を、えーと、どんな条件ででも、表示することができます(see Section 3.11 [Listing Groups], p. 31)。
- 外部サーバーを概観して、それらのサーバーのグループを購読することができます(see Section 3.14 [Browse Foreign Server], p. 34)。
- Gnus はサーバーとの二つ目の接続で、記事を非同期に取ってくることができます(see Section 4.11 [Asynchronous Fetching], p. 77)。
- 記事をローカルにキャッシュすることができます(see Section 4.12 [Article Caching], p. 79)。
- uudecode の関数が拡張され、一般化されました(see Section 4.17 [Decoding Articles], p. 86)。
- uuencode された記事をまだ投稿することができます。これは過去にGNUS のあまり知られていない機能でした(see Section 4.17.5.3 [Uuencoding and Posting], p. 89)。
- 親記事(と他の記事) の取得は、今では調子が悪くなることも無く、実際に動作するようになりました(see Section 4.23 [Finding the Parent], p. 109)。
- Gnus はFAQ とグループの説明を取得することができます(see Section 3.18.2 [Group Information], p. 44)。

- まとめ送りされた記事(および他のファイル) を、グループとして使えるようになりました(see Section 7.6.3 [Document Groups], p. 204)。
- 記事をハイライトし、カスタマイズすることができます(see Section 5.4 [Customizing Articles], p. 129)。
- URL と他の外部参照がボタンになるようになりました(see Section 4.18.6 [Article Buttons], p. 98)。
- Gnus のウィンドウとフレームの設定でたくさんの変なことをできるようになりました(see Section 10.5 [Window Layout], p. 276)。

12.1.5.2 September Gnus

Gnus 5.2/5.3 の新しい機能:

- 新しいメッセージ作成モードが使われます。`mail-mode`, `rnews-reply-mode` と`gnus-msg` のすべての古いカスタマイズ変数は今や旧式になりました。
- Gnus は「まばら」スレッドを作成することができるようになりました---スレッドの失われた記事があるところは、空の節で表現されるようになっています(see Section 4.9.1 [Customizing Threading], p. 70)。

```
(setq gnus-build-sparse-threads 'some)
```

- 外に出ていく記事は、特別な保管サーバーに保存されるようになりました(see Section 6.5 [Archived Messages], p. 137)。
- 記事が参照されたときに、スレッドの部分作成が行なわれるようになりました。
- Gnus は GroupLens の予測を利用することができるようになりました。
- Picons (personal icons) (個人アイコン) が表示できるようになりました(see Section 10.16.4 [Picons], p. 289)。
- `trn` のような木バッファーを表示できるようになりました(see Section 4.25 [Tree Display], p. 112)。

```
(setq gnus-use-trees t)
```

- ニュースリーダー`nn` のような、選んで読むマイナーモードを概略バッファーで使うことができるようになりました(see Section 4.24.1 [Pick and Read], p. 111)。

```
(add-hook 'gnus-summary-mode-hook 'gnus-pick-mode)
```

- バイナリーグループで特別なバイナリーマイナーモードを使うことができるようになりました(see Section 4.24.2 [Binary Groups], p. 112)。
- グループ群を折り畳みトピック階層にグループ分けできるようになりました(see Section 3.16 [Group Topics], p. 35)。

```
(add-hook 'gnus-group-mode-hook 'gnus-topic-mode)
```

- メールの再送と、弾かれたメールを送り直すことができるようになりました(see Section 4.5.1 [Summary Mail Commands], p. 57)。
- グループがスコアを持つことができるようになり、訪れる回数に基づいた並べ替えが可能になりました(see Section 3.7 [Group Score], p. 20)。

```
(add-hook 'gnus-summary-exit-hook 'gnus-summary-bubble-group)
```

- グループにプロセス印を付けられるようになり、グループのグループに対して命令を実行できるようになりました(see Section 3.8 [Marking Groups], p. 21)。
- 仮想グループでキャッシュができるようになりました。

- nndoc はすべての種類のまとめ送り、メールボックス、rnews ニュースの一括配送、ClariNet の要約集、そしてその他のすべてを理解できるようになりました(see Section 7.6.3 [Document Groups], p. 204)。
- Gnus は SOUP パケットを作成/読み込みをするための新しいバックエンド(nnsoup)を持っています。
- キャッシュがずっと速くなりました。
- グループを多くの基準に従って並べ替えることができるようになりました(see Section 3.12 [Sorting Groups], p. 32)。
- メーリングリストのアドレスと期限切れ消去の時間を設定する、新しいグループパラメーターが導入されました(see Section 3.10 [Group Parameters], p. 23)。
- すべてのフォーマット指定で、フェースを指定できるようになりました(see Section 10.4.5 [Formatting Fonts], p. 275)。
- *M P* 副キーマップに、プロセス印の付いた記事の設定/削除/実行のための複数の命令が追加されました(see Section 4.7.6 [Setting Process Marks], p. 66)。
- 広範囲の基準に基づいて、概略バッファーが利用可能な記事の一部だけを表示するように制限できるようになりました。これらの命令は/ 副マップのキーにバインドされています(see Section 4.8 [Limiting], p. 67)。
- * 命令によって、記事を永続させることができるようになりました(see Section 4.13 [Persistent Articles], p. 80)。
- 記事の要素を隠すすべての関数は、トグルになりました。
- 記事のヘッダーにボタンを付けることができるようになりました(see Section 4.18.4 [Article Washing], p. 94)。
- すべてのメールバックエンドで、Message-ID による記事の取得をサポートするようになりました。
- 重複メールを適切に扱うことができるようになりました(see Section 7.4.11 [Duplicates], p. 186)。
- すべての概略モード命令を、記事バッファーから直接使用できるようになりました(see Section 5.5 [Article Keymap], p. 132)。
- フレームがgnus-buffer-configuration の部分になることができるようになりました(see Section 10.5 [Window Layout], p. 276)。
- デーモンのプロセスによって、新着メールを検査できるようになりました(see Section 10.10 [Daemons], p. 284)。
- グループを常に見えるようにしておくことが(訳注: そのグループに未読記事が無くても)、できるようになりました(see Section 3.11 [Listing Groups], p. 31)。

```
(setq gnus-permanently-visible-groups "^nnml:")
```

- カスタマイズを楽にするために、多くの新しいフックが導入されました。
- Gnus はMail-Copies-To ヘッダーを尊重するようになりました。
- References ヘッダーを調べることによって、スレッドを集めることができるようになりました(see Section 4.9.1 [Customizing Threading], p. 70)。

```
(setq gnus-summary-thread-gathering-function
      'gnus-gather-threads-by-references)
```

- 再取得を避けるために、既読記事を特別なバックログ・バッファーに貯めることができるようになりました(see Section 4.15 [Article Backlog], p. 81)。

```
(setq gnus-keep-backlog 50)
```

- 簡単にトリートメントを行なうことができるようにするために、現在の記事の完全な複製がいつも別バッファーに置かれるようになりました。
- Gnus がどこに記事を保存するかを提案できるようになりました(see Section 4.16 [Saving Articles], p. 81)。
- 記事を保存するときに、多くを入力しなくても良いようになりました(see Section 4.16 [Saving Articles], p. 81)。

```
(setq gnus-prompt-before-saving t)
```

- gnus-uu は記事を取得している間に、非同期でデコードされたファイルを表示できるようになりました(see Section 4.17.5.2 [Other Decode Variables], p. 88)。

```
(setq gnus-uu-grabbed-file-functions 'gnus-uu-grab-view)
```

- 記事バッファーで、引用されたテキストの折り返しが適切に動作するようになりました(see Section 4.18.4 [Article Washing], p. 94)。
- 引用されたテキストを表示するか隠すかを切り替えるためのボタンが追加されました。また、どのくらいの引用文を隠すかをカスタマイズできるようになりました(see Section 4.18.3 [Article Hiding], p. 92)。

```
(setq gnus-cited-lines-visible 2)
```

- 興味の無いヘッダーを隠すことができます(see Section 4.18.3 [Article Hiding], p. 92)。
- スコアのディフォルト値をメニューバーから設定できるようになりました。
- 送信される記事の更なる構文チェックが追加されました。

12.1.5.3 Red Gnus

Gnus 5.4/5.5 の新しい機能:

- nntp.el は非同期に動作するやり方で、完全に改訂されました。
- 記事の先行取得を行なう機能が Gnus に編入されました(see Section 4.11 [Asynchronous Fetching], p. 77)。
- スコア付けはand, or, not のような論理演算子と、親記事にさかのぼってリダイレクトすることで実行できるようになりました(see Section 8.15 [Advanced Scoring], p. 253)。
- 記事の洗濯状態を記事のモード行に表示できるようになりました(see Section 5.6 [Misc Article], p. 133)。
- gnus.el が多くの小さいファイルに分割されました。
- Message-ID に基づいて、記事の重複を抑制することができるようになりました(see Section 4.30 [Duplicate Suppression], p. 121)。

```
(setq gnus-suppress-duplicates t)
```

- どのスコアと適応ファイルが、ホームスコアと適応ファイルであるかを指定する(see Section 8.7 [Home Score File], p. 247) 新しい変数が加えられました。
- nndoc がより簡単に拡張できるように改訂されました(see Section 7.6.3.1 [Document Server Internals], p. 205)。
- グループは親のトピックからグループパラメーターを継承できるようになりました(see Section 3.16.5 [Topic Parameters], p. 40)。

- 記事を編集するための機能が継ぎはぎされて、実際に使用可能になりました。
- 署名がもっと利口なやり方で認識されるようになりました(see Section 4.18.10 [Article Signature], p. 102)。
- 概略ピックモードがもっと(ニュースリーダー) `nn` らしくなりました。行番号が表示され、記事を選ぶために . 命令を使うことができるようになりました(`Pick and Read`)。
- あるサーバーから別のサーバーへ`.newsrc.eld` を移動する命令が加えられました(see Section 2.5 [Changing Servers], p. 7)。
- 今では、バッファーの行を作成するときに、抑制される「興味の無い」部分を指定する方法があります(see Section 10.4.3 [Advanced Formatting], p. 273)。
- グループバッファーの複数の命令について、それらで行なったことを*C-M-_* で元に戻すことができるようになりました(see Section 10.12 [Undo], p. 285)。
- 新しいスコア型`w` を使うことによって、単語でスコア付けをすることが可能になりました(see Section 8.4 [Score File Format], p. 239)。
- 表題の一語一語を基にして、適応スコアをすることができるようになりました(see Section 8.6 [Adaptive Scoring], p. 244)。

```
(setq gnus-use-adaptive-scoring '(word))
```

- スコアを減衰させることができるようになりました(see Section 8.16 [Score Decays], p. 255)。

```
(setq gnus-decay-scores t)
```

- 正規表現を使って、日付のヘッダーでスコア付けを実行できるようになりました。日付は最初にコンパクトな ISO 8601 様式で正規化されます(see Section 8.4 [Score File Format], p. 239)。
- 記事に関するすべてのデータを、基本のサーバーから取り除く命令が加えられました(see Section 2.5 [Changing Servers], p. 7)。
- 文書を寄せ集めたものを読むための新しい命令(`nndoc` グループのてっぺんで`nnvirtual` を使います) が加えられました—*C-M-d* (see Section 4.27.4 [Really Various Summary Commands], p. 118)。
- プロセス印の設定を push と pop でスタックに出し入れできるようになりました(see Section 4.7.6 [Setting Process Marks], p. 66)。
- NNTP サーバーが投稿を許可していない場合でも、新しい mail-to-news バックエンドが、投稿することを可能にしました(see Section 7.6.4 [Mail-To-News Gateways], p. 207)。
- ウェブ検索エンジン(*DejaNews*, *Alta Vista*, *InReference*) からの検索結果を読むための、新しいバックエンドが加えられました(see Section 7.5.1 [Web Searches], p. 198)。
- 標準の並べ替え関数を使って、トピックの中にあるグループを並び代えることができるようになりました。また、それぞれのトピックを独立して並べ替えることができるようになりました(see Section 3.16.3 [Topic Sorting], p. 39)。
- グループ群の一部を、独立して並べ替えることができるようになりました(`Sorting`)。
- キャッシュされた記事を、グループに引き込むことができるようになりました(see Section 4.27.3 [Summary Generation Commands], p. 118)。
- スコアファイルがもっと信頼できる順番で適用されるようになりました(see Section 8.3 [Score Variables], p. 236)。
- メールメッセージが分割されてどこに行くかの報告を、作成することができるようになりました(see Section 7.4.3 [Splitting Mail], p. 164)。

- 入って来たメールを保存する前にがらくたを取り除くフックと関数が、もっと追加されました(see Section 7.4.10 [Washing Mail], p. 184)。
- 強調表示することを指定されたテキストが、適切に表示されるようになりました。

12.1.5.4 Quassia Gnus

Gnus 5.6 の新しい機能:

- Gnus をオフラインニュースリーダーとして使う新機能が加えられました。過剰なほどの新しい命令とモードが追加されました。全貌についてはSection 7.9 [Gnus Unplugged], p. 216, を見てください。
- `nndraft` バックエンドが戻ってきました。でも、依然とは違う動作をします。すべてのメッセージバッファーは、今では自動的に作成される`nndraft` グループの記事でもあります。
- `gnus-alter-header-function` を、ヘッダーの値を変えるために使うことができるようになりました。
- `gnus-summary-goto-article` が Message-ID を受け付けるようになりました。
- メッセージの本文において、指定したリージョン以外のテキストを消去するための新しいメッセージ命令があります: *C-c C-v*。
- *C-u C-c C-c* によって`nnvirtual` グループを構成しているグループに投稿できるようになりました。
- `nntp-rlogin-program`---カスタマイズを簡単にするための新しい変数です。
- `gnus-article-edit-mode` における`C-u C-c C-c` 命令は、記事バッファーの再ハイライトを抑制するようになりました。
- `gnus-boring-article-headers` に、`long-to` という新しい要素があります。
- *M-i* シンボル接頭引数命令があります。詳細はSection 10.3 [Symbolic Prefixes], p. 272, を見てください。
- 概略バッファーにおける*L* と*I* は、`all.SCORE` ファイルにスコア規則を加えるためのシンボル接頭引数`a` を受け付けるようになりました。
- 変数`gnus-simplify-subject-functions` によって、表題の単純化を強力に制御できるようになりました。
- *A T*---現在のスレッドを取得するための新しい命令です。
- */ T*---現在のスレッドを制限に含めるための新しい命令です。
- *M-RET* は、引用文の途中に割って入るための新しいメッセージ命令です。
- '`\\1`' のような表現が`nnmail-split-methods` で有効になりました。
- 関数`custom-face-lookup` が取り除かれました。あなたの初期化ファイルでこの関数を使っていたのなら、代わりに`face-spec-set` を使うように書き直さなければなりません。
- 投稿のキャンセルに、現在の選択方法を使うようになりました。シンボル接頭引数`a` で、普通の投稿方法を強制することができます。
- マ■■■ソ■■ sm*rtq**t*s を適切なテキストに翻案する新しい命令があります—*W d*。
- `nntp` のデバッグを楽にするために、`nntp-record-commands` を`nil` ではない値に設定することができます。

- nntp は~/.authinfo を使うようになりました。これは.netrc のようなファイルで、どこのNNTP サーバーにはどのようにAUTHINFO を送るかを制御するためのものです。
- 概略バッファーのグループパラメーターを編集するための命令が加えられました。
- メールがどこに分割されたかの履歴を利用できるようになりました。
- 記事の日付を表示するための新しい命令が加えられました—article-date-iso8601。
- gnus-score-thread-simplify を設定することによって、スレッドを作成するときの表題を単純化できるようになりました。
- メッセージで引用をするための新しい関数が加えられました— message-cite-original-without-signature。
- article-strip-all-blank-lines---新しい記事命令です。
- 記事の終わりまでを切り取る(kill する) 新しいメッセージ命令が加えられました。
- 変数gnus-adaptive-word-minimum を使うことによって、最小限度の適応スコアを指定することができます。
- gnus-start-date-timer 命令によって「記事が投稿されたときからの経過時間」ヘッダーが継続的に更新されるようになりました。
- ウェブで提供されているメーリングリストのアーカイブを、nnlistserv バックエンドによって読むことができるようになりました。
- 古い dejanews アーカイブをnnweb で読むことができるようになりました。

12.1.5.5 Pterodactyl Gnus

Gnus 5.8 の新しい機能:

- メールを取り込む機能が変わりました。たくさんの詳細についてはマニュアルを見てください。特に procmail で取り込むためのすべての変数が無くなっています。

  以下のような procmail の使い方は

  ```
  (setq nnmail-use-procmail t)
  (setq nnmail-spool-file 'procmail)
  (setq nnmail-procmail-directory "~/mail/incoming/")
  (setq nnmail-procmail-suffix "\\.in")
  ```

  現在では次のように変わっています。

  ```
  (setq mail-sources
        '((directory :path "~/mail/incoming/"
                     :suffix ".in")))
  ```

  See Section 7.4.4.1 [Mail Source Specifiers], p. 166.
- Gnus はMIME に対応したリーダーになりました。これは Gnus の多くの部分に影響していて、たくさんの新しいコマンドが追加されています。詳細はマニュアルを参照してください。
- しかも Gnus は各国語対応になりました。ここでは要約できないくらいに Gnus の多くの部分に影響していて、新しいたくさんの変数が追加されています。
- gnus-auto-select-first が、ポイントを置く場所を決定するための関数であってもよくなりました。
- 概略バッファーとNOV ファイルに含める追加のヘッダーを、利用者が決めることができるようになりました。

- gnus-article-display-hook が削除されました。代わりにgnus-treat- で始まるたくさんの変数が追加されました。
- Gnus posting styles が再び作り直されました。現在は微妙に違うやり方で動作します。
- 新しいウェブに基づいたバックエンドが追加されました。nnslashdot, nnwarchive およびnnultimate です。nnweb は常に変化する構成をとり続けるために、再び作り直されました。
- Gnus はnnimap によってIMAP のメールを読むことができます。

12.1.5.6 Oort Gnus

Gnus 5.10 の新しい機能:

- インストールに関する変更
  - Oort を使ったことがある場合の、以前の(安定な) 版からのグレードアップ。

    Oort (このリリースに先立つ安定ではない Gnus の枝) を使ってみたものの、安定版に戻してしまったならば、この版にグレードアップするときに注意してください。特に、おそらくすべての.marks (nnml) と.mrk (nnfolder) ファイルを消去する必要があるでしょう。その目的は、この版(の Gnus) がフラグを格納する.marks/.mrk ファイルではなくて.newsrc.eld からフラグが読まれるようにするためです(訳注: 言い換えると、古い様式の.marks/.mrk ファイルを新しい Gnus が読んではいけないということです。それらは新たに作成されます)。後述の項目で、印(marks) に関するより多くの情報を読んでください。グレードを下げても一般には助けにならないことに注意してください。
  - Lisp ファイルがディフォルトで.../site-lisp/gnus/ にインストールされるようになりました。以前は.../site-lisp/ がディフォルトでした。加えて、新しいインストーラーは他にインストールされている、新しい Gnus より優先される Gnus を検出して警告を発します。それらを手動で取り除いても良いし、make remove-installed-shadows を使って削除することもできます。
  - ~/News/overview/ は不要。

    以下の変更の結果、もはや~/News/overview/ ディレクトリーは要りません。すべての階層を安全に削除することができます。
  - (require 'gnus-load)

    単独で配布されている Gnus を使う場合には、load-path に Gnus の lisp ディレクトリーを追加してから、~/.emacs に(require 'gnus-load) を加えるのが良いです。

    gnus-load.el ファイルは、そのうちのいくつかは Emacsen の配布に入っていないかもしれない自動読み込み(autoload) コマンド、関数および変数を含んでいます。
- Gnus に内蔵された新しいパッケージとライブラリー
  - 改定された Gnus FAQ がマニュアルに含まれています。See Section 12.9 [Frequently Asked Questions], p. 386.
  - TLS ラッパーが Gnus に同梱されました。

    TLS/SSL が GnuTLS を介してIMAP とNNTP でサポートされるようになりました。

- 改良された spam 対抗機能。

  Gnus は非常に変化に富んだプログラムと濾過の規則を使って、メールやニュースの奔流から spam を抜き取ってしまうことができるようになりました。対応している方式は、RBL blocklists、bogofilter それにホワイト/ブラックリストです。また SpamAssassin や Hashcash のような外部パッケージを容易に使うための hook も新しくなりました。Section 10.18 [Thwarting Email Spam], p. 291, およびSection 10.19 [Spam Package], p. 295.

- Gnus は Sieve を使ったサーバー側でのメールの濾過をサポートします。

  Sieve の規則はグループパラメーターとして加えることができ、グループバッファーで*D g* を使うと完全な Sieve スクリプトが生成されます。そうしたら、生成された Sieve バッファーで*C-c C-l* を使って、サーバーにアップロードしてください。Section 3.18.5 [Sieve Commands], p. 45、それに新しい Sieve のマニュアル(see *Emacs Sieve*) を参照してください。

- グループモードの変更
  - `gnus-group-read-ephemeral-group` を*G M* キーで対話的に呼ぶことができます。
  - 憲章とコントロールメッセージの取得。

    二つの新しいコマンドで、ニュースグループの憲章を取り込む(*H c*) ことと、コントロールメッセージを取得する(*H C*) ことができます。
  - 新しい変数`gnus-parameters` を、グループパラメーターを設定するために使うことができます。

    これは初期には、パラメーターを`~/.newsrc.eld` に格納する*G p* (または*G c*) でしか行なうことができませんでしたが、この変数によってカスタマイズの威力を堪能することができます。また、その変数は`~/.newsrc.eld` ではなくて`~/.gnus.el` で設定するので、バックアップが簡単になります。その変数は、グループ名に合致する正規表現を、以下のような流儀でグループパラメーターに割り当てます:

```
(setq gnus-parameters
      '(("mail\\..*"
         (gnus-show-threads nil)
         (gnus-use-scoring nil))
        ("^nnimap:\\(foo.bar\\)$"
         (to-group . "\\1"))))
```

  - nnimap のグループにおける未読の数が正確になりました。

    グループバッファーで表示される nnimap グループの未読記事の数の見積りが正確になったはずです。これは`gnus-setup-news-hook` (起動時に呼ばれる) と`gnus-after-getting-new-news-hook` (新しいメールを取得した直後に呼ばれる) から`nnimap-fixup-unread-after-getting-new-news` を呼ぶことによって成し遂げられます。これらの変数をディフォルトではない値に変えている場合は、重ねて`nnimap-fixup-unread-after-getting-new-news` を追加する必要があるかもしれません。見積りに満足していて、新しいメールを取得するときにいくらかの(わずかな) 時間を節約したいのであれば、その関数を外してください。
  - グループ名は、ディフォルトで UTF-8 であるものとして取り扱われます。

    これは USEFOR が移行しようとしていると想定されるものです。カスタマイズするには`gnus-group-name-charset-group-alist` および`gnus-group-name-charset-method-alist` を参照してください。

- `gnus-group-charset-alist` と`gnus-group-ignored-charsets-alist`
  これらの変数に設定された正規表現は、完全な(full) グループ名と比較されます。Gnus 5.8 では実際の(real) グループ名が比較の対象でした。したがって、これらの変数をカスタマイズしている利用者は、正規表現を変更しなければなりません。例です:

  ```
  ("^han\\>" euc-kr) -> ("\\(^\\|:\\)han\\>" euc-kr)
  ```

- 入ってきたメールを一時蓄えるファイル(`Incoming*`) の古いものは、即時ではなく何日か後に消去されます。See Section 7.4.4.3 [Mail Source Customization], p. 172. (Gnus 5.10.10 / Emacs 22.2 の新機能)

- 概略モードと記事モードの変更
  - 領域が活性化されている場合に、*F* キー(`gnus-article-followup-with-original`) および*R* キー(`gnus-article-reply-with-original`) は、その領域にあるテキストだけを yank します。
  - ドラフト・グループで*e* キーが`gnus-draft-edit-message` コマンドに割り当てられました。`gnus-summary-edit-article` コマンドには、代わりに*B w* キーを使ってください。
  - 記事のボタン。

    URL、メールアドレス、Message-ID、Info へのリンク、man ページと Emacs または Gnus に関連した参考文献のための、より多くのボタンが追加されました。See Section 4.18.6 [Article Buttons], p. 98. すべての記事のボタンの見栄えを制御するために`gnus-button-*-level` 変数を使うことができます。See Section 4.18.7 [Article Button Levels], p. 100.
  - 単一の yenc でエンコードされた添付パートをデコードすることができます。
  - Picons

    Picon のコードが、GNU Emacs で動作させるために再実装されました。以前のいくつかのオプションが、削除または改名されています。

    Picon は、利用者、ドメイン、それにニュースグループを表現するための「個人的なアイコン(personal icons)」で、記事バッファーに表示することができます。See Section 10.16.4 [Picons], p. 289.
  - 新しいオプション`gnus-treat-body-boundary` を非-`nil` にすると、ヘッダーのおしまいに境界線が描かれます。
  - 署名された記事のヘッダー(X-PGP-Sig) を、*W p* で認証することができます。
  - 概略バッファーは fringe の中の矢印で現在の記事を示します。これを無効にするには`(setq gnus-summary-display-arrow nil)` を使ってください。
  - ニュースにメールで返信しようとしたら警告します。

    間違ってニュースにメールで返信しようとしてしまうことが、しょっちゅうありませんか? そんなあなたに新オプション`gnus-confirm-mail-reply-to-news`。
  - 新しいオプション`gnus-summary-display-while-building` を非-`nil` にすると、概略バッファーが作られていく様子が表示されます。
  - Gnus は RFC 2369 のメーリングリストのヘッダーをサポートします。また、メーリングリストのグループ用に数々のコマンドを用意しました。See Section 4.32 [Mailing List], p. 124.

- 日付ヘッダーを、英語で発音できる形式で表示することができます。See Section 4.18.8 [Article Date], p. 100.
- `mm-uu-diff-groups-regexp` に合致するグループでは、差分(diffs) が自動的にハイライトされます。
- マイクロソフト引用様式のより良い取り扱い。

  いくつかのマイクロソフトのメイラーが、メッセージの残りの部分が引用であることを示すために使う台無しにされたヘッダーブロックを、たとえそれが引用符で囲まれていなくても、Gnus はとにかく認識しようとします。変数`gnus-cite-unsightly-citation-regexp` は、それらの引用の先頭に合致します。

  新しい*W Y f* コマンド(`gnus-article-outlook-deuglify-article`) で、醜く壊れた Outlook (Express) の記事を整形し直すことができます。

- `gnus-article-skip-boring`

  `gnus-article-skip-boring` を`t` に設定すると、Gnus はうんざりする文しか含んでいないページを見せるために、下方にスクロールしません。`gnus-article-boring-faces` を使って、何を読み飛ばしてしまっても良いかをカスタマイズすることができます。

  てっぺんに少しだけある新規な内容に、長くて刈り込まれていない引用が続いているたくさんの記事を読む場合に、これは特に役に立ちます。

- スマイリー(‘`:-)`’, ‘`;-)`’ など) が Emacs でもアイコン化されるようになりました。

  これを働かないようにするには、`(setq gnus-treat-display-smileys nil)` を`~/.gnus.el` に置いてください。

- Face ヘッダーを扱えるようになりました。See Section 10.16.2 [Face], p. 288.
- 概略バッファーで、新しいコマンド*/ N* は新着メッセージを挿入し、*/ o* は古いメッセージを挿入します。
- *W m* を押すと、Gnus はモールスでエンコードされたメッセージをデコードします。
- `gnus-summary-line-format`

  ディフォルト値が‘`%U%R%z%I%(%[%4L: %-23,23f%]%) %s\n`’ に変わりました。さらに、受信者の名前かNNTP グループに投稿したグループ名で利用者名を置き換えるために、`gnus-extra-headers`、`nnmail-extra-headers` および`gnus-ignored-from-addresses` のディフォルト値が変わりました。

- 添付ファイルの消去。

  `gnus-mime-save-part-and-strip` コマンド(MIME ボタン上で*C-o* に割り当てられている) は、パートをセーブしてから外部のそれと置き換えます。`gnus-mime-delete-part` (MIME ボタン上で*d* に割り当てられている) は、パートを削除します。これは編集をサポートしているバックエンドでだけ動作します。

- `gnus-default-charset`

  ディフォルト値は`iso-8859-1` に代わって`current-language-environment` 変数によって決定される値になります。また、`gnus-group-charset-alist` にあった‘`.*`’ の項目は削除されました。

- 印刷の性能が向上しました。

  Gnus はそれ自身が、概略と記事バッファーにおける*O P* で Muttprint をサポートします。さらにMIME ボタン上で*p* を使うことによって、個々のMIME パートのそれぞれを印刷することができます。

- 拡張された書法仕様(format specs)。

  書法仕様'`%&user-date;`' が`gnus-summary-line-format-alist` に追加されました。それに、利用者定義による拡張されたフォーマットの仕様もサポートされています。拡張された書法仕様は'`%u&foo;`' のようなもので、関数`gnus-user-format-function-`*foo* を起動します。'`&`' がエスケープ文字に使われているので、古い利用者定義書法である'`%u&`' は今ではサポートされていません。

- */ **(`gnus-summary-limit-include-cached`) が書き直されました。

  これは*Y c* (`gnus-summary-insert-cached-articles`) の別名でした(訳注: 以前は)。新しい関数は他の記事を濾過して除去します。

- いくつかの制限命令は*C-u* 接頭引数で合致の否定を扱うことができます。

  C-u を subject、author または extra ヘッダー、すなわち*/ s*、*/ a* および*/ x* (`gnus-summary-limit-to-{subject,author,extra}`) で使うと、結果としてその表現に合致しないすべての記事が表示されます。

- Gnus は外部パート(message/external) をインライン表示します。

- Message モードの変更と関連する Gnus の機能

  - 遅延記事。

    Message バッファーにおける*C-c C-j* で、メッセージの送信を遅らせることができます。メッセージは指定された時刻に配送されます。これはあなた自信のための忘備録として役に立つでしょう。See Section 4.6 [Delayed Articles], p. 61.

  - `auto-compression-mode` が有効になっていると、添付ファイルを見るときに自動で圧縮が解かれます。

  - 新しいオプション`gnus-gcc-mark-as-read` は、Gcc の記事に自動的に既読の印を付けます。

  - 添付ファイルの切り離し(externalizing)。

    `gnus-gcc-externalize-attachments` または`message-fcc-externalize-attachments` が非`-nil` になっていると、ローカルファイルを外部パートとして添付します。

  - Sendmail を使うときのエンベロープ送信者(envelope sender) のアドレスが、カスタマイズできるようになりました。See Section “メール変数” in *The Message Manual*.

  - Gnus は今では Sender: ヘッダーを自動では生成しません。

    それは初期においては、利用者が設定した email アドレスが Gnus が想定した利用者のディフォルトのアドレスと違っていた場合に生成されました。今日ではその想定アルゴリズムが正しいことはまれで、Sender: ヘッダーの唯一の(議論の的になる) 用途は、ニュースを cancel/supersede する資格があるかどうかを検査すること(これは代わりに、他の章で述べられる Cancel Locks によって解決されました) なので、そのヘッダーの生成はディフォルトで抑制されています。変数`message-required-headers`、`message-required-news-headers` および`message-required-mail-headers` を参照してください。

- サードパーティーによるmessage-utils.el の機能がmessage.el に加えられました。

  Message は表題の行から‘(was: <old subject>)’ を削除するかどうかを尋ねるようになりました(message-subject-trailing-was-query 参照)。*C-c M-m* と*C-c M-f* は挿入されたテキストを示す印を挿入します。*C-c C-f a* は X-No-Archive: ヘッダーを付け加えます。*C-c C-f x* は、適切なヘッダーと、クロスポストとフォロー先についての注意書きを本文に挿入します(message-cross-post-* 変数群を見てください)。
- 今やmessage-generate-headers-first がnil だったら、メッセージの作成を始めるときに References と X-Draft-From ヘッダーは生成されません。
- X-Faces ヘッダーの挿入が簡単になりました。See Section 10.16.1 [X-Face], p. 287.
- グループカーボンコピー(GCC) を引用符で囲む。

  空白や他の変な文字を含むグループを扱えるようにするために、グループは Gcc: header に置かれる前に引用符で囲まれます。これは、空白を含むグループが使えるようにするために、もはやgnus-message-archive-group のような変数に引用文字を含めるべきではないことを意味します。さらに、文字列‘nnml:foo, nnml:bar’ (二つのグループに Gcc を格納することを示す) を使っているならば、("nnml:foo" "nnml:bar") というリストを返すように変更しなければなりません。さもないと、Gcc: 行は間違った囲まれ方をされてしまうでしょう。初期のころに文字列‘nnml:foo, nnml:bar’ を返すようにしたことが間違いだったことに着目してください。それは直接挿入されたので、まったく問題を生じませんでした。
- message-insinuate-rmail

  (message-insinuate-rmail) と(setq mail-user-agent 'gnus-user-agent) を~/.emacs に加えることによって、message-mode でメッセージの作成、返信および転送を行なうように Rmail を説得することができます。そこではMML の威力を堪能することができます。
- message-minibuffer-local-map

  この下の行は、メッセージを再送するときに BBDB を使えるようにします:

  ```
  (keymap-set message-minibuffer-local-map "TAB" 'bbdb-complete-name)
  ```
- gnus-posting-styles

  このような合致の様式が加わりました。

  ```
  ((header "to" "larsi.*org")
   (Organization "Somewhere, Inc."))
  ```

  下記のような古い様式は時代遅れになりましたが、まだ受け入れられます。

  ```
  (header "to" "larsi.*org"
          (Organization "Somewhere, Inc."))
  ```
- message-ignored-news-headers とmessage-ignored-mail-headers

  ‘X-Draft-From’ と‘X-Gnus-Agent-Meta-Information’ が、これら二つの変数に加えられています。それらをカスタマイズする場合に、もしかするとそれら二つのヘッダーも加える必要があります。
- Gnus は“format=flowed”(RFC 2646) パラメーターをサポートします。メッセージを作成するときに、それはuse-hard-newlines で活性化されます。format=flowed のデコードは以前からできましたが、初期の版では説明の文書がありませんでした。

- `mm-fill-flowed` オプションで“format=flowed”メッセージを流動テキストとして処理することをやめさせることができます。また、PGP 署名が埋め込まれたメッセージを送信するとき、流動テキストの処理は行なわれません。See Section “流動テキスト” in *The Emacs MIME Manual*. (Gnus 5.10.7 の新機能)
- Gnus は RFC 2298 の開封確認要求の生成をサポートします。

  これはメッセージモードの*C-c M-n* キーで呼び出されます。
- Message は Importance: ヘッダー(RFC 2156) をサポートするようになりました。

  メッセージバッファーで*C-c C-f C-i* か*C-c C-u* を使うと、可能な値が循環します。
- Gnus はニュースの Cancel Locks をサポートします。

  投稿するニュース記事に‘`Cancel-Lock`’ ヘッダーが挿入されることです。これは、記事をあなたが書いたのかどうかを確かめるために使います(キャンセルと置き換えのとき)。最初に記事を投稿するときに、Gnus はランダムなパスワード文字列を生成し、カスタムの機構を使って`~/.emacs` にセーブします。その変数は`canlock-password` と呼ばれますが、機密を気にするデータではありません。ウェブ上で canlock を公開しても、以前から彼女にできなかった何かを、誰かができるようにするものではありません。`message-insert-canlock` をカスタマイズすることによって、振る舞いを変更するができます。
- Gnus はPGP (RFC 1991/2440)、PGP/MIME (RFC 2015/3156) およびS/MIME (RFC 2630--2633) をサポートします。

  これにはS/MIME と OpenPGP が実装されている必要があります。でも追加の Lisp ライブラリーは要りません。メッセージの作成時に、いくつかのメニューと*C-c* `RET` キーの割り当てが Attachments メニューに追加されます。これはまた、`gnus-article-hide-pgp-hook` を時代遅れにしました。
- MML (Mime 作成) コマンドの接頭キーが、*M-m* から*C-c C-m* に変わりました。

  この変更によって、標準キー割り当ての`back-to-indentation` との衝突が回避されました。このコマンドもまた、メッセージモードでは役に立つのです。
- `message-forward-show-mml` のディフォルトが`best` というシンボルに変わりました。

  値`best` の振る舞いは、それがふさわしい場合はMML を表示する(すなわちMIME に変換する) ことです。変換がディジタル署名を無効にしてしまうので、署名された、または暗号化されたメッセージを転送するときはMML は使われません。
- `auto-compression-mode` が有効になっていると、添付ファイルを見るときに自動で圧縮が解かれます。
- 非-ASCII ドメイン名のサポート。

  Message は From:, To: および Cc: にある非-ASCII ドメイン名をサポートし、メッセージの送信をしようとしたときにエンコードするかどうかを尋ねます。`message-use-idna` 変数でこれを制御します。Gnus もまた、メッセージを見るときに From:, To: および Cc: にある非-ASCII ドメイン名をデコードします。これを制御するのは`gnus-use-idna` 変数です。
- Message バッファーに添付ファイルをドラッグ&ドロップすることができます。

  `mml-dnd-protocol-alist` と`mml-dnd-attach-options` を参照してください。See Section “MIME” in *The Message Manual*.

- `auto-fill-mode` が Message モードでディフォルトで有効になります。`message-fill-column` を参照してください。See Section “いろいろなメッセージ変数” in *The Message Manual*.

- バックエンドの変更
  - Gnus は RSS のニュース配送を、ニュースグループとして表示します。See Section 7.5.2 [RSS], p. 199.
  - nndoc バックエンドは、mailman のまとめ送りと exim が弾いたメッセージをサポートするようになりました。
  - Gnus は Maildir グループをサポートします。

    Gnus は新バックエンドである`nnmaildir.el` を含んでいます。See Section 7.4.13.5 [Maildir], p. 190.
  - nnml と nnfolder バックエンドは、グループ毎に印(marks) を格納するようになりました。

    これは nnml/nnfolder サーバー/グループを`~/.newsrc.eld` と切り離して、しかし印は守りつつ、バックアップすることを可能にします。さらに、例えば研究室や職場などの組織内で、(`~/.newsrc.eld` ファイルを共有すること無しに) 利用者間で記事と印を共有することをも可能にします。これは、`~/.newsrc.eld` に格納される印を、グループ毎の`.marks` ファイル(nnml 用) と*`groupname`*`.mrk` ファイル(*groupname* の名前を持つ nnfolder 用) に格納することによって動作します。nnml/nnfolder を他のマシンに引っ越しても、Gnus は`~/.newsrc.eld` にある情報の代わりに、自動的に`.marks` か`.mrk` ファイルを使います。新しいサーバー変数である`nnml-marks-is-evil` と`nnfolder-marks-is-evil` が、この機能を抑制するために使うことができます。

- 外見に関すること
  - グループと概略バッファーのメニューバー項目の名前“Misc”は“Gnus”に改名されました。
  - Message mode で“MML”と名付けられたメニュー項目は“Attachments”に改名されました。このメニューは、署名と暗号化(see Section “セキュリティー” in *The Message Manual*) のような、セキュリティーに関連したものも含んでいることに着目してください。
  - ツール・バーがグループ、概略および Message モードで GNOME のアイコンを使うように更新されました。ツール・バーはカスタマイズ可能です: *`M-x customize-apropos`* `RET` *`-tool-bar$`* から始めてください。これは Gnus 5.10.10 の新機能です。
  - ツール・バーのアイコンが正しく有効に(または無効に) なります。グループバッファーで変数`gnus-group-update-tool-bar` を参照してください。そのディフォルト値は Emacs のバージョンに依存しています。これは Gnus 5.10.9 の新機能です。

- その他の変更
  - `gnus-agent`

    Gnus エージェントは大規模な更新を経て、今やディフォルトで有効になります。そして`gnus-select-method` と`gnus-secondary-select-method` で指定されるすべての nntp と nnimap のサーバーが、ディフォルトでエージェント化されます。初期においては`gnus-select-method` のサーバーだけがディフォルトでエージェ

ント化され、エージェントはディフォルトでは有効にされませんでした。エージェントが有効にされると、今では可能ならばバックエンドに代わってエージェントのキャッシュからヘッダーが取り寄せられます。初期には、これはオフライン(unplugged) の状態でのみ行なわれていました。サーバーバッファーで`J a` と`J r` を使うことによって、サーバーの登録と削除を行なうことができます。グループバッファーから`J u` か`J s` を使って命令しない限り、Gnus は記事をエージェントのキャッシュにダウンロードしません。(`setq gnus-agent nil`) を設定することによって、エージェントが有効にされていなかった昔の振る舞いに戻すことができます。もはや`~/.gnus.el` に(`gnus-agentize`) を置いておく必要が無いことに注意してください。

- Gnus は plugged のときに、エージェントにNOV と記事を読み込みます。

  Plugged のときに記事を読む場合に、その記事がすでにエージェントにあるならば、もう一度ダウンロードすることはありません。(`setq gnus-agent-cache nil`) は旧式の動作に戻します。

- Dired の統合。

  `gnus-dired-minor-mode` (see Section 10.23 [Other modes], p. 324) は dired のバッファーで、添付ファイルを送信する、mailcap の適切な項目を使ってファイルを開く、それに mailcap の項目を使ってファイルを印刷するためのキーを割り当てます。

- ポイントの位置決めのための書法仕様(format spec) である`%C` は、`%*` に変更されました。

- `gnus-child-unplugged`

  オフラインの Gnus を子どもモードで起動する新しいコマンドです。

12.1.5.7 No Gnus

No Gnus の新しい機能:

- サポートしている Emacs のバージョン

  No Gnus は以下の Emacs のバージョンをサポートします:

  - Emacs 22 以上

- インストールに関する変更

  - No Gnus を使ったことがあるが、以前の(安定した) 版に戻してしまった人たちへの注意。

    No Gnus (このリリースにつながる不安定な Gnus の枝) を試してみたものの、安定版に戻してしまっている場合、このバージョンへアップグレードするときには注意してください。特に、`~/News/marks` ディレクトリーの内容を(もしかしたら注意深く選んで) 削除する必要があるでしょう。(訳注: 削除しないと、新しい No Gnus を再び使い始めたときに安定版を使っていた時期に更新されなかった marks ファイルが読み込まれて、`~/.newsrc.eld` ファイルの既読記事番号などを古いもので上書きしてしまいます。) 削除することによって、このリリースで nntp のフラグを保存している marks ファイルからではなく`~/.newsrc.eld` からフラグを読むようになります(訳注: そして新しい marks ファイルが作られます)。nntp marks については、次の項目でさらに詳しい情報を得ることができます。一般にダウングレードすることは安全ではありません。

- Emacs 23 から Emacs 22 に切り替えると非互換性の問題が生じます。

  Emacs 23 では、Gnus は記事のドラフトと`~/.newsrc.eld` を保存するために Emacs の新しい内部 coding system である`utf-8-emacs` を使います。これらのファイルは Emacs 22 以下では正しく読み込まれないかもしれません。Gnus を異なるバージョンの Emacs にまたがって使いたい場合は、`mm-auto-save-coding-system` を`emacs-mule` に設定すれば良いでしょう。

- Lisp ファイルがディフォルトで`.../site-lisp/gnus/` にインストールされるようになりました。

  以前は`.../site-lisp/` がディフォルトでした。加えて、新しいインストーラーは他にインストールされている、新しい Gnus より優先される Gnus を検出して警告を発します。それらを手動で取り除いても良いし、`make remove-installed-shadows` を使って削除することもできます。

- インストール先のディレクトリー名が空白文字を含むことを許容します。

- Gnus に含まれる新しいパッケージとライブラリー
  - `nnimap` の新バージョン

    `nnimap` はほとんどの互換性を保って再実装されました。新しいインターフェースについての説明は Gnus マニュアルを見てください。特に`nnimap-inbox` とクライアント側での分割方法が変化しています。

  - Gnus は Emacs Lisp SASL ライブラリーを含むようになりました。

    これによって、Emacs の中からSASL の機構を利用するために、すっきりしたAPI を使うことができます。利用者の目に見える利点は、以前は無かったDIGEST-MD5 とNTLM がサポートされるようになったことです。See Section “Emacs SASL” in *Emacs SASL*.

  - ManageSieve の接続に、ディフォルトでSASL ライブラリーを使うようになりました。

    これによる主な変更点は、サーバーがサポートしている場合にDIGEST-MD5 とNTLM をサポートするようになったことです。

  - Gnus は password.el にパスワードをキャッシュする機構を含めました。

    パスワードキャッシュはディフォルトで有効です(`password-cache` を参照)。タイムアウトは 16 秒と短いです(`password-cache-expiry` を参照)。PGG をPGP のバックエンドとして使う場合に、PGP のパスフレーズはこの機構で管理されます。ManageSieve 接続のパスワードは、利用者にそうするかどうかを尋ねてから、この機構が管理します。

  - Gnus は EasyPG を使います。

    EasyPG を使うことができる場合、Gnus はPGG の代わりにそれを使います。EasyPG は GNU Privacy Guard へのユーザーインターフェースです。See Section “EasyPG Assistant user’s manual” in *EasyPG Assistant user’s manual*. EasyPG は Emacs 23 に含まれていますが、単独でも利用可能です。

- グループモードの変更
  - `gcc-self` のようなシンボルは`gnus-parameters` と同じ、つまり他の実在の変数がそうであるような優先度決定規則を持つようになりました。ある条件に最初に合致するものではなく、最後に合致するものが採用されます。

- 入ってきたメールを一時蓄えるファイル(`Incoming*`) の古いものは、即時ではなく何日か後に消去されます。See Section 7.4.4.3 [Mail Source Customization], p. 172. (Gnus 5.10.10/No Gnus 0.8 の新機能)

- 概略モードと記事モードの変更
  - HTML をどう描画するかを決めるのは、現在はたったひとつの変数です: `mm-text-html-renderer`.
  - 粘着記事バッファーを使えるようになりました。これは、別の記事を選択したときに再利用されない記事バッファーです。See Section 4.14 [Sticky Articles], p. 80.
  - Gnus は*K H* で‘`text/html`’ 記事を WWW ブラウザーで選択的に表示することができます。See Section 4.19 [MIME Commands], p. 103.
  - 国際化ホスト名(IDNA) を、*W i* (`gnus-summary-idna-message`) を使うことによって、記事の本文中でデコードできるようになりました。この機能を使うには GNU Libidn (`https://www.gnu.org/software/libidn/`) をインストールしておく必要があります。
  - 非-ASCII グループ名の取り扱いが非常に改善されました。完全にサポートしているバックエンドは現在`nntp`, `nnml`, および`nnrss` です。エージェント、キャッシュ、それに印(marks) もそれらのバックエンドで機能します。See Section 3.17 [Non-ASCII Group Names], p. 41.
  - Gnus は dns-mode を使って text/dns として送信されたDNS マスターファイルを表示します。
  - Gnus は概略バッファーで新しい制限コマンド*/ r* (`gnus-summary-limit-to-replied`) と*/ R* (`gnus-summary-limit-to-recipient`) をサポートします。See Section 4.8 [Limiting], p. 67.
  - *Y t* (`gnus-summary-insert-ticked-articles`) を使って、サーバーからすべての可視記事を取り寄せることができるようになりました。See Section 4.27.3 [Summary Generation Commands], p. 118.
  - Gnus は概略バッファーで新しい並べ替えコマンド*C-c C-s C-t* (`gnus-summary-sort-by-recipient`) をサポートします。See Section 4.22 [Summary Sorting], p. 108.
  - S/MIME がLDAP の利用者証明書の検索に使えるようになりました。`smime-ldap-host-list` でサーバーを設定する必要があります。
  - OpenPGP ヘッダーにある URL をクリックすると、ヘッダーがダウンロードされてあなたの PGP の鍵束に取り込まれます。
  - Picon はテキストの対象物の右側に表示できるようになりました。`gnus-picon-style` を見てください。See Section 10.16.4 [Picons], p. 289.
  - ANSI SGR 制御シーケンスを*W A* で変換することができます。

    中国語のニュース階層のグループにおいて、記事をハイライト表示するためにANSI シーケンスが使われます(`gnus-article-treat-ansi-sequences`)。
  - Gnus は記事に“MIME-Version”ヘッダーがなくても記事を MIME デコードします。このために`gnus-article-loose-mime` のディフォルト値が変更されました。
  - `gnus-decay-scores` をスコアファイルに合致する正規表現にできます。例えば‘`\\.ADAPT\\'`’ に設定すると、適応スコアファイルだけが減衰されるようになります。See Section 8.16 [Score Decays], p. 255.

- gnus-ignored-from-addresses を使う場合に、概略行においてTo とNewsgroup ヘッダーに相当する場所の最初に表示する文字列を、gnus-summary-to-prefix およびgnus-summary-newsgroup-prefix でカスタマイズすることができます。See Section 4.1.2 [To From Newsgroups], p. 50.
- MIME パートを外部にある本体で置き換えることができます。gnus-mime-replace-part とgnus-article-replace-part を見てください。See Section 4.19 [MIME Commands], p. 103, Section 5.2 [Using MIME], p. 126.
- mm-fill-flowed オプションで format=flowed なメッセージの取り扱いを無効にすることができます。また、PGP の署名が埋め込まれたメッセージを送信するときに、flowed text は無効にされます。See Section “流動テキスト” in *The Emacs MIME Manual*. (Gnus 5.10.7 の新機能)
- 新コマンド*S W* (gnus-article-wide-reply-with-original) は記事バッファーで広い返答をするためのもので、もし領域が設定されたテキストがあれば*R* コマンド(gnus-article-reply-with-original) と同様にそれを引用します。以前は*R* コマンドに広い返答をさせるために使われた接頭引数が、今では受け付けられないことに注意してください。See Section 5.5 [Article Keymap], p. 132.
- 記事バッファーで使う新コマンド*C-h b* (gnus-article-describe-bindings) は、記事コマンドだけでなく記事バッファーから利用することができる真の簡略コマンドも表示します。

- Message モードの変更
  - すべての送信済みメッセージのディフォルトの保存先が、月ごとの nnfolder アーカイブになりました。
  - Gnus は“hashcash”client puzzle anti-spam の機構をサポートします。(setq message-generate-hashcash t) で有効になります。See [Hashcash], p. ⟨undefined⟩.
  - メッセージバッファーに添付ファイルをドラッグ＆ドロップできます。mml-dnd-protocol-alist とmml-dnd-attach-options を見てください。See Section “MIME” in *Message Manual*.
  - message-yank-empty-prefix オプションで、引用文の空行にどんな引用符を付けるかを制御することができます。See Section “挿入するための変数” in *Message Manual*.
  - Gnus はメッセージバッファーでヘッダーを隠すために、それら以外の部分だけが見えるようにバッファーを狭めます。References はディフォルトで表示されません。すべてのヘッダーが見えるようにするには(setq message-hidden-headers nil) としてください。See Section “メッセージヘッダー” in *Message Manual*.
  - You can highlight different levels of citations like in the article buffer. See gnus-message-highlight-citation.
  - 記事バッファーでできるのと同様に、引用された文のレベルの違いに応じたハイライトを行うことができます。gnus-message-highlight-citation を参照してください。
  - Message モードではauto-fill-mode がディフォルトで ON になります。message-fill-column を参照してください。See Section “メッセージヘッダー” in *Message Manual*.

    - 署名ファイルを`message-signature-directory` 変数で指定するディレクトリーに置くことができます。
    - "○○さん writes:" のような行の形式を、オプション`message-citation-line-format` で指定することができます。これを使うには`message-citation-line-function` 変数を`message-insert-formatted-citation-line` に設定する必要があります。
- サーバー閲覧モードの変更
    - Gnus の洗練された購読方法がサーバー閲覧バッファーで利用可能です。
      変数`gnus-browse-subscribe-newsgroup-method` を使ってください。
- バックエンドの変更
    - nntp バックエンドは記事の印を`~/News/marks` に保管します
      そのディレクトリーは`nntp-marks-directory` という(カスタマイズ可能な) 変数で変更することができます。また、nntp で印を使うことを`nntp-marks-is-evil` という変数(バックエンド変数) で無効にすることができます。印を使うことの利点は、`~/News/marks` を別のホストにインストールされた Gnus にも(rsync, scp などを使って) コピーすることによって、どの記事を読んでどの記事に印を付けたかの情報を、そこでも維持できることです(訳注: 同じ nntp サーバーに接続する場合に限ります)。`~/News/marks` のデータは`~/.newsrc.eld` にある同じデータより優先されます。
    - RSS の購読情報をOPML のファイルから取り込み、または書き出すことができるようになりました。See Section 7.5.2 [RSS], p. 199.
    - IMAP の identity (RFC 2971) をサポートします。
      ディフォルトでは Gnus はそれ自身に関する情報を送信しませんが、`nnimap-id` 変数を使ってそれをカスタマイズすることができます。
    - `nnrss` バックエンドは多言語テキストをサポートします。`nnrss` グループでは非-ASCII 文字列を使ったグループ名もサポートされます。See Section 7.5.2 [RSS], p. 199.
    - POP3 によるメールの取得において、SSL/TLS と StartTLS をサポートするようになりました。
    - nnml バックエンドではメッセージを圧縮するために`gzip` 以外のプログラムも使うことができます。See Section 7.4.13.3 [Mail Spool], p. 188.
    - nnml バックエンドではグループを圧縮することができます。
      関数`gnus-group-compact-group` (グループバッファーの*`G z`* キー) および`gnus-server-compact-server` (サーバーバッファーの*`z`* キー) で呼び出すこの機能は、グループのすべての記事の番号を 1 から順に振り直して、すきまを取り除きます。その結果として、正しい全記事数を得ることができます(再びメッセージが削除されるまでは)。
- 外見
    - ツール・バーが GNOME のアイコンを使うように更新されました。ツール・バーをカスタマイズすることもできます: *`M-x customize-apropos`* `RET` *`-tool-bar$`* で始めてください。
    - ツール・バーのアイコンがグループバッファーで正しく活性化(または不活性化) されるようになりました。変数`gnus-group-update-tool-bar` を参照してください。そのディフォルト値は Emacs の版に依存します。

- その他の変更
  - サーバーバッファーで外部グループのための select-method を変更すると、すぐにそのサーバーを使うグループの購読に反映されるようになりました。例えば`nntp-via-address` を‘`foo.example.com`’ から‘`bar.example.com`’ に変更すると、Gnus は次回から中間ホスト‘`bar.example.com`’ を経由してニュースサーバーに接続するようになります。
  - `W e` で`all.SCORE` ファイルをグループバッファーから編集することができます。
  - `gnus-mark-copied-or-moved-articles-as-expirable` を`nil` ではない値に設定することによって、読み終わった記事を auto-expire が有効になっているグループにコピーするか移動するとき、それらに期限切れ消去可能の印を自動的に付けるようにすることができます。ディフォルト値は`nil` で、記事のコピーと移動の動作は従来通りです。つまり、記事を auto-expire が有効になっていないグループにコピーするか移動するときに期限切れ消去可能の印が削除される以外は、印は変化しません。See Section 7.4.9 [Expiring Mail], p. 181.
  - NoCeM のサポートは削除されました。
  - Carpal mode は削除されました。

12.1.5.8 Ma Gnus

本当の綴りは「真 Gnus」です。Ma Gnus は 2012 年に始まった開発版の Gnus のコード名でした。最近では Gnus は Emacs と一緒にのみリリースされています。

真 Gnus の新しい機能:

- インストールに関する変更
  - インストールされる Lisp のソースファイルと info ファイルはディフォルトで gzip によって圧縮されます。

    もしそれらのファイルが圧縮されることを望まないのであれば、configure オプションの‘`--without-compress-install`’ を使ってください。インストールされるディレクトリーにおいてコンパイルされた elc 版を持たない Lisp のソースファイルは圧縮されません。
- 概略モードと記事モードの変更
  - 添付ファイル(もしあれば) のためのMIME パート・ボタンを、記事ボディーの底に加えて記事のヘッダーの最後にディフォルトで表示します。そのため何度も記事をスクロールしなくても、すぐにそれらを見つけることができます。See Section 4.19 [MIME Commands], p. 103.
- Message モードの変更および関連する Gnus の機能
  - 新しい hook である`gnus-gcc-pre-body-encode-hook` および`gnus-gcc-post-body-encode-hook` が、送信したメッセージを`Gcc` コピーする際、そのボディーをエンコードする前後に実行されます。See Section 6.5 [Archived Messages], p. 137.

    最近の変更については、Emacs の`NEWS` ファイルを参照してください。

12.2 マニュアル

このマニュアルは TeXinfo ファイルから作成され、それからtexi2dvi を通して、あなたの手元にあるものになりました。

以下の習慣が用いられました:

1. これは‘**文字列**’ です。
2. これは**キー打鍵** です。
3. これは**ファイル** です。
4. これは**シンボル** です。

ですから、私が「flargnoze を‘yes’ に設定する」と言ったときは、次のような意味です:

```
(setq flargnoze "yes")
```

もし、私が「flargnoze をyes に設定する」と言ったときは、次のような意味です:

```
(setq flumphel 'yes)
```

‘yes’ とyes は二つのまったく 違ったものです---絶対に混同しないでください。

12.3 マニュアルを書く

おそらく、たいていのマニュアルは事後に書かれていると思います。つまり、すでにあるプログラムを文書化しているということです。このマニュアルはそういう方法で書かれていません。何かを実装するときは、何かをそのままマニュアルの一節に書きます。それから機能の説明が難しいことを発見して、それがどのようにあるべきであるかを書き、次には実装を変更します。文書とコードを書くことは協調して行なわれていきます。

もちろん、これはこのマニュアルには流れ構造がほとんど無いか、あっても少しだということを意味します。Gnus の完全にすべてのことが説明されていますが、あなたが探している場所ではないということがよくあります。これはリファレンスマニュアルであって、Gnus を始めるための手引きではありません。

それはこのリファレンスマニュアルを元にして書かれた、まったく違った本になるでしょう。とても違ったものになるはずです。

12.4 用語

ニュース*(news)*
: これは、あなたがそれを読むために使うことになっているもの、つまり、それというのがニュースです。ニュースは一般的には近くのNNTP サーバーから取得され、一般的にはすべての人が公に利用することができます。もしニュースを投稿すると、あなたがまさに書いたものを全世界の人たちが読むことになるでしょう。そして、みんながいたずらっぽくクスクス笑うでしょう。あなたの知らないところで。

メール*(mail)*
: あなたに個人的に配送されるすべてのものがメールです。いくつかのニュース/ メールリーダー(Gnus のような) はメールとニュースの区別を曖昧にしますが、違いがあります。メールは私的です。ニュースは公的です。メールを送信することは投稿ではなく、返信はフォローアップではありません。

返信*(reply)*
: あなたが読んでいるものを書いた人にメールを送ることです。

フォローアップ*(follow up)*
: あなたが読んでいる記事に応答して、現在のニュースグループに記事を投稿することです。

バックエンド*(back end)*
: Gnus はメールとニュースがほとんど同じだとみなします。本当に。違いは実際の記事にどのようにアクセスするかだけです。メールメッセージはローカルディスクのファイルから読めるのに対して、ニュース記事は一般にNNTP プロトコルで取得します。Gnus の内部構造は、それらのために「フロントエンド」と数々の「バックエンド」から成り立っています。内部的には、あなたがグループに入る(そう、`RET` をたたく) と、それによって Gnus のフロントエンドの機能を呼び出します。そうするとフロントエンドは、バックエンドに「foo グループの記事のリストをくれ」とか「4711 番の記事を見せてよ」と「話す」のです。

  そういうわけで、バックエンドは主にプロトコルか、ファイルの形式とディレクトリーの配置のどちらかを定義します。前者は`nntp` バックエンドがNNTP でニュースにアクセスしたり、`nnimap` バックエンドがIMAP でメールにアクセスすることを指します。また、後者は`nnspool` バックエンドが共通の「スプールディレクトリー」形式にアクセスしたり、それととてもよく似たファイルの形式とディレクトリーの配置を介して`nnml` バックエンドがメールにアクセスすることを指します。

  Gnus は基礎的なメディアを扱いません。言わばこれは、すべてバックエンドによって行なわれるということです。バックエンドは記事にアクセスするための機能の集成です。

  しかし、「バックエンド」という用語は「サーバー」と言った方がふさわしい場面でときどき使われます。そして同じことを指すことができる「選択方法」(select method) という用語があります。かように Gnus の用語はとても混乱しています。

基本*(native)*
: Gnus はいつも一つの方法(とバックエンド) を、ニュースを得るための「基本」もしくはディフォルトの手段として使います。「基本」の選択方法で取得するグループは‘gnu.emacs.gnus’ のような名前になります。

外部*(foreign)*
: 同時に任意の数の外部グループをアクセスできる状態にすることもできます。これらはニュースを取得するための、基本ではなく、二次のでもないバックエンドを使うグループです。外部グループは‘nntp+news.gmane.io:gmane.emacs.gnus.devel’ のような名前になります。

二次の*(secondary)*
: 二次のバックエンドは、基本と外部の間くらいに位置するバックエンドですが、ほとんど基本と同じように動作するものです。しかしそれらも‘nntp+news.gmane.io:gmane.emacs.gnus.devel’ のような名前になります。

記事*(article)*
: ニュースとして投稿されたメッセージです。

メールメッセージ*(mail message)*
: メールで送られたメッセージです。

メッセージ
: メールメッセージもしくはニュース記事です。

ヘッド*(head)*
: メッセージの最上部で、管理情報(等) が入れられているところです。

本文*(body)*
: 記事の残りの部分です。ヘッドに無いものはすべて本文です。

ヘッダー*(header)*
: 記事のヘッドの行です。

ヘッダー群*(headers)*
: そのような行の集合もしくは、ヘッドの集合です。もしくは、NOV 行の集合です。

NOV
: NOV は News OverView の略です。それらはニュースサーバーのヘッダーの一つの形式で、記事の簡潔なヘッダーの情報を含むデータを提供するために、サーバーが自ら作ります。nntp バックエンドでは Gnus はNNTP サーバーが作るものを使いますが、いくつかのサーバー(特にnnml) のためには Gnus 自身が作ります。

  グループに入ると、Gnus はグループのすべての未読記事のヘッダーをバックエンドに要求します。ほとんどのサーバーは News OverView 様式をサポートしています。それは標準のHEAD 様式よりコンパクトで、とても速く、読んで解析することができます。

  NOV データは一つ以上のテキスト行(see Section “Motion by Text Lines” in *The Emacs Lisp Reference Manual*) から成り、それぞれの行は一つの記事のヘッダー情報を持っています。ヘッダー情報はタブで区切られた一連のヘッダーの内容で、記事番号、表題、著者、日付、Message-ID、References などを含んでいます。

それらのデータは Gnus が概略行をすばやく生成することを可能にします。しかしサーバーがNOV をサポートしていなかったり、故意にまたはある理由でそれを使わないようにしてしまうと、Gnus はそれぞれの記事のヘッダーを一つずつ解析することによってヘッダー情報を生成しようとするでしょう。それには時間がかかります。したがって、サーバーが間違ったNOV データを作ることがわかっている場合以外は、nn*-nov-is-evil (see Section 12.5.1 [Slow/Expensive Connection], p. 359) を`nil` ではない値にすることは、通常は良い考えではありません。

レベル*(level)*
: それぞれのグループは何らかの「レベル」(1--9) で購読されています。低いレベルのものは高いレベルのものより「より」購読されています。実際のところ、レベル 1--5 のグループは「購読」; 6--7 は「未購読」; 8 は「ゾンビ」; 9 は「切られた」(killed) と見なされます。グループの一覧を表示したり、新しい記事を走査する命令は、すべて数値接頭引数を「動作レベル」として使います。

切られたグループ*(killed groups)*
: 切られたグループの情報は保存されたり更新されたりしないので、切られたグループを扱うのは購読されているグループよりも簡単です。

ゾンビグループ*(zombie groups)*
: ほとんど切られたグループと同じで、それより少し死んでいるだけです。

アクティブファイル*(active file)*
: ニュースサーバーは、どの記事を持っているかとどのグループが存在するかを記録しておかなければなりません。アクティブファイルに格納されるすべてのこの情報は、あなたが推測するように比較的大きいです。

偽グループ*(bogus groups)*
: `.newsrc` ファイルに存在するけれどもサーバーが知らないグループ(すなわち、それはアクティブファイルにありません) は偽グループ です。おそらくそのグループは(もはや) 存在していないのでしょう。

活性化*(activating)*
: サーバーにグループの情報を尋ねて未読記事の数を演算する行為は「グループを活性化(activate) する」と呼ばれています。活性化されていないグループは、グループバッファーに‘`*`’ とともに一覧表示されます。

スプール*(spool)*
: ニュースサーバーは何らかのやり方で記事をローカルに保存します。ある古い流儀の保存方法は、単に記事毎に一つのファイルを持つことです。それは伝統的なスプール(traditional spool) と呼ばれます。

サーバー*(server)*
: 接続して、ニュース(もしくはメール) を取得することができるマシンです。

選択方法*(select method)*
: バックエンドと、サーバーおよび仮想サーバーの設定を指定する構造です。

仮想サーバー*(virtual server)*
: 名前が付けられていて、その名前で指定することができる選択方法です。選択方法は(物理的な) サーバーに関するすべてを定義するので、ものごとを全体として捉えるのは仮想サーバーになります。

洗濯*(washing)*
: バッファーを持ってきて、何らかの種類のフィルターにかけることです。結果は(たいてい) 元のものよりもよりきれいで喜ばしいものになるでしょう。

一時グループ*(ephemeral groups)*
: たいていのグループはどの記事を読んだかのデータを保存します。「一時」グループはデータが溜められないグループです---グループを出ると、それは天空のかなたに消え去ります。

固定グループ*(solid groups)*
: これは一時グループの反対です。グループバッファーに一覧表示されているすべてのグループは固定グループです。

まばら記事*(sparse articles)*
: `gnus-build-sparse-threads` が有効にされているときに、それらは概略バッファーに表示される(存在しない) 記事のための場所取りです。

スレッド化*(threading)*
: 応答の記事を、それが応答した元記事の直後に置くことです---階層的な流儀で。

根*(root)*
: スレッドの最初の記事が根です。それはスレッドのすべての記事の祖先です。

親*(parent)*
: 応答が得られた記事です。

子*(child)*
: それとは別の記事、すなわち親に応答する記事です。

まとめ送り*(digest)*
: 複数のメッセージを一つのファイルに集めたものです。最も一般的なまとめ送りの様式は、RFC1153 で規定されています。

分割*(splitting)*
: ある規則によってメールを区分けする行為です。ときどき間違ってメールの濾過(filtering) と呼ばれます。

12.5 カスタマイズ

すべての変数は、このマニュアルのどこか他のところで適切に説明されています。この章は、非常に良くある状況でどのように Gnus をカスタマイズすれば良いかを調べるための、総合的な案内になるように作られています。

12.5.1 遅くて高価な接続

Emacs をローカルのマシンで実行していて、非常に細いひもの向こうのマシンからニュースを取り寄せているとしたら、Gnus がサーバーから取って来なければならないデータの総量を減らしたくなるでしょうね。

`gnus-read-active-file`
: これを`nil` にしてください。これは Gnus がサーバーにアクティブファイル全体を要求することを禁止します。このファイルはしばしば非常に大きいです。さらに、Gnus が不意にアクティブファイルをとにかく取り寄せようと決意しないように、`gnus-check-new-newsgroups` および`gnus-check-bogus-newsgroups` も`nil` に設定する必要があります。

`gnus-nov-is-evil`
: 通常これは常に `nil` にしておかなければなりません(それがディフォルトです)。例えばもし`nntp` バックエンドでNOV (see Section 12.4 [Terminology], p. 355) を使わないようにしたい(see Section 4.29 [Crosspost Handling], p. 121) のであれば、これを設定する代わりに`nntp-nov-is-evil` を`nil` 以外の値にしてください。しかし Gnus はNNTP サーバーがNOV をサポートするかどうかを自分で調べるので、通常`nntp-nov-is-evil` を設定する必要はありません。とにかく Gnus がNOV を利用しないようにすると、NNTP サーバーから記事のヘッダーを掴み取ってくる動作が、あまり速くなりません。

: 他のバックエンドのための変数として`nndiary-nov-is-evil`, `nndir-nov-is-evil`, `nnfolder-nov-is-evil`, `nnml-nov-is-evil` および`nnspool-nov-is-evil` があります。`gnus-nov-is-evil` に`nil` 以外の値を与えると、それらすべての変数を上書きしてしまうことに注意してください。

12.5.2 遅いターミナル接続

Emacs と Gnus を実行しているシステムに、家のコンピューターをダイアルアップで接続しているとしましょう。モデムが遅い場合は、電線を伝って送られているデータの総量を(可能な限り) 減らしたくなるでしょう。

`gnus-auto-center-summary`
: Gnus が概略バッファーをリセンターする(訳注: 現在の記事が真ん中に表示されるようにする) するために、これを`nil` に設定してください。これが`vertical` だったら、垂直方向のリセンターだけをします。`nil` でも`vertical` でも無ければ、水平方向と垂直方向の両方でリセンターを行ないます。

`gnus-visible-headers`
: 記事に含まれるヘッダーを最小限に減らします。実際のところ、それらが無くてもすべて間に合わせることができます---たいていの役に立つデータは、とにかく概略バッファーにありますから。この変数を‘`^NEVVVVER`’ や‘`From:`’ や、何でも必要になりそうなものに設定してください。

: 利用できるすべての「隠す」機能を有効にするために、以下を使ってください:

```
(setq gnus-treat-hide-headers 'head
      gnus-treat-hide-signature t
      gnus-treat-hide-citation t)
```

gnus-use-full-window
: これをnil に設定することによって、すべてのウィンドウを小さくすることができます。これは総じてそんなに減らしませんが、この記事は何が何でも読みたくなかったんだと決心する前に、それを少ししか見ないで済みます。

gnus-thread-hide-subtree
: これをnil ではない値にしておくと、すべての概略バッファーのスレッド(の親以外) は、初めは隠されているようになります。

gnus-updated-mode-lines
: これをnil にすると、Gnus はバッファーのモード行に情報を表示しないので、いくらか時間を節約できるでしょう。

12.5.3 少ないディスク容量

起動ファイルはやや大きくなり得るので、空き容量が少なくなってきているときは、そのサイズを少し小さくする必要があるでしょう。

gnus-save-newsrc-file
: これをnil にすると、Gnus は決して.newsrc を保存しません—.newsrc.eld だけを保存します。これは Gnus 以外のニュースリーダーが使えなくなることを意味します。この変数はディフォルトでt です。

gnus-read-newsrc-file
: これがnil であれば、Gnus は.newsrc を決して読みません—.newsrc.eld だけを読みます。これは Gnus 以外のニュースリーダーが使えなくなることを意味します。この変数はディフォルトではt です。

gnus-save-killed-list
: これがnil であると、Gnus は死んだグループのリストを保存しません。この変数をnil に設定したときは、gnus-check-new-newsgroups をask-server に、gnus-check-bogus-newsgroups をnil に設定するべきでしょう。この変数はディフォルトでt です。

12.5.4 遅いマシン

遅いマシンを持っているか、または本当は単に忍耐力が無いだけでも、Gnus を速く走らせるためにできることが少しあります。

起動を速くするためにgnus-check-new-newsgroups およびgnus-check-bogus-newsgroups をnil に設定してください。概略バッファーに入ることと抜けることを速くするために、gnus-show-threads とgnus-use-cross-reference、それにgnus-nov-is-evil をnil に設定してください。Section 12.5.1 [Slow/Expensive Connection], p. 359, も参照してください。

12.6 問題解決

Gnus は箱から出してすぐに非常に よく動作します---どんな問題が起きることも想像できません、本当に。

オッホン。

1. コンピューターの電源が入っていることを確かめてください。
2. FAQ と入門書を読むために、ヘルプグループ(グループバッファーで*G h*) を読んでください。
3. Gnus は多くの再帰構造で動作しているので、何か極端な(そして非常に希な) 場合には、Gnus は再帰を「あまりに深く」降りすぎてしまい、Emacs があなたにビープ音を鳴らすことがあります。もしこれが起こったなら、`max-lisp-eval-depth` を 2000 かそこいらの値に設定してください。

もし他のすべてが失敗したなら、バグとして問題を報告してください。

もし Gnus のバグを見つけたなら、*M-x gnus-bug* 命令で報告することができます。*M-x toggle-debug-on-error* とタイプして、私にバックトレースを送ってください。私はバグを修正しようとしますが、あなたがバグを再現させる方法を正確に書いてくれないと、それを修正することができません。

バグ報告では、詳細すぎることは決してありません。バグ報告をするときは、いつも*M-x gnus-bug* 命令を使ってください。それを使うたびに 10KB のメールができてしまっても、そしてあなたの環境のことを以前私に 500 回送ったことがあったとしてもです。

私がどんなたぐいの記憶も持っていないことを、覚えておくことも重要です。もしあなたがバグ報告を送ると、私は返答を送ります。その後で、あなたが「いや、そうじゃない! このうすのろめっ!」とだけ送り返してきても、私はあなたが何について私を侮辱しているかがわかりません。常に、すべてを説明し過ぎてください。それは私たちすべてにとって、もっとやり易くなります---もし私が必要なすべての情報を得られなかったら、私はあなたにメールを送ってさらなる情報を求め、その結果すべてがより多くの時間を費やすことになります。

もしあなたの直面している問題が非常に視覚的で、それをうまく説明できない場合は、Emacs のウィンドウをファイルにコピーして(例えば、*xwd* で)、それをどこか手の届くどこかにおいて、その画像の URL をバグ報告に含めてください。

もしあなたがバグの修正や改善のためのパッチを寄稿してくださるのでしたら、すみませんがそのパッチは‘`diff -u`’ で作ってください。

問題を報告する前にもっとデバッグしたければ、あなた自身で問題を解決してパッチを送るために edebug を使うことができるでしょう。Lisp コードのデバッグについては ELisp マニュアル(see Section “Debugging Lisp Programs” in *The GNU Emacs Lisp Reference Manual*) に書かれています。edebug を始めるには、もし*c* を押したときにある変な振舞いが発見されるならば、第一歩は*C-h k c* をタイプし、ドキュメンテーション・バッファー中でハイパーリンクをクリックして、その関数定義を参照することです。そしてその関数名の場所で*M-x edebug-defun* `RET` をタイプして Gnus に戻り、そのコードを呼び出すために*c* を押してください。Lisp バッファーでは、`SPC` でシングルステップ動作、*M-:* で式を評価、*C-h v* で変数を検査、*q* で実行を中断、あるいは*c* か*g* で実行を再開することができます。

ときどき、直接に Emacs Lisp のエラーを起こさないものの、Gnus が非常に遅くなるために明らかになる問題があります。そんな場合には*M-x toggle-debug-on-quit* を使って、

遅くなったときに*C-g* を押し、しかる後にバックトレースを解析してください(その手続きを繰り返すことは、真の問題領域を分離するのに役立ちます)。

より上等なやり方は elisp プロファイラー(訳注: プログラムの実行時の動きを分析する道具) ELP を使うことです。プロファイラーについてはどこか他の場所で完全に文書化されているはずですが、それを始めるために必要な手順を少々書いておきましょう。第一に、プロファイルしてみたい Gnus の部分を計測するための設定を、例えば*M-x elp-instrument-package* RET *gnus* や*M-x elp-instrument-package* RET *message* で行なってください。そして、遅い動作を行なわせてから*M-x elp-results* を押しましょう。すると、どの動作が時間を食っているかを見て、それらをさらにまたデバッグすることができます。動作全体が、プロファイラーの出力の中で最も遅い関数で費やされた時間よりはるかに長くかかるのは、たぶん Gnus の間違っている部分をプロファイルしたせいでしょう。プロファイルの統計をリセットするには*M-x elp-reset-all* を使ってください。*M-x elp-restore-all* はプロファイルする動作を取り除くことになっていますが、Gnus によって複雑にされかつ動的なコード生成の影響を受けるため、それは必ずしも完全には動作しないかもしれません。

もし手助けが欲しいだけであれば、'gnu.emacs.gnus' で尋ねるのが良いでしょう。私はあまり役に立ちません。また、ding メーリングリスト—ding@gnus.org で尋ねることもできます。購読するにはding-request@gnus.org にメールを送ってください。

12.7 Gnus リファレンスガイド

誰かが Gnus でできる何か粋なものに知恵を働かせて、その粋なものを書いてもくれることが私の願いです。それを促進するためには、Gnus の内部動作を説明するのが良いだろうと思いました。それに、さほど内部ではない動作をいくつかと、私が今やっていることも。

プログラムの内部の仕様が変更されることはない、などと思ってはいけませんが、Gnus とそのバックエンドの間のインターフェース(これは完全に記述されています) や、スコアファイルの形式(同じく)、データ構造(これは他のものほどには変更されないでしょう)、それに一般的な操作のメソッドを、(細部にわたって) 定義していきます。

12.7.1 Gnus の有用な関数

フックなどから実行される小さな関数を書くときは、Gnus の内部関数や変数にアクセスすることが絶対に必要です。以下が最もよく使われるものの一覧です。

`gnus-newsgroup-name`
: この変数は現在のニュースグループの名前を持っています。

`gnus-find-method-for-group`
: *group* の選択方法を返す関数です。

`gnus-group-real-name`
: 正規の(接頭語付きの) Gnus グループ名を受け取って、接頭語が無い名前を返します。

`gnus-group-prefixed-name`
: 接頭語が無いグループ名と選択方法を受け取って、正規の(接頭語付きの) Gnus グループ名を返します。

`gnus-get-info`
: *group* のグループ情報のリストを返します(see Section 12.7.6 [Group Info], p. 380)。

`gnus-group-unread`
: *group* の未読記事の数か、それが分からない場合は`t` を返します。

`gnus-active`
: *group* に関するアクティブファイルの項目(最小と最大の記事番号) を返します。

`gnus-set-active`
: *group* に関するアクティブファイルの項目を設定します。

`gnus-add-current-to-buffer-list`
: Gnus を終了するときに消去するバッファーのリストに、現在のバッファーを追加します。

`gnus-continuum-version`
: 引数として Gnus のバージョン文字列を受け取って、浮動小数点の数値を返します。古いバージョンは必ず新しいバージョンよりも小さい数になります。

`gnus-group-read-only-p`
: *group* が読み出し専用かどうかを示します。

`gnus-news-group-p`
group がニュースバックエンドかどうかを示します。

`gnus-ephemeral-group-p`
group が一時ニュースグループかどうかを示します。

`gnus-server-to-method`
server に対応している選択方法を返します。

`gnus-server-equal`
二つの仮想サーバーが実質的に同一かどうかを示します。例えば二つの仮想サーバーが異なる順序で並んでいるサーバー・パラメーターを持っていても、この関数はそれらを同一であると見なします。

`gnus-group-native-p`
group が基本グループかどうかを示します。

`gnus-group-secondary-p`
group が二次グループかどうかを示します。

`gnus-group-foreign-p`
group が外部グループかどうかを示します。

`gnus-group-find-parameter`
group のパラメーターのリストを返します(see Section 3.10 [Group Parameters], p. 23)。二つ目の引数を与えると、*group* 用のそのパラメーターの値を返します。

`gnus-group-set-parameter`
三つの引数*group*, *parameter*, *value* を与えて、パラメーターとして設定します。

`gnus-narrow-to-body`
現在のバッファーを、記事の本文に狭めます。

`gnus-check-backend-function`
二つの引数*function* と*group* を取ります。*group* のバックエンドが*function* をサポートしているなら、`nil` ではない値を返します。

```
(gnus-check-backend-function "request-scan" "nnml:misc")
⇒ t
```

`gnus-read-method`
利用者に選択方法を入力してもらう関数です。

12.7.2 バックエンドインターフェース

Gnus はNNTP やスプール、メール、仮想グループについては何も知りません。ただ仮想サーバー*virtual servers* と対話する方法を知っているだけです。仮想サーバーはバックエンド*back end* といくつかのバックエンド変数*back end variables* からなります。前者の例としては`nntp`, `nnspool`, `nnmbox` などがあります。後者の例としては`nntp-port-number` や`nnmbox-directory` があります。

Gnus がバックエンド---例えば`nntp`---に何かの情報を要求するとき、通常は関数の引数として仮想サーバー名を含めます。(無い場合は、バックエンドは「現在の」仮想サーバー

を使うべきです。) 例えばnntp-request-list は、その唯一の(省略可能な) 引数として仮想サーバーを使います。もしこの仮想サーバーとの接続が開かれていないと、この関数の実行は失敗するはずです。

仮想サーバー名は、物理的なサーバー名とは何の関係も無いことに注意してください。例を挙げましょう:

```
(nntp "odd-one"
      (nntp-address "ifi.uio.no")
      (nntp-port-number 4324))
```

ここで物理サーバー名は‘ifi.uio.no’ であるのに対して、仮想サーバー名は‘odd-one’です。

バックエンドは複数の仮想サーバーを切り替えることができなければなりません。標準のバックエンドは、必要なときに仮想サーバーの環境を取り出し・押し込みを行なう連想リストを保持することによって、これを実現しています。

インターフェース関数には二つのグループがあります。必ず存在しなければならない必須関数*required functions* と、呼び出す前にそれが存在するかどうかを常に Gnus が確認する任意関数*optional functions* です。

これらすべての関数は、その戻り値のデータをnntp-server-buffer (*nntpd*) バッファーに返すことが求められます。これはちょっと不運な名前付けですが、これで我慢しなければなりません。私が結果のデータ*resulting data* と言ったときは、そのバッファーの中のデータを指しています。戻り値*return value* と言ったときは、関数呼び出しによって返される関数の値のことを言っています。関数が失敗したときは、戻り値としてnil を返さなくてはいけません。

バックエンドにはサーバー型*server-forming* のバックエンドと呼ばれるものがあり、またそう呼ばれないものもあります。後者は一般には、同時には一つのグループだけしか操作しないバックエンドで、「サーバー」の概念がありません。このサーバーとは、グループを持ち、そのグループの情報を配送するもので、それ以上のものではありません。

Gnus はグループ名と記事番号によって、それぞれのメッセージを特定します。それら記事番号に関するちょっとした説明をすることは有益かもしれません。まず第一に、その数値は正の整数です。第二に Gnus を混乱させることなく古い記事番号を、後で「再使用」することは普通はできません。すなわち、もしあるグループにかつて 42 番の記事があったとしたら、別の記事がその番号を持つことができないか、または Gnus が激しく混乱してしまうということです。[1] 第三に、記事番号はそのグループでの到着順になっていなければならないことです。メッセージの日付も、必ず到着順になっているわけではありませんが。

すでに前の節で、記事番号は一回使われただけで役目を終わらなければならない「厳しい」制限について説明しました。しかし、記事番号の並びに抜けがあると Gnus はとても混乱してしまうので、連続した通し 番号を付けることが有用なのかもしれません。ただし「再使用不可」の制限があるので、完全に番号の抜けを回避できるとは限りません。また、可能な限り記事番号を 1 から始めることは、番号を使い切ってしまうことを避けるために役立ちます。

慣例として、バックエンドはnn なんたら と名付けられますが、Gnus にはnnheader.el、nnmail.el およびnnoo.el のように、いくつかのバックエンドではないnn かんたら があることに注意してください。

[1] nnchoke-request-update-info 関数の説明を見てください。See Section 12.7.2.2 [Optional Back End Functions], p. 369.

ここでの例と定義では、想像上のバックエンドnnchoke を引き合いに出すことにします。

12.7.2.1 必須バックエンド関数

(nnchoke-retrieve-headers *articles* &optional *group server fetch-old*)

articles は記事番号の範囲か、Message-ID のリストのどちらかです。現在のバックエンドは、どちらも完全にサポートしているわけではありません---記事番号のひと続き(リスト) だけで、多くのバックエンドはMessage-ID による取得をサポートしていません。でも、それらは両方サポートすることに努めるべきです。

結果のデータは HEADs かNOV 行のいずれかであるべきで、戻り値はこれを反映したheaders かnov のどちらかでなければなりません。これは今後、HEADsとNOV 行が混在するvarious に拡張されるかもしれませんが、現在の Gnus ではサポートされていません。

fetch-old がnil ではなかったら、ある意味での「余分なヘッダー」を取得しようとします。これは通常、articles の中の最小番号の記事よりも小さい番号を持っている(最大で) *fetch-old* 個の記事と、articles の中で欠番になっている記事の、余分なヘッダーを取得します。もしバックエンドがこの要求に従うことを煩わしいと思った場合には、このパラメーターの存在は無視されることもあります。この値がnil でも数値でもなかったら、最大限の取得を行ないます。

これが HEAD の例です:

```
221 1056 Article retrieved.
Path: ifi.uio.no!sturles
From: sturles@ifi.uio.no (Sturle Sunde)
Newsgroups: ifi.discussion
Subject: Re: Something very droll
Date: 27 Oct 1994 14:02:57 +0100
Organization: Dept. of Informatics, University of Oslo, Norway
Lines: 26
Message-ID: <38o8e1$a0o@holmenkollen.ifi.uio.no>
References: <38jdmq$4qu@visbur.ifi.uio.no>
NNTP-Posting-Host: holmenkollen.ifi.uio.no
.
```

そういうわけで、headers という戻り値は、データバッファーにその要素数と同じ個数のヘッダーがあることを暗示します。

これがそういうバッファーの BNF 定義です:

```
headers        = *head
head           = error / valid-head
error-message  = [ "4" / "5" ] 2number " " <error message> eol
valid-head     = valid-message *header "." eol
valid-message  = "221 " <number> " Article retrieved." eol
header         = <text> eol
```

(ここで使っている拡張 BNF のバージョンは ABNF で、インターネット RFC で使われています。RFC 5234 を参照してください。)

戻り値がnov だった場合は、データバッファーには*network overview database* 行が含まれていなければなりません。これは基本的には複数の欄をタブで区切ったものです。

```
nov-buffer = *nov-line
nov-line   = field 7*8[ <TAB> field ] eol
field      = <text except TAB>
```

これらの欄に何が含まれるべきかをきちんと調べたいのならば、Section 12.7.4 [Headers], p. 379, を参照してください。

`(nnchoke-open-server` *`server`* `&optional` *`definitions`*`)`

ここでの*server* は仮想サーバー名です。*definitions* はこの仮想サーバーを定義する`(VARIABLE VALUE)` の組のリストです。

サーバーと接続できなかった場合でも、エラーをシグナルして処理を中断してはいけません。バックエンドは、これ以後さらにこのサーバーに接続しようとする試みを、拒否することを選ぶことができます。実際、そうすべきです。

すでにそのサーバーと接続されていた場合には、この関数は`nil` ではない値を返さなければなりません。このとき、返される結果のデータはありません。

`(nnchoke-close-server &optional` *`server`*`)`

server との接続を閉じて、これに関連するすべてのリソースを開放します。もし何らかの理由でサーバーを閉じることができない場合は、`nil` を返します。

返される結果のデータはありません。

`(nnchoke-request-close)`

すべてのサーバーとの接続を閉じて、バックエンドが保有していたすべてのリソースを開放します。このバックエンドによって作られたすべてのバッファーを削除しなければなりません。(もっとも`nntp-server-buffer` は削除されませんが。) 普通この関数は Gnus が終了するときにのみ呼び出されます。

返される結果のデータはありません。

`(nnchoke-server-opened &optional` *`server`*`)`

server が現在の仮想サーバーで、かつその物理サーバーへの接続が生きている場合、この関数は`nil` ではない値を返さなければなりません。どんな状況でも、この関数は接続が失われたサーバーへの再接続を試みてはいけません。

返される結果のデータはありません。

`(nnchoke-status-message &optional` *`server`*`)`

この関数は*server* からの最後のエラーメッセージを返します。

返される結果のデータはありません。

`(nnchoke-request-article` *`article`* `&optional` *`group server to-buffer`*`)`

この関数の結果のデータは、*article* で指定された記事でなければなりません。`Message-ID` か番号のいずれかを指定することができます。`Message-ID` による記事の取得を実装するかどうかは任意ですが、それが可能になっている方が良いでしょう。

to-buffer が`nil` ではなかったら、結果のデータは通常のデータバッファーの代わりに、このバッファーに返さなければなりません。Gnus は主に、記事バッファーに直接記事を挿入するように要求しますが、これによって、多量のデー

タをあるバッファーから別のバッファーにコピーするのを避けることが可能になります。

もし少しでも可能なら、この関数は cons セルを返すべきです。その`car` は取得した記事があるグループ名で、`cdr` は記事の番号です。これによって、`Message-ID` で記事を取得したときに、Gnus が本当のグループと記事番号を知ることができるようになるでしょう。これが不可能な場合は、記事の取得に成功したときに`t` を返さなければなりません。

(nnchoke-request-group *group* &optional *server fast info*)

group のデータを取得します。この関数には*group* を現在のグループにするという副作用もあります。

fast が設定されたなら、有用なデータを返す面倒を行なわずに、単に*group* を現在のグループにします。

info が与えられると、バックエンドがグループの情報構造(*info*) を更新することを可能にします。

これが結果のデータの例と、定義それ自体です:

```
211 56 1000 1059 ifi.discussion
```

最初の数値は状態で、これは 211 でなくてはなりません。次はそのグループにある記事の総数、最小の記事番号、最大の記事番号、そして最後がグループ名です。しかし、いくつかの記事はキャンセルされているかもしれないので、記事の総数は、記事の最大・最小番号から単純に考えられる数よりも小さいかもしれないことに注意してください。Gnus は総数を単に捨ててしまうので、(それが問題であるときに) 正しい値を生成する面倒を負うべきかどうかは、読者への課題として残してあります。もしそのグループに記事が無かったら、最小記事番号は 1、最大は 0 として報告しなければなりません。

```
group-status = [ error / info ] eol
error        = [ "4" / "5" ] 2<number> " " <Error message>
info         = "211 " 3* [ <number> " " ] <string>
```

(nnchoke-close-group *group* &optional *server*)

group を閉じて、それに関連するすべてのリソースを開放します。ほとんどのバックエンドは何もすることが無いでしょう。

返される結果のデータはありません。

(nnchoke-request-list &optional *server*)

server 上で利用可能なすべてのグループのリストを返します。本当に全部 という意味です。

これは、たった二つしかグループを持っていないサーバーの場合の例です:

```
ifi.test 0000002200 0000002000 y
ifi.discussion 3324 3300 n
```

各行にはグループ名、そのグループ内の最大の記事番号、最小の記事番号、そして最後にフラグがあります。もしそのグループに記事が無かったら、最小記事番号は 1、最大は 0 として報告しなければなりません。

```
active-file = *active-line
active-line = name " " <number> " " <number> " " flags eol
name        = <string>
```

```
flags       = "n" / "y" / "m" / "x" / "j" / "=" name
```

フラグは、そのグループが読み出し専用(‘n’) か、司会者付き(‘m’) なのか、死んでいる(‘x’) か、どこか他のグループの別名(‘=other-group’) なのか、それらのどれでもない(‘y’) のかを示します。

(nnchoke-request-post &optional *server*)
この関数は、現在のバッファーを投稿しなければなりません。投稿が成功したかどうかを返しても構いませんが、必須ではありません。例えば、投稿が非同期に行なわれる場合は、この関数が終了した時点では、投稿は普通完了していません。その場合この関数は、投稿を完了させることができないときに、それをはっきりと利用者に知らせる見張り(sentinel) のようなものを設定するべきでしょう。

この関数から返される結果のデータはありません。

12.7.2.2 任意バックエンド関数

(nnchoke-retrieve-groups *groups* &optional *server*)
groups はグループのリストです。また、この関数はそれら全部のグループのデータを要求しなければなりません。どうやってそれを行なうかは Gnus の知ったことではありませんが、これをできるだけ迅速な方法で行なうことに挑まなければなりません。

この関数の戻り値はactive かgroup のどちらでも良く、それが結果のデータの形式が何であるかを示します。前者はnnchoke-request-list によるデータと同じ形式です。一方後者はnnchoke-request-group が返すものと同じ形式の、バッファーを埋める行です。

```
group-buffer = *active-line / *group-status
```

(nnchoke-retrieve-group-data-early *server* &optional *infos*)
これは*server* からの早期非同期取得を開始して、*infos* (現在のグループ情報構造リスト) を更新するためのものです。戻り値は、対応するnnchoke-finish-retrieve-group-infos のトークンです。

(nnchoke-finish-retrieve-group-infos *server infos data*)
これはnnchoke-retrieve-group-data-early の非同期取得を終了するために後で呼び出されます。そのトークンは*data* として渡され、*infos* を更新するはずです。

(nnchoke-request-update-info *group info* &optional *server*)
Gnus のグループ情報(see Section 12.7.6 [Group Info], p. 380) が、バックエンドのそれを改変するために渡されます。これはバックエンドが(仮想グループやimap グループの場合のように)、本当にすべての情報を持っている場合に役に立ちます。この関数は、その要求に適合させる情報を破壊的に置き換えて、nil ではない値を返さなければなりません(例外的にnntp-request-update-info は、ネットワーク資源を浪費しないように常にnil を返します)。

この関数が返す結果のデータはありません。

(nnchoke-request-type *group* &optional *article*)
利用者が「ニュースを送信する」命令(例えば、概略バッファーで*F*) を実行したときに、Gnus は利用者がフォローアップしようとしている記事がニュースなの

かメールなのかを知っている必要があります。この関数は*group* の中の*article* がニュースであれば`news` を、メールであれば`mail` を、その種別を判定できない場合は`unknown` を返さなければなりません。(*article* 引数は、メールグループとニュースグループがごちゃまぜになっているかもしれない`nnvirtual` において必要です。) *group* と*article* は両方とも`nil` であるかもしれません。

この関数が返す結果のデータはありません。

`(nnchoke-request-set-mark` *group action* `&optional` *server*`)`

記事の印を設定/消去/追加します。通常 Gnus は記事の印(既読、可視、期限切れ消去など) を内部で扱い、`~/.newsrc.eld` に保存します。しかし、いくつかのサーバー(例えばIMAP) は記事のすべての情報をサーバーで持っているので、Gnus が印の情報をサーバーに伝搬させる必要があります。

action は印を設定する要求のリストで、以下の様式を持ちます:

```
(range action mark)
```

range は印を付けたい記事の範囲です。*action* は`add` または`del` で、印を追加したり消すために使われます(言及されていないすべての印は保存します)。*mark* は印のリストです。それぞれの印はシンボルです。現在使われている印は`read`, `tick`, `reply`, `expire`, `killed`, `dormant`, `save`, `download`, `unsend` および`forward` ですが、あなたのバックエンドは、可能ならこれらを制限をするべきではありません。

矛盾する動作が指定された場合は、リストの最後の動作が効力を持つものになるべきです。すなわち、*action* が記事 1 に**可視** 印を追加する要求を含んでいて、リストのおしまいの方で、同じ記事から印を消去することを要求していたならば、印は実際には消去されるべきです。

action リストの例です:

```
(((5 12 30) 'del '(tick))
 ((10 . 90) 'add '(read expire))
 ((92 94) 'del '(read)))
```

関数は印を設定できなかった記事の範囲を返さなければなりません(現在はどんな目的のためにも使われていません)。

この関数が返す結果のデータはありません。

`(nnchoke-request-update-mark` *group article mark*`)`

バックエンドが嫌う印を利用者が設定しようとしたら、この関数がその印を変更することができます。この関数が返したどんなものでも、Gnus は*article* への印として元の*mark* の代わりに使います。バックエンドがそれでも構わない場合には、元の*mark* を返さなければなりません。`nil` やその他のゴミを返してはいけません。

私に考えられるこれの利用法は、それで`nnvirtual` が行なっていることだけです---その仮想グループで既読の印を付けると、もし構成要素のグループが自動期限切れ消去可能ならば、結果としてその記事に期限切れ消去の印が付けられます。

この関数が返す結果のデータはありません。

`(nnchoke-request-scan &optional` *group server*`)`

バックエンドが新着記事を検査する要求を(Gnus か他の何かによって) 行なうときはいつでも、あれやこれやとこの関数が呼び出されるでしょう。この関数

が起動されると、一般にメールバックエンドはスプールファイルを読むかPOPサーバーに問い合わせます。*group* に留意する必要はありません---バックエンドが、一つのグループだけを走査するのが大変すぎると判断した場合には、すべてのグループを総ががりで走査しても構いません。ですが、その方が実用的ならば、局所に限定するのが良いでしょう。

この関数が返す結果のデータはありません。

(nnchoke-request-group-description *group* &optional *server*)

この関数が返す結果のデータは、*group* の説明でなければなりません。

```
description-line = name <TAB> description eol
name             = <string>
description      = <text>
```

(nnchoke-request-list-newsgroups &optional *server*)

この関数が返す結果のデータは、サーバー上で利用できるすべてのグループの説明でなければなりません。

```
description-buffer = *description-line
```

(nnchoke-request-newgroups *date* &optional *server*)

この関数から返される結果のデータは、'date' 以降に作成されたすべてのグループでなければなりません。'date' は人間が読める普通の日付の形式(すなわち、メールやニュースのヘッダーで使われる形式で、ディフォルトは関数message-make-date が返すもの) です。データは active バッファーの形式でなければなりません。

この関数が「多すぎる」グループを返すのはオッケーです。いくつかのバックエンドでは、新しいグループだけではなくて、すべてのグループのリストを返す方が安上がりに済むことを見出すかもしれません。しかし、たくさんのグループがあるバックエンドで、これをしてはいけません。普通、利用者が自分で作ったグループならば多すぎることはないでしょうから、nnml とそれに類するものはたぶん心配ありません。しかしnntp のようなバックエンドでは、グループはサーバーによって作られているので、いかにもたくさんのグループがありそうです。

(nnchoke-request-create-group *group* &optional *server*)

この関数は*group* という名前の空のグループを作成しなければなりません。

返されるデータはありません。

(nnchoke-request-expire-articles *articles* &optional *group server force*)

この関数は、*articles* の範囲のすべての記事に対して期限切れ消去の処理を行ないます(現在*articles* は記事番号の単純なリストです)。記事がどれだけ古いかを、この関数で消去される前に判定することは、バックエンドに任されています。*force* がnil ではない値だったら、それらがどんなに新しくても、すべての*articles* を消去しなければなりません。

この関数は削除しなかった、あるいは削除することができなかった記事のリストを返さなければなりません。

返される結果のデータはありません。

(nnchoke-request-move-article *article group server accept-form* &optional *last*)

この関数は*group* にある記事*article* (番号) を、*accept-form* を呼び出すことによって移動しなければなりません。

この関数は、当の記事を移動させるための準備として、それが記事に付加したどんなヘッダー行をも削除して、記事を大体において「きれい」にしておく必要があります。そして「きれい」な記事があるバッファーで、*accept-form* をeval しなければなりません。これは実際に複製を行ないます。もしこのevalがnil 以外の値を返したら、その記事を削除しなければなりません。

もし*last* がnil だったら、それはこの直後にさらに要求が発行される見込みが高いことを意味し、これによっていくらか最適化ができるようになります(訳注: 例えばnil だったらサーバーとの接続を閉じないでおくとか)。

この関数は、移動先のグループ名がcar で、移動先の記事番号がcdr である cons セルを返さなければなりません。

返されるデータはありません。

訳注: 移動先のグループは*accept-form* の中で指定します。そこで使われるのが、次のnnchoke-request-accept-article です。

(nnchoke-request-accept-article *group* &optional *server last*)

この関数は、現在のバッファーの中身を*group* に挿入します。*last* がnil だったら、この関数へのさらなる呼び出しが直ちに行なわれるだろうという意味です。

この関数はグループ名がcar で、移動先の記事番号がcdr である cons セルを返さなければなりません。

そのグループは、記事を受け入れてもらうことをバックエンドが要求する前に存在しなければなりません。

返されるデータはありません。

(nnchoke-request-replace-article *article group buffer*)

この関数は*group* から記事*article* (番号) を削除して、代わりに*buffer* の中身をそこに挿入しなければなりません。

返されるデータはありません。

(nnchoke-request-delete-group *group force* &optional *server*)

この関数は*group* を消去しなければなりません。もし*force* が設定されていたら、そのグループ内のすべての記事を本当に消去して、そしてそのグループ自身を消去しなければなりません。(もし「グループ自身」というものがあれば。)

返されるデータはありません。

(nnchoke-request-rename-group *group new-name* &optional *server*)

この関数はグループ名を*group* から*new-name* に変更しなければなりません。*group* 内にあるすべての記事は、*new-name* に移動しなければなりません。

返されるデータはありません。

12.7.2.3 エラーメッセージの発行

バックエンドはエラーの状況の報告にnnheader-report を使わなければなりません---要求を実行できないときにエラーを生起させてはいけません。この関数の最初の引数はバック

エンド名のシンボルで、残りは、複数の引数があればformat への引数として解釈され、一つであればただの文字列です。この関数は常にnil を返さなければなりません。

```
(nnheader-report 'nnchoke "You did something totally bogus")

(nnheader-report 'nnchoke "Could not request group %s" group)
```

一方 Gnus は、サーバーからnil を返されたときにnnheader-get-report を呼び出します。するとこの関数が、当のバックエンドに対して最後に報告されたメッセージを返します。この関数は一つの引数---サーバーのシンボルを取ります。

内部的には、これらの関数は*back-end*-status-string にアクセスます、したがってnnchoke バックエンドはそのエラーメッセージをnnchoke-status-string に格納します。

12.7.2.4 新しいバックエンドを書く

多くのバックエンドはよく似通っています。nnml はnnspool と瓜二つですが、サーバー上の記事を編集することができます。nnmh はまるでnnml のようですが、アクティブファイルを使わないし、概要データベースも保持しません。nndir はnnml にとても似ていますが、これには「グループ」の概念は無く、記事の編集はできません。

新しいバックエンドを書くときに他のバックエンドから関数を「継承」できたらなあ、と思うのは理に適っています。そしてまさに、あなたがそうしたければ、それができるのです。(あなたがそうしたくなければしなくても良いですよ、もちろん。)

すべてのバックエンドは、公共変数と公共関数をnnoo というパッケージを使って宣言します。

他のバックエンドから関数を継承するには(そして現在のバックエンドから他のバックエンドに関数を継承できるようにするには)、以下のマクロを使用しなければなりません:

nnoo-declare
: このマクロは、最初の引数を、その後に続く引数の子供であることを宣言します。例えば:

```
(nnoo-declare nndir
  nnml nnmh)
```

ここでnndir は、nnml とnnmh の両方から関数を継承するつもりであることを宣言しています。

defvoo
: このマクロはdefvar と等価ですが、その変数を公共サーバー変数として登録します。ほとんどの状態志向型の変数は、defvar ではなくdefvoo によって宣言するべきです。

通常のdefvar の引数に加えて、このマクロは親バックエンドにおける変数のリストを取ります。それらの親バックエンドで定義されている関数を子のバックエンドで実行するときに、その関数の中でアクセスされる親の変数を、子の変数で置き換えます。

```
(defvoo nndir-directory nil
  "Where nndir will look for groups."
  nnml-current-directory nnmh-current-directory)
```

これはnndir のためにnnml の関数が呼び出されたときに、nnml-current-directory はnndir-directory に設定されるという意味です。(nnmh も同様です。)

`nnoo-define-basics`
: このマクロは、ほとんどすべてのバックエンドが持つべき共通関数をいくつか定義します。

```
(nnoo-define-basics nndir)
```

`deffoo` : このマクロはまさに`defun` のようなもので、同一の引数を取ります。通常の`defun` の処理に加えて、このマクロは他のバックエンドがそれを継承できるように、その関数が公共物になっているものとして登録します。

`nnoo-map-functions`
: このマクロは、現在のバックエンドの関数から親バックエンドの関数への、置き換えができるようにします。

```
(nnoo-map-functions nndir
  (nnml-retrieve-headers 0 nndir-current-group 0 0)
  (nnmh-request-article 0 nndir-current-group 0 0))
```

これは`nndir-retrieve-headers` が呼び出されたときに、一番目、三番目、および四番目の引数が`nnml-retrieve-headers` に渡され、一方、二番目の引数は`nndir-current-group` の値として設定されるという意味です。

`nnoo-import`
: このマクロは他のバックエンドから関数を輸入します。これは単にまだ定義されていない関数を定義するだけなので、ソースファイルの最後に書かなければなりません。

```
(nnoo-import nndir
  (nnmh
   nnmh-request-list
   nnmh-request-newgroups)
  (nnml))
```

これは、`nndir-request-list` への呼び出しは単に`nnmh-request-list` に引き渡されなければならず、一方`nnml` の公共関数でまだ`nndir` で定義されていないものをここで定義するということです。

以下は`nndir` バックエンドのちょっと短縮した版です。

```
;;; nndir.el — 単一のディレクトリーをニュースグループにする
;; Copyright (C) 1995,1996 Free Software Foundation, Inc.

;;; Code:

(require 'nnheader)
(require 'nnmh)
(require 'nnml)
(require 'nnoo)
(eval-when-compile (require 'cl))

(nnoo-declare nndir
  nnml nnmh)

(defvoo nndir-directory nil
```

```
  "nndir がグループを探す場所。"
  nnml-current-directory nnmh-current-directory)

(defvoo nndir-nov-is-evil nil
  "これが nil でなかったら NOV ヘッダーを取得しません。"
  nnml-nov-is-evil)

(defvoo nndir-current-group ""
  nil
  nnml-current-group nnmh-current-group)
(defvoo nndir-top-directory nil nil nnml-directory nnmh-directory)
(defvoo nndir-get-new-mail nil nil nnml-get-new-mail nnmh-get-new-mail)

(defvoo nndir-status-string "" nil nnmh-status-string)

;;; インターフェース用の関数。

(nnoo-define-basics nndir)

(deffoo nndir-open-server (server &optional defs)
  (setq nndir-directory
        (or (cadr (assq 'nndir-directory defs))
            server))
  (unless (assq 'nndir-directory defs)
    (push `(nndir-directory ,server) defs))
  (push `(nndir-current-group
          ,(file-name-nondirectory
            (directory-file-name nndir-directory)))
        defs)
  (push `(nndir-top-directory
          ,(file-name-directory (directory-file-name nndir-directory)))
        defs)
  (nnoo-change-server 'nndir server defs))

(nnoo-map-functions nndir
  (nnml-retrieve-headers 0 nndir-current-group 0 0)
  (nnmh-request-article 0 nndir-current-group 0 0)
  (nnmh-request-group nndir-current-group 0 0)
  (nnmh-close-group nndir-current-group 0))

(nnoo-import nndir
  (nnmh
   nnmh-status-message
   nnmh-request-list
   nnmh-request-newgroups))

(provide 'nndir)
```

12.7.2.5 新しいバックエンドを Gnus に繋げる

あなたの新しいバックエンドを Gnus で使い始めるのはとても簡単です---単にgnus-declare-backend 関数で宣言するだけです。これはバックエンドをgnus-valid-select-methods 変数に追加します。

gnus-declare-backend は二つの引数を取ります---バックエンドの名前と任意の数の能力*abilities* です。

これが例です。

```
(gnus-declare-backend "nnchoke" 'mail 'respool 'address)
```

そして上記の行がnnchoke.el ファイルに入ります。

能力には以下のものがあります:

`mail` これはメール風バックエンドです---フォローアップは(たいていは) メールで送られるはずです。

`post` これはニュース風バックエンドです---フォローアップは(たいていは) ニュースで送られるはずです。

`post-mail`
このバックエンドはメールとニュースの両方をサポートします。

`none` これはニュースでもメールでもないバックエンドです---まったく違った何かです。

`respool` これは再スプールをサポートします---というか、その元の記事とグループを書き換えることができます。

`address` サーバーの名前が仮想サーバー名に含まれていなければなりません。これはほとんど全部のバックエンドに当てはまります。

`prompt-address`
グループバッファーで*B* などの命令を実行したときに、利用者はアドレスの入力を求められるはずです。例えばこれはnntp のようなバックエンドに当てはまりますが、nnmbox はそうではありません。

12.7.2.6 メール風バックエンド

メールバックエンドがその他のバックエンドに対して一線を画しているのは、ほとんどのメールバックエンドがnnmail.el で定義されている共通の関数に強く依存しているという点です。例えばこれはnnml-request-scan の定義です:

```
(deffoo nnml-request-scan (&optional group server)
  (setq nnml-article-file-alist nil)
  (nnmail-get-new-mail 'nnml 'nnml-save-nov nnml-directory group))
```

単にnnmail-get-new-mail にいくつか引数を与えて呼び出すだけで、nnmail がメールの移動や分割のすべての面倒を見てくれます。

この関数は四つの引数を取ります。

method これは、どのバックエンドがこの呼び出しの責任を負うかを指示するシンボルでなければなりません。

exit-function
この関数は、分割が実行された後で呼び出されるものでなければなりません。

temp-directory
一時ファイルを格納する場所です。

group この引数は省略可能です。分割が一つのグループだけに対して行なわれる場合は、この引数でグループ名を指定しなければなりません。

`nnmail-get-new-mail` は、それぞれの記事を保存するために*back-end*`-save-mail` を呼び出します。*back-end*`-active-number` は、この記事に割り当てられた記事番号を調べるために呼び出されます。

この関数は次の変数も使用します: *back-end*`-get-new-mail` (このバックエンドの新着メールを受け取るかどうか)、新しいアクティブファイルを生成するための*back-end*`-group-alist` および*back-end*`-active-file` です。*back-end*`-group-alist` は、以下のようなグループとアクティブの連想リストです:

```
(("a-group" (1 . 10))
 ("some-group" (34 . 39)))
```

12.7.2.7 Web フィードのバックエンド

新しいタイプの web フィード(RSS、Atom) またはその他のタイプのフィードのバックエンドを作成したい場合のために、そのようなバックエンドを作成することができる「抽象」バックエンド(`nnfeed`) があります。Gnus に関する最小限の知識が必要です。

`nnfeed` は、サーバー変数に格納された関数を使って、フィードの情報を解析する汎用パーサーを定義します(ただし、実際には`nnfeed` 自体はそのような関数を定義しません)。

フィードから解析されたデータはサーバー変数に保存されます(そして`gnus-directory` の、バックエンド名に対応する名前のサブディレクトリに、フィードごとに保存されます)。

フィードから解析されたデータを使う Gnus バックエンドインターフェースも定義されます。

したがって新しいバックエンドは、`nnfeed` を継承し、フィードのタイプのための(かなり) 汎用的な解析関数を定義して、必要なサーバー変数を設定するだけで済みます。

`nnfeed` は元々Atom 配信フォーマットの web フィード(see Section 7.5.3 [Atom], p. 202) をサポートするために作成されましたが、非常に汎用的です(これを書いている時点では、サポートされているすべての機能を有効に使うことが可能な標準 web フィードは存在しません)。単一のフィードに含まれる複数のグループをサポートします。これにより、フィード全体を事前に読み込むことができない(または読み込むべきではない) 状況を許容し、解析されたデータから記事を実際に印刷する非常に高度なカスタマイズを(適度に強力なディフォルトのメソッドを提供しながら) 可能にします。

実装の詳細については`nnfeed.el` の色々な`nnfeed-*` サーバー変数のドキュメントと冒頭の Commentary、および他のコメントを参照してください。

12.7.3 スコアファイルの構文

スコアファイルは簡単に分析できるだけでなく、極めて柔軟な対応ができるようになっています。それには Emacs Lisp のリストとして読み込むことができるような構文がふさわしいだろうと判断されました。

これは良くあるスコアファイルです:

```
(("summary"
  ("Windows 95" -10000 nil s)
```

```
 ("Gnus"))
("from"
 ("Lars" -1000))
(mark -100))
```

スコアファイルの BNF 定義です。

```
score-file      = "" / "(" *element ")"
element         = rule / atom
rule            = string-rule / number-rule / date-rule
string-rule     = "(" quote string-header quote space *string-match ")"
number-rule     = "(" quote number-header quote space *number-match ")"
date-rule       = "(" quote date-header quote space *date-match ")"
quote           = <ascii 34>
string-header   = "subject" / "from" / "references" / "message-id" /
                  "xref" / "body" / "head" / "all" / "followup"
number-header   = "lines" / "chars"
date-header     = "date"
string-match    = "(" quote <string> quote [ "" / [ space score [ "" /
                  space date [ "" / [ space string-match-t ] ] ] ] ] ")"
score           = "nil" / <integer>
date            = "nil" / <natural number>
string-match-t  = "nil" / "s" / "substring" / "S" / "Substring" /
                  "r" / "regex" / "R" / "Regex" /
                  "e" / "exact" / "E" / "Exact" /
                  "f" / "fuzzy" / "F" / "Fuzzy"
number-match    = "(" <integer> [ "" / [ space score [ "" /
                  space date [ "" / [ space number-match-t ] ] ] ] ] ")"
number-match-t  = "nil" / "=" / "<" / ">" / ">=" / "<="
date-match      = "(" quote <string> quote [ "" / [ space score [ "" /
                  space date [ "" / [ space date-match-t ] ] ] ] ")"
date-match-t    = "nil" / "at" / "before" / "after"
atom            = "(" [ required-atom / optional-atom ] ")"
required-atom   = mark / expunge / mark-and-expunge / files /
                  exclude-files / read-only / touched
optional-atom   = adapt / local / eval
mark            = "mark" space nil-or-number
nil-or-number   = "nil" / <integer>
expunge         = "expunge" space nil-or-number
mark-and-expunge = "mark-and-expunge" space nil-or-number
files           = "files" *[ space <string> ]
exclude-files   = "exclude-files" *[ space <string> ]
read-only       = "read-only" [ space "nil" / space "t" ]
adapt        = "adapt" [ space "ignore" / space "t" / space adapt-rule ]
adapt-rule      = "(" *[ <string> *[ "(" <string> <integer> ")" ] ")"
local           = "local" *[ space "(" <string> space <form> ")" ]
eval            = "eval" space <form>
space           = *[ " " / <TAB> / <NEWLINE> ]
```

認識不可能なスコアファイルの要素は無視されるべきですが、捨ててしまってはいけません。

ご覧のように空白が必要ですが、空白の量と型は重要ではありません。つまり、スコアファイルの様式はプログラマーに任されています---すべてを一つの長ーーい行に吐き出す方がより簡単なのであれば、それでも構いません。

いろいろなアトムの意味は、このマニュアルのどこかで説明されています(see Section 8.4 [Score File Format], p. 239)。

12.7.4 ヘッダー

Gnus は記事のヘッダーを溜めておくために、内部的にはNOV の規格を怪しげなやり方で踏襲する様式を使っています。NOV の仕様を見た作者が、恥知らずにもすべてを盗んだ と思うかもしれませんが、それは正しいです。

「ヘッダー」はひどく荷の重い用語です。「ヘッダー」は RFC 5536 では記事の頭の行(例えば、`From`) について話すのに用いられています。それは多くの人が「ヘッド」---「ヘッダーと本文」の同義語として使っています。(私に言わせれば、これは避けるべきです。) そして Gnus は、私がここで話す「ヘッダー」と言う様式を内部的に使っています。これは基本的には九つの要素からなるベクトルで、それぞれのヘッダー(あ痛っ) が一つの場所を占めます。

これらの場所は、順番に`number`, `subject`, `from`, `date`, `id`, `chars`, `lines`, `xref`, および`extra` です。これらの場所を読み出したり設定するマクロがあります---それらはすべて、それぞれ`mail-header-` と`mail-header-set-` いう予想しやすい名前を持っています。

`extra` のための場所がヘッダーと値の対の連想リストであることを除いて、これらすべての場所には文字列が入ります(see Section 4.1.2 [To From Newsgroups], p. 50)。

12.7.5 範囲

GNUS は非常に有用な概念を導入してくれました。私はそれをたくさん使い、かなり入念に仕上げました。

設問は単純です: 何か番号で呼ぶことができる大量のもの(粗雑な 例としては、例えば記事) を持っていて、それらが「含まれている」ことを表現したいとしましょう。それらを順番に並べるのは、あまり便利ではありません。(20,000 個を順番に並べたものは、ちょっと長たらしいですよね。)

解決策は設問と同じくらい単純です。単にその並びを折りたためば良いのです。

```
(1 2 3 4 5 6 10 11 12)
```

は次のように変形されます。

```
((1 . 6) (10 . 12))
```

単独のものを表すために‘`(13 . 13)`’ のようなやっかいな要素を持たなくても良いように、‘`13`’ は有効な要素になっています。例えば:

```
((1 . 6) 7 (10 . 12))
```

以下のような二つの範囲を比較して、それらが等しいがとうかを調べるのは、少し手のこんだことになります:

```
((1 . 5) 7 8 (10 . 12))
```

と

```
((1 . 5) (7 . 8) (10 . 12))
```

は等しいです。実際のところ、下り順で並んでいないリストは範囲です:

```
(1 2 3 4 5)
```

これはかなり長ったらしいものですが、完璧に有効な範囲です。以下も有効です:

```
(1 . 5)
```

そして、これはその前の範囲と等しいものです。

これは範囲の BNF 定義です。もちろん、数値の並びが下り順であってはならないことを覚えておかなければなりません。(同じ数値を任意の回数にわたって繰り返すことができますが、範囲の扱いにおいて消え去る傾向があります。)

```
range           = simple-range / normal-range
simple-range    = "(" number " . " number ")"
normal-range    = "(" start-contents ")"
contents        = "" / simple-range *[ " " contents ] /
                  number *[ " " contents ]
```

現在 Gnus は既読記事と記事の印を維持するために範囲を使っています。当局が私にそれをさせたがっているのなら、私は数の範囲の操作を C で実装しようと思っています。(私はまだ尋ねていません。と言うのは、普通の連続体に変換し直さずに、世の中を完全に範囲に基づいたものにするためには何が必要かを、私はもっと考えなければならないからです。)

12.7.6 グループ情報

Gnus はグループのすべての永続的な情報を*group info* リストに格納します。このリストは 3 から 6 (またはそれ以上) の長さの要素で、徹底的にグループを記述します。

ここにあるのはグループ情報(group info) の二つの例です。一つは非常に単純なグループですが、二つ目はもっと複雑なものです:

```
("no.group" 5 ((1 . 54324)))

("nnml:my.mail" 3 ((1 . 5) 9 (20 . 55))
                  ((tick (15 . 19)) (replied 3 6 (19 . 3)))
                  (nnml "")
                  ((auto-expire . t) (to-address . "ding@gnus.org")))
```

最初の要素は「グループ名」---とにかく Gnus が知っているグループです。二番目の要素は「購読度」で、普通は小さな整数です。(それは「階級」(rank) になることもできます。`car` がレベルで`cdr` がスコアのコンスセルです。) 三番目の要素は既読記事の範囲のリストです。四番目の要素はいろいろな種類の記事の印のリストのリストです。五番目の要素は選択方法です(もしくは、そう言いたければ仮想サーバーです)。六番目の要素は「グループパラメーター」のリストで、この章はそのためにあります(訳注: ほんとうに?)。

最後の三つの要素はどれでも、必要が無ければ存在しないこともあります。実際、グループの非常に大部分は最初の三つの要素だけを持ち、それは(最後の三要素が省略できることは) 非常に多くのコンスセルを節約します。

これはグループ情報様式の BNF 定義です:

```
info          = "(" group space ralevel space read
                [ "" / [ space marks-list [ "" / [ space method [ "" /
                space parameters ] ] ] ] ] ")"
group         = quote <string> quote
ralevel       = rank / level
```

```
level         = <integer in the range of 1 to inf>
rank          = "(" level "." score ")"
score         = <integer in the range of 1 to inf>
read          = range
marks-lists   = nil / "(" *marks ")"
marks         = "(" <string> range ")"
method        = "(" <string> *elisp-forms ")"
parameters    = "(" *elisp-forms ")"
```

実は‘marks’ の規則はごまかしです。‘marks’ は‘range’ とともに cons を構成する‘<string>’ ですが、疑似 BNF でそれを現すのは難しいのです。

情報の要素群にアクセスして、それらの値を取得または設定するために、Gnus は一連のマクロを提供しています。

gnus-info-group
gnus-info-set-group
: グループ名を取得/設定(get/set) します。

gnus-info-rank
gnus-info-set-rank
: グループの階級(rank) を取得/設定します(see Section 3.7 [Group Score], p. 20)。

gnus-info-level
gnus-info-set-level
: グループのレベルを取得/設定します。

gnus-info-score
gnus-info-set-score
: グループのスコアを取得/設定します(see Section 3.7 [Group Score], p. 20)。

gnus-info-read
gnus-info-set-read
: 既読記事の範囲を取得/設定します。

gnus-info-marks
gnus-info-set-marks
: 印が付いている記事の範囲のリストを取得/設定します。

gnus-info-method
gnus-info-set-method
: グループの選択方法を取得/設定します。

gnus-info-params
gnus-info-set-params
: グループパラメーターを取得/設定します。

取得するためのすべての関数は一つの引数を取ります---情報のリストです。設定するための関数は二つの引数を取ります---情報のリストと新しい値です。

グループ情報の最後の三つの要素は必須ではないので、要素を設定する前にグループ情報を拡張する必要があるでしょう。それが必要な場合、最後の三つの設定するための関数の第三引数にnil ではない値を指定すれば、自動的に拡張させることができます。(訳注: 例えば三つの要素しかない情報に四つ目の要素を加える処理を第三引数を使わずに行なう

と、(setcar (nthcdr 3 INFO) VALUE) というコードが実行される結果、エラーになってしまいます。)

12.7.7 対話形式の拡張

Gnus は Emacs 標準のinteractive の仕様を、シンボル接頭引数を簡単に使うことができるようにするために、少し拡張しています(see Section 10.3 [Symbolic Prefixes], p. 272)。これはその使い方の例です:

```
(defun gnus-summary-increase-score (&optional score symp)
  (interactive (gnus-interactive "P\ny"))
  ...
  )
```

最上のものはinteractive の式を返すマクロとしてgnus-interactive を実装することでしょうが、Emacs は関数が対話的かどうかを調べるために、ラムダ式に対して単純にassq を行なうので、これは不可能です。そこで、文字列を受け取ってinteractive で使うことができる値を返すgnus-interactive 関数を、代わりに持つことにしました。

この関数は(ほとんど) すべてのinteractive の指定を受け付けますが、もう少し加えることにします。

'y' 現在のシンボル接頭引数---変数gnus-current-prefix-symbol です。

'Y' 現在のシンボル接頭引数のリスト---変数gnus-current-prefix-symbol です。

'A' 現在の記事番号---関数gnus-summary-article-number です。

'H' 現在の記事ヘッダー---関数gnus-summary-article-header です。

'g' 現在のグループ名---関数gnus-group-group-name です。

12.7.8 いろいろなファイル様式

12.7.8.1 アクテイブファイルの様式

アクティブファイルは、対象になっているサーバーのすべての使用可能なグループの目録を保持します。そこには、それぞれのグループの最高と最低の記事番号の目録もあります。

これは普通のアクティブファイルからの抜粋です:

```
soc.motss 296030 293865 y
alt.binaries.pictures.fractals 3922 3913 n
comp.sources.unix 1605 1593 m
comp.binaries.ibm.pc 5097 5089 y
no.general 1000 900 y
```

これはこのファイルの疑似 BNF 定義です:

```
active      = *group-line
group-line  = group spc high-number spc low-number spc flag <NEWLINE>
group       = <non-white-space string>
spc         = " "
high-number = <non-negative integer>
low-number  = <positive integer>
flag        = "y" / "n" / "m" / "j" / "x" / "=" group
```

このファイルの完全な説明は、'innd' のマニュアルページ、特に'active(5)' を見てください。

12.7.8.2 ニュースグループファイルの様式

ニュースグループファイルは、グループの目録をそれらの説明とともに保持します。サーバーにあるすべてのグループがある必要は無いし、そのファイルにあるすべてのグループがサーバーに存在しなければならないこともありません。このファイルは純粋に利用者の情報のためにあります。

様式はとても単純です: グループ名、タブ、そして説明です。これが定義です:

```
newsgroups    = *line
line          = group tab description <NEWLINE>
group         = <non-white-space string>
tab           = <TAB>
description   = <string>
```

12.8 異教徒への Emacs

信じるかどうかはともかく、Gnus Love Boat の旅に搭乗する前にあまり Emacs を使ったことが無いという Gnus の利用者たちがいます。あなたがそのような不運な人々の一人で、“*C-M-a*” や「リージョンを kill する」、それに「`gnus-flargblossen` を連想リストに設定してください。そのキーはグループ名に合致するために使われる正規表現です。」といったことが、あなたにとって少しかまったく意味の無い魔法の言葉ならば、この付録はあなたのためにあります。もしあなたがすでに Emacs に親しんでいるのであれば、これを無視してあなたの猫を可愛がりに行ってください。

12.8.1 打鍵

- Q: 経験のある Emacs の利用者とは何ですか?
- A: 端末にペダルがあったらなぁと願う人のことです。

はい、Emacs を使うとコントロールキー、シフトキー、メタキーをたくさん使うようになるでしょう。これは一部の人々(主に`vi` 利用者) には非常に煩わしいものですが、私たちはその地獄を愛します。諦めてそれを甘受してください。Emacs は本当は“Escape-Meta-Alt-Control-Shift”の略で、あなたがいかがわしい(Emacs の作者のような) 出どころから聞いているかもしれない“Editing Macros”ではありません。

シフトキーは普通は両手の小指の近くにあって、普通は大文字などを打つために使われています。あなたは絶え間なくそれを使いますよね。コントロールキーには普通“CTRL”のような印が付いています。メタキーは奇妙なことにどのキーボードでもそういう印が付いていません。それは普通はキーボードの左手側にあって、最下段にあるのが一般的です。

さて、私たち Emacs 人は、それがひどく不便なので「meta-control-m キーを押す」とは言いません。私たちが使うのは「*C-M-m* を押す」です。*M-* は「メタ」を現す接頭語で、“C-”は「コントロール」を意味する接頭語です。ですから「*C-k* を押す」は、「コントロールキーを押し続けながら次に*k* を押す」ということです。「*C-M-k* を押す」は「メタキーとコントロールキーを押し続けながら次に*k* を押す」ということです。簡単です、よね?

このことは、すべてのキーボードがメタキーを持っているわけではないという事実によって、多少ややこしくなっています。そういう場合には「エスケープ」キーを使えばよいでしょう。ただしメタキーを持っているときより作業が増えるので、メタキーのあるキーボードを手に入れていただくことを謹んでお勧め申し上げます。それ無しでは生きて行けないでしょう。

12.8.2 Emacs Lisp

Emacs はエディターの王様です。なぜなら、それが真の Lisp インタープリターだからです。あなたが叩くすべてのキーは、何らかの Emacs Lisp コードの小片を実行します。Emacs Lisp はインタープリターで実行される言語なので、どのキーが何のコードを実行するかを任意に設定することができます。あなたは、ただそうすれば良いのです。

Gnus は Emacs Lisp によって書かれていて、インタープリターで実行されるたくさんの関数によって動作します。(これらは速度のためにバイトコンパイルされていますが、インタープリターで実行されることに変わりはありません。) もし Gnus のある動作が好みではないと感じたら、それを違うやり方で実行させるのは取るに足らないことです。(えーと、少なくとも Lisp コードの書き方を知っていれば。) でもそれはこのマニュアルの範疇ではないので、Gnus をカスタマイズするために`~/.gnus.el` ファイルで普段使われるいくつかの一般的な構文のことだけを話すことにしましょう。(`~/.emacs` ファイルを使うこともできますが、Gnus に関する設定には`~/.gnus.el` ファイルを使う方がはるかに良いです。)

もし変数gnus-florgbnize を四(4) に設定したいのであれば、以下のものを書きましょう:

```
(setq gnus-florgbnize 4)
```

この関数(本当は「特殊形式」(special form)) setq は、変数を何かの値に設定することができるものです。これがあなたが本当に知っていなければならないことのすべてです。これからは Gnus の動作を変更するために、たくさんのこういうもので~/.gnus.el ファイルを埋め尽くすことができます。

そういうものを~/.gnus.el ファイルに入れておくと、それらは次回に Gnus を起動したときに読み込まれ、eval (それは「実行」の Lisp 語です) されます。もし変数をすぐに変更したいのであれば、閉じ括弧の後ろでC-x C-e とタイプすれば良いのです。それは前にある「式」(ここでは簡単なsetq 文) をeval します。

さあ、やってみましょう---あなたが Emacs の前にいるのなら試してみてください。C-x C-e をタイプすると、エコーエリア(訳注: 一般には Emacs の画面の一番下) に‘4’ が現われるのが見えるでしょう。それはあなたがeval した式の戻り値です。

いくつかの落とし穴:

もしマニュアルが「gnus-read-active-file をsome に設定しなさい」と言ったなら、それは

```
(setq gnus-read-active-file 'some)
```

ということです。

一方、マニュアルが「gnus-nntp-server-file を/etc/nntpserver に設定しなさい」と言ったなら、それは

```
(setq gnus-nntp-server-file "/etc/nntpserver")
```

ということです。

ですから、文字列(後者) をシンボル(前者) と混同しないように注意してください。マニュアルは明確に区別していますが、混乱しやすいかもしれません。

12.9 よく尋ねられる質問

要約

これは新しい Gnus のよく尋ねられる質問のリストです。

機能に関することや提案はding list に送ってください。

12.9.1 序

これは Gnus のよく尋ねられる質問のリストです。

Gnus は Emacs の要素として実装された Usenet ニュースリーダーおよび電子メールのユーザーエージェントです。それは 1990 年代の初期から何らかの形で存在しており、その期間の多くにおいて Emacs の標準要素として配布されてきました。Gnus 5 は最新の(そして最も偉大な) 作品です。オリジナルの版は GNUS と言い、梅田政信さんが書きました。1994 年の秋が忍び寄る頃、退屈していたラルス・マッグヌ・イングブリグットスン(Lars Magne Ingebrigtsen) は Gnus を書き直そうと決心しました。

その最大の強みは、極めてカスタマイズに適しているという事実にあります。このことを始めて目にすると引いてしまうかもしれませんが、それを利用する準備ができるまでは、複雑なもののほとんどは無視することができます。そこそこの量の(いろんなメーリングリストに配信される) 電子メールがやって来るのならば、流通量が多いメーリングリストを読みたいけれども遅れずについていくことができないのならば、流通量が多いニュースグループを読んでいるのならば、あるいは単に退屈しているのならば、Gnus はあなたが望むものです。

この FAQ は 2002 年 3 月まで Justin Sheehy によって維持されていました。彼は、それ以前にすばらしい仕事をしてくれた Steve Baur と Per Abrahamsen に感謝を表明しています。私たちも同じことをしましょう: ありがとう Justin!

ここにある情報は、Gnus 開発メーリングリストの援助でコンパイルされました。どんなエラーあるいはミスプリントも Gnus チームが犯した誤りです。すみません。

訳注: そしてどんな誤訳の責任も gnus-doc-ja チームにあります。

12.9.2 インストールに関する FAQ

質問 1.1

Gnus の最新版は何で、どこで見つかりますか?

回答

最新版の Gnus は Emacs に含まれています。

質問 1.2

ときどき目にする No Gnus や Oort Gnus って何ですか?

回答

Oort Gnus は Gnus の開発版の名前で、2003 年の秋に Gnus 5.10 になりました。No Gnus は開発版の名前で、Gnus 5.12 になりました。

12.9.3 起動/ グループバッファー

質問 2.1

いつも Gnus を起動すると"Gnus auto-save file exists. Do you want to read it?" というメッセージを受け取るのですが、これは何を意味しているのですか? また、どうやったら回避できますか?

回答

このメッセージが意味するのは、最後に Gnus を使ったときに適切に終了させなかったために、ディスクにその情報(例えばどのメッセージを読んだかの) を書き込むことができなかったので、今、それらの情報を自動保存(auto-save) ファイルから復活させるかどうかを尋ねられているということです。

このメッセージが発せられるのを回避するには、Gnus を終了させるときに単に Emacs を kill するのではなく、グループバッファーで`q` を使うようにしてください。

質問 2.2

Gnus は私がどのグループを購読するかを覚えてくれません。どうしてですか?

回答

Gnus を起動したときに、上記の質問と回答(see [FAQ 2-1], p. 387) で述べられているメッセージを受け取りませんでしたか? これは同じ問題の別の症状なので、上記の回答を読んでください。

質問 2.3

グループバッファーの各行の形式を変更するには、どうしたら良いですか?

回答

変数`gnus-group-line-format` の値を調整しなければなりません。そのやり方についてはマニュアル(see Section “グループ行の仕様” in *The Gnus Manual*) を見てください。このための例です(質問者の`~/.gnus.el` ファイルからの推測です:-):

```
(setq gnus-group-line-format "%P%M%S[%5t]%5y : %(%g%)\n")
```

質問 2.4

私のグループバッファーはちょっと混んでいるのです。それらをもっと楽に巡回できるように、グループをカテゴリーごとにまとまるように並べ変える方法はありますか?

回答

Gnus はトピックモードを提供します。それによってグループをその中へ入れて並べ変えることができるようになります。例えば Linux を扱うすべてのグループは linux というトピックに収め、音楽を扱うすべては music というトピックに収め、スコットランド音楽を扱うすべては scottish という music のサブトピックに収めます。

トピックモードに入るには、グループバッファーで単に`t` を叩いてください。すると`T n` を使って現在位置でトピックを作ったり、あるグループを`T m` を使って指定したトピックに移すことができます。さらなるコマンドについてはマニュアルかメニューを見てください。グループ行をより良く行下げ(indent) させるためには、*gnus-group-line-format* 変数の先頭に‘`%P`’ 書法仕様を含ませる必要があるでしょう。

質問 2.5

グループバッファーを手で並べ変えるにはどうしたら良いですか? トピック内のグループを並べ変えるにはどうしたら良いですか?

回答

移動させたいグループの上にポイントを移動して*C-k* を叩き、次にそのグループを移動させる目的の場所にポイントを移動して*C-y* を叩いてください。

12.9.4 メッセージの取得

質問 3.1

今まさに Gnus をインストールして*M-x gnus* で起動したのですが、"nntp (news) open error" としか言ってくれません。どうしたら良いですか?

回答

どこからニュースを取得すべきかを Gnus に教えてあげなければなりません。やり方については文献を読んでください。初めて起動するのであれば、次のようなものを`~/.gnus.el` ファイルに書き込んでみてください:

```
(setq gnus-select-method '(nntp "news.yourprovider.net"))
(setq user-mail-address "you@yourprovider.net")
(setq user-full-name "Your Name")
```

質問 3.2

Windows を使っているのですが`~/.gnus.el` の意味がわかりません。

回答

`~/` とは Gnus と Emacs が設定ファイルを探す場所であるホームディレクトリーのことです。でも、実はその意味を知らなくても構わないのです。Emacs がわかっていれば十分ですから。:-) *C-x C-f ~/.gnus.el* `RET` とタイプすれば(そう、Windows でもスラッシュでいいのです) Emacs は正しいファイルを開いてくれるでしょう(それが新規ファイルであるために中身が空っぽであることは、おおいにあり得るでしょう)。しかしちょっと待ってください。Emacs が選ぶディレクトリーは、きっとあなたの希望通りにはならないので、正しいやり方でそれを行ないましょう。第一に、例えば c:\myhome のような適当なディレクトリーを(名前に空白を含めないで) 作ってください。そして、このディレクトリーを環境変数 HOME に設定しましょう。これを Windows 9x か Me で行なうには、`autoexec.bat` ファイルに以下の行を追加して、再起動してください。

```
SET HOME=C:\myhome
```

NT, 2000 および XP では、システム・オプションを入力するために*Winkey* + *Pause/Break* を叩いて(もしそれが使えなかったら「コントロールパネル-> システム-> 詳細」を辿って) ください。そこで環境変数を設定できるでしょう。HOME という名前で値が C:\myhome のものを作ってください。再起動は不要です。

では、Emacs に*C-x C-f ~/.gnus.el* `RET` *C-x C-s* を指示して、`~/.gnus.el` ファイルを作りましょう。

質問 3.3

ニュースサーバーが認証を要求します。ディスクにユーザー名とパスワードを保存しておくには、どうすれば良いですか?

回答

次のように、それぞれのサーバーに関する行を含んだ`~/.authinfo` ファイルを作ってください。

```
machine news.yourprovider.net login YourUserName password YourPassword
```

OS が対応しているならば、そのファイルを他人が読めないようにしておきましょう。Unix ではシェル上で次のコマンドを実行してください。

```
chmod 600 ~/.authinfo
```

質問 3.4

Gnus はうまく起動したようなのですが、グループを購読する方法が見つけられません。

回答

そのグループの名前がわかっているのなら、グループバッファーで*U name.of.group* RET を使ってください(タブ補完を使え、ルーク(訳注: オビ = ワン・ケノービの声で))。あるいはグループバッファーで^ を使いましょう。これはあなたをサーバーバッファーへ連れて行きます。その場合は、目的のグループがあるサーバーの上にポイント(カーソル) を置いてRET を叩き、読みたいグループにポイントを移動してから*u* でそのグループを購読しましょう。

質問 3.5

Gnus がすべてのグループを表示してくれません/ このサーバーへの投稿が許可されません。どうしてですか?

回答

プロバイダーのいくつかは匿名での接続を制限していて、認証してからでないと完全なアクセスを許しません。Gnus に認証のための情報(authinfo) を送出させるには、`~/.authinfo` ファイルの該当するサーバーの行の最後に

```
force yes
```

を追加してください。

質問 3.6

複数のサーバーからニュースを取り込みたいのですが、それは可能ですか?

回答

もちろん。変数gnus-secondary-select-methods に、もっと多くの記事の供給元を設定することができます。以下のようなものを`~/.gnus.el` ファイルに追加してください:

```
(add-to-list 'gnus-secondary-select-methods
             '(nntp "news.yourSecondProvider.net"))
(add-to-list 'gnus-secondary-select-methods
             '(nntp "news.yourThirdProvider.net"))
```

質問 3.7

それから、ローカル・スプール・ファイルからニュースを取り込むことは可能ですか?

回答

問題ありません。それは`nnspool` というもう一つの選択方法です。こんなふうに設定してください:

```
(add-to-list 'gnus-secondary-select-methods '(nnspool ""))
```

あるいはNNTP を第一ニュースソースとして使う必要が無いのであれば、こうしてください:

```
(setq gnus-select-method '(nnspool ""))
```

これで Gnus は`/usr/spool/news` にあるスプール・ファイルを探します。何か違うことをやりたいのならば、上記の行を次のようなものに変更してください:

```
(add-to-list 'gnus-secondary-select-methods
             '(nnspool ""
                       (nnspool-directory "/usr/local/myspoolddir")))
```

これは、このサーバーだけのために、スプールが存在するディレクトリーを設定します。さらに記事を投稿するために使うプログラムなどを設定する必要があるかもしれません。そういう場合にどうしたら良いかについては、Gnus のマニュアルを参照してください。

質問 3.8

ニュースを読むのうまくいきましたが、Gnus でメールも読めるようにしたいのです。どうすれば良いですか?

回答

それは少しばかり難しいです。使うことができるメールソースはいっぱいあるし、メールを格納する方法はたくさんあるし、送信するための方法も様々なので。最もありふれているのは、次の二つの事例のようなものでしょう:

1. POP3 サーバーからメールを読み、SMTP サーバーに直接メールを送信します。
2. fetchmail のようなプログラムでメールを取り込んで、Gnus が読むことになっているディレクトリーに格納します。外に行くメールは Sendmail, Postfix または他の MTA によって送出されます。

ときには、これらを併用する必要さえあります。

しかし最初に行なうことは、どの方法でメールを格納するか、Gnus の用語で言うと、どのバックエンドを使うかを Gnus に指示することです。Gnus は多くの異なるバックエンドをサポートしますが、最も一般的に使われているのは`nnml` です。それは一通のメールを一つのファイルに格納し、そのため極めて高速です。でも、あなたが使っているファイルシステムがたくさんの小さなファイルを扱う上で問題があるのならば、一つのグループのすべてのメールを一つのファイルに収める方法を使う必要があるかもしれません。おそらくそういう場合の選択肢が`nnfolder` バックエンドです。

`nnml` を使うには、以下を`~/.gnus.el` ファイルに加えてください:

```
(add-to-list 'gnus-secondary-select-methods '(nnml ""))
```

`nnfolder` を使いたいのならば、あなたの想像した通り、こうすれば良いでしょう:

```
(add-to-list 'gnus-secondary-select-methods '(nnfolder ""))
```

次に、メールをどこから取得するかを Gnus に伝えなければなりません。それがPOP3 サーバーであるのなら、このように設定してください:

```
(with-eval-after-load "mail-source"
  (add-to-list 'mail-sources '(pop :server "pop.YourProvider.net"
                                   :user "yourUserName"
                                   :password "yourPassword")))
```

もしそこにパスワードを書いておくのなら、他人が~/.gnus.el ファイルを読めないようにしておいてください。メールをローカルマシンの伝統的なスプールファイルから読みたい場合は、以下のように設定してください:

```
(with-eval-after-load "mail-source"
  (add-to-list 'mail-sources '(file :path "/path/to/spool/file"))
```

もしそれが Postfix, Qmail または(そういうふうに設定されている) fetchmail によって使われる、一通/一ファイル形式の Maildir であるのならば、こんなふうにしましょう:

```
(with-eval-after-load "mail-source"
  (add-to-list 'mail-sources '(maildir :path "/path/to/Maildir/"
                                       :subdirs ("cur" "new")))
```

そして最後に、メールを一つのディレクトリーにある複数のファイルから読むようにしたいのであれば(例えば procmail がすでに分割してあるという理由で)、設定は以下のようになります:

```
(with-eval-after-load "mail-source"
  (add-to-list 'mail-sources
               '(directory :path "/path/to/procmail-dir/"
                           :suffix ".prcml")))
```

ここで:suffix に指定した".prcml" は、Gnus に拡張子が.prcml のファイルだけを使うことを指示するためのものです。

さあ、後はどうやってメールを送信するかを Gnus に教えるだけです。メールの送信に sendmail を使いたければ(または、あなたのシステムの MTA が何であれ sendmail の役を演じるのならば)、何もしなくても良いのです。でも、もしSMTP サーバーにメールを送りたいのだったら、以下のようなものが~/.gnus.el ファイルに書かれている必要があります:

```
(setq send-mail-function 'smtpmail-send-it)
(setq message-send-mail-function 'smtpmail-send-it)
(setq smtpmail-default-smtp-server "smtp.yourProvider.net")
```

質問 3.9

それから、IMAP でメールを読めるようにするには、どうすれば良いですか?

回答

Gnus でIMAP を使うには二つの方法があります。最初のはIMAP をPOP3 のように使うもので、Gnus はIMAP サーバーからメールを取得してディスクに格納します。これをやりたいのなら(本当にそんなことをしたい人はいないでしょうけれど)、以下の設定を~/.gnus.el ファイルに加えてください:

```
(add-to-list 'mail-sources '(imap :server "mail.mycorp.com"
                                  :user "username"
                                  :pass "password"
```

```
                                        :stream network
                                        :authentication login
                                        :mailbox "INBOX"
                                        :fetchflag "\\Seen"))
```

`:stream` および/または`:authentication` の項はいじる必要があるかもしれません。使うことができる値については Gnus マニュアル(see Section “メールソース指示子” in *The Gnus Manual*) を参照してください。

IMAP をそれが意図された方法で使いたい場合は、違うやり方に従う必要があります。そうするには選択方法に`nnimap` を加え、そこでサーバーに関する情報を与えてください:

```
(add-to-list 'gnus-secondary-select-methods
             '(nnimap "Give the baby a name"
                      (nnimap-address "imap.yourProvider.net")
                      (nnimap-port 143)))
```

さらに Gnus が正しいやり方を推測できない場合には、サーバーに認証してもらう方法を指定しなければならないでしょう。詳しい情報はマニュアル(see Section “IMAP” in *The Gnus Manual*) を参照してください。

質問 **3.10**

職場で MS Exchange サーバーの一つを使っているのですが、Gnus を使ってそこからメールを読むことはできますか?

回答

サーバーにIMAP という新しい運動靴を一足はかせてくれるように、管理人にお願いしてください。そして上記の説明に従って、必要なことを行なってください。

質問 **3.11**

POP3 でメールを取得するサーバーで、メールを消さないように Gnus に指示することはできますか?

回答

はい、POP3 サーバーがUIDL をサポートしていれば(たぶん近ごろではほとんどのサーバーがサポートしています)。そのためには:`leave` *`value`* というペアを、それぞれのメールソースに加えてください。*value* についてはSee Section 7.4.4.1 [Mail Source Specifiers], p. 166.

12.9.5 メッセージを読む

質問 **4.1**

グループに入ると、以前に読んだメッセージが全部なくなってしまっています。もう一度読むには、どうしたら良いですか?

回答

グループバッファーにおいて、そのグループの上にポイントが置かれている状態で`RET` を使うと、未読と可視記事の印(ticked) が付いたメッセージだけが現れます。存在するすべてのメッセージが表示されるようにするには、代わりに*`C-u`* `RET` を使ってください。もし最新の 300 通だけが所望ならば、*`C-u 300`* `RET` を使いましょう。

スレッド表示を有効にしていると、未読のメッセージだけを読み込むのはじれったいかもしれません。スレッドがちぎれてしまうことを防止するのに十分な量の古い記事を取り込むには、以下の設定を追加すれば良いでしょう:

```
(setq gnus-fetch-old-headers 'some)
```

some をt に替えると、すべての記事を表示するようになります(警告: どちらの設定もグループに入ったときに取り込むデータ量を増加させ、グループに入る処理を遅くしてしまいます)。

N 通の記事を取り込むためには、概略バッファーで/o N 命令を使ってください。

すべての古いメッセージは要らないけれども、今まさに読んでいるメッセージの親を辿りたいならば^ 命令を使ってください。今まさに読んでいるメッセージが属しているスレッドのすべての記事を表示したい場合は、A T 命令があなたの望むものです。

質問 4.2

大事なメッセージを、すでに読んだことがあっても、グループに入ったときはいつでも見えるように Gnus に指示するには、どうすれば良いですか?

回答

大事なメッセージには可視記事の印(ticked) を付けることができます。それには、概略バッファーでポイントがその記事を指しているときにu を叩いてください。可視記事の印を消す命令はd (これは可視記事の印を消してから未読の印を付けます) またはM c (そのメッセージに付いているすべての印を消します) です。

質問 4.3

メッセージのヘッダーを見るにはどうしたら良いですか?

回答

t 命令を使うとすべてのヘッダーを表示し、もう一度使うと再びそれらを隠します。

質問 4.4

整形されていない生のメッセージを見るには、どうすれば良いですか?

回答

C-u g 命令を使うと生のメッセージを表示し、g で通常の表示に戻ります。

質問 4.5

記事バッファーの先頭に Gnus がディフォルトで表示するヘッダーを変更するには、どうしたら良いですか?

回答

変数gnus-visible-headers がどのヘッダーを表示するかを制御します。その値は正規表現で、それに合致するヘッダー行が表示されます。したがって、著者、表題、日付、そしてもし存在する場合は Followup-To と MUA / NUA を表示したい場合には、~/.gnus.el ファイルで以下の設定を行なってください:

```
(setq gnus-visible-headers
      '("^From" "^Subject" "^Date" "^Newsgroups" "^Followup-To"
        "^User-Agent" "^X-Newsreader" "^X-Mailer"))
```

質問 4.6

HTML メールを描画するのではなく、テキストのパートがある場合には、そちらの方を Gnus に表示して欲しいです。どうすれば良いですか?

回答

`~/.gnus.el` ファイルで以下の設定を行なってください:

```
(with-eval-after-load "mm-decode"
  (add-to-list 'mm-discouraged-alternatives "text/html")
  (add-to-list 'mm-discouraged-alternatives "text/richtext"))
```

HTML のパートと同じ内容のテキストのパートが無い場合でさえもHTML のパートを描画して欲しくないのであれば、以下の設定を追加してください:

```
(setq mm-automatic-display (remove "text/html" mm-automatic-display))
```

質問 4.7

HTML メールを w3m 以外のブラウザーを使って描画させることはできますか?

回答

'`shr`', '`w3m`', '`links`' および'`lynx`' の中から選択することができます。どれを使うかは変数`mm-text-html-renderer` で指定するので、もし links でメールを描画したいならば、こうしてください:

```
(setq mm-text-html-renderer 'links)
```

質問 4.8

見苦しい体裁のメールをもっと読みやすくするための何かがありますか?

回答

Gnus は入ってきたメールを「洗濯」するために複数の関数を提供します。それらはメニューの Article->Washing 項目を辿ることによって見つけることができるでしょう。最も興味深いものはおそらく“Wrap long lines” (`W w`)、“Decode ROT13” (`W r`)、および Microsoft の製品を使っている多くのユーザーが送ってくる間の抜けた引用付き返信(訳注: 折り畳まれた長い引用行の二行目以降に引用符が前置されないなど) を修繕する“Outlook Deuglify” です(`W Y f` は完全な deuglify を提供します。`W Y C-h` を参照するか、他の deuglification のメニューを参照してください)。

質問 4.9

特定の著者が送信したものや、表題に特定の語が含まれているメッセージを、自動的に無視する方法はありますか? また、より興味深いものを何らかの方法で強調表示させることはできますか?

回答

あなたに必要なのはスコア付けです。スコア付けと言うのは、それぞれのメッセージに整数の値を割り付けるための規則を定義することです。値に基づいて概略バッファーでメッセージを強調表示したり(それが+2000 といった高い値だったら)、自動的に既読にしたり(その値が例えば−800 のような低い値だったら)、あるいは他のいくつかの作用を行なったりします。

メッセージにスコアの値を割り付ける規則を設定するための、基本的な三つのやり方があります。最初の最も簡単なやり方は、今まさに読んでいる記事に基づいて規則を設定することです。いつもたわごとを書いてよこすやつからのメッセージを読んで、今後はそいつからのメッセージを無視しようと決心したならば、*L* を叩いてスコアを下げる規則を設定してください。すると Gnus は、スコアを下げる基準をどれにすれば良いかを尋ねてくるでしょう。*?* を二回叩くと、すべての候補を見ることができますが、ここで選択すべきなのは著者(From ヘッダー) を意味するa です。次に Gnus はどの種類の合致を使うべきかを尋ねてくるので、厳密な合致のための*e* を叩くか、あるいは文字の一部への合致を求める*s* を叩いてから、後ですべてを削除してください。どんな電子メールアドレスであっても、それを持つすべての著者のスコアを下げるために、その名前が使われます。さらに、その規則をいつ適用するべきか、またどれくらいの時間続けるべきかを Gnus に伝える必要があります。*p* をたたくと、その規則を今すぐ適用して、それを永久に持続させます。スコアを下げるのではなくて上げたいのであれば、*L* の代わりに*I* を使ってください。

規則を手で設定することもできます。そうするには概略バッファーで*V f* 命令を使ってください。するとスコアファイルの名前を尋ねられるでしょう。一つのグループだけで有効なのはname.of.group.SCORE で、すべてのグループに対して有効なのがall.SCORE です。厳密な構文については Gnus のマニュアルを参照してください。それは一つの大きなリストで、その要素もまた多くのリストです。その後者の単位リストの第一要素はスコアの対象であるヘッダーで、残りは、何に合致するか、いくらのスコアを割り当てるか、規則を期限切れ消去するのはいつか、そしてどうやって合致させるかを示すもう一つのリストです。私にとても興味を持ったならば、以下のようなものをall.SCORE ファイルに加えてください:

```
(("references" ("hschmi22.userfqdn.rz-online.de" 500 nil s))
 ("message-id" ("hschmi22.userfqdn.rz-online.de" 999 nil s)))
```

これは私が書いたメッセージのスコアに 999 を加えて、それに対する(たぶん間接的な) 回答のメッセージのスコアに 500 を加えるでしょう。もちろん分別のある人は誰も、こんなことはしないでしょうが。:-)

三番目の選択肢は適応スコア付けです。これは、Gnus があなたを観察して、あなたが何に興味を持って何に幻滅するかを見つけ出し、それらを反映した規則を設定してくれるものです。流通量の多いグループを読むときに、適応スコア付けは大きな助けになるでしょう。適応スコア付けをやってみたいならば、~/.gnus.el ファイルで以下の宣言を行なってください:

```
(setq gnus-use-adaptive-scoring t)
```

質問 4.10

いくつかの(例えばメールの) グループで、スレッド表示をさせなくすることはできますか? あるいは、いくつかのグループに固有の変数を設定することができますか?

回答

グループバッファーにいるときに、そのグループにポイントを移動してから*G c* を叩いてください。すると、そのグループのためのオプションを設定することができるバッファーが開きます。そして、そのバッファーのおしまいの方で、そのグループでローカルに変数を設定するための項目を見つけることができるでしょう。スレッド表示をさせなくするには、変数名としてgnus-show-threads を、値としてnil を入力してください。作業を終えたら、そのバッファーの最初の方にある[done] ボタンを叩きましょう。

質問 4.11

私が書いたメッセージとフォロー記事を、強調表示させることはできますか?

回答

「できますか?」なんて質問はやめてください。Gnus 村では答えはいつも yes なのですから。:-) これは三段階の作業になります。まず、それらの投稿のためのフェース(概略行がどんなふうに見えるかの仕様) を作りましょう。続いて、それらにいくばくかの特別なスコアを与えましょう。最後に、その新しいフェースを使うように Gnus に指示しましょう。

質問 4.12

特にメールのグループで、Gnus がグループバッファーに表示するメッセージの合計の数が非常に大きいのです。これはバグですか?

回答

いいえ、これは Gnus の設計上の問題で、これを直すには Gnus のバックエンドの主要な部分を実装し直さなければなりません。Gnus は最大の記事番号から最小の記事番号を減算したものが合計の記事数であると解釈します。これは Usenet のグループでは OK なのですが、メールのグループでたくさんのメッセージを消去したり移動すると計算に失敗します。この病を治療するには*C-u* `RET` でそのグループに入り(この命令は Gnus にすべてのメッセージを持って来させます)、*M P b* ですべてのメッセージに印を付けてから、*B m name.of.group* でメッセージを全部、それらが元あったグループに移動してください。この処理によってそれらは新しいメッセージ番号を持たせられて、合計の記事数は再び正しくなるでしょう(またもやそれらを消したり別のグループに移動するまでは)。

質問 4.13

概略バッファーと記事バッファーの配置が気に入らないのですが、どうやったら変更できますか? できるなら三面で表示させたいです。

回答

`gnus-add-configuration` 関数を呼ぶことによって、ウィンドウの配置を制御することができます。構文はいささか複雑ですが、マニュアルで非常に良く説明されています(see Section “ウィンドウの配置” in *The Gnus Manual*)。いくつかのやさしい例を挙げてみましょう。

概略 25% 記事 75% というディフォルトの割合を、概略 35% 記事 65% の割合に変更します(記事のための 1.0 は、残った空き地を取るという意味です):

```
(gnus-add-configuration
 '(article (vertical 1.0 (summary .35 point) (article 1.0))))
```

グループバッファーが左、概略バッファーが右上、記事バッファーが右下という三面配置です:

```
(gnus-add-configuration
 '(article
   (horizontal 1.0
               (vertical 25
                         (group 1.0))
               (vertical 1.0
                         (summary 0.25 point)
```

```
                        (article 1.0)))))
(gnus-add-configuration
 '(summary
   (horizontal 1.0
               (vertical 25
                         (group 1.0))
               (vertical 1.0
                         (summary 1.0 point)))))
```

質問 4.14

概略バッファーを見せるやり方が好きではありません。調整するには、どうすれば良いですか?

回答

変数gnus-summary-line-format をいじくり回す必要があります。その値は著者、日付、表題などのようなものを表すシンボルの文字列です。使うことができる仕様のリストはsee Section “概略バッファーの行” in *The Gnus Manual* と、しばしば忘れられてしまうsee Section “書法仕様変数” in *The Gnus Manual* およびその下位の各章で見つかるはずです。そこでは書法仕様の概略を使うことができるようにしてくれる、カーソルの位置指定やタブ位置の指定のようなものを見つけることができるでしょう。残念ながら 5.8.8 ではタブ位置を固定させる機能が壊れています。

Gnus は非常にみごとな新しい書法仕様変数を提供しています。例えば %B はスレッド木を描き、また %&user-date は、どう表示するかが記事が発信されてからの経過時間に依存している日付を表示します。これらの両方を使っている例です:

```
(setq gnus-summary-line-format
      ":%U%R %B %s %-60=|%4L |%-20,20f |%&user-date; \n")
```

結果はこんなふうになります:

```
:O     Re: [Richard Stallman] rfc2047.el          |  13 |Lars Magne Ingebrigt |Sat 23:
:O     Re: Revival of the ding-patches list       |  13 |Lars Magne Ingebrigt |Sat 23:
:R  >  Re: Find correct list of articles for a gro|  25 |Lars Magne Ingebrigt |Sat 23:
:O  \->  ...                                      |  21 |Kai Grossjohann      | 0:01
:R  >  Re: Cry for help: deuglify.el - moving stuf|  28 |Lars Magne Ingebrigt |Sat 23:
:O  \->  ...                                      | 115 |Raymond Scholz       | 1:24
:O    \->  ...                                    |  19 |Lars Magne Ingebrigt |15:33
:O     Slow mailing list                          |  13 |Lars Magne Ingebrigt |Sat 23:
:O     Re: '@' mark not documented               |  13 |Lars Magne Ingebrigt |Sat 23:5
:R  >  Re: Gnus still doesn't count messages prope|  23 |Lars Magne Ingebrigt |Sat 23:
:O  \->  ...                                      |  18 |Kai Grossjohann      | 0:35
:O    \->  ...                                    |  13 |Lars Magne Ingebrigt | 0:56
```

質問 4.15

やってきたメールをいろいろなグループに振り分けるには、どうしたら良いですか?

回答

Gnus はメールを分割するための二つの手段として、やさしいnnmail-split-methods と、もっと強力な特級メール分割の機能を提供します。ここでは最初のものだけについて述べ

ますが、後者についてはマニュアル(see Section “特級メール分割” in *The Gnus Manual*)を参照してください。

`nnmail-split-methods` の値はリストで、それぞれの要素は分割の規則を表すリストです。それぞれの規則は「合致する記事が行くべきグループ」と「合致すべき正規表現」の形式を持っていて、最初に合致した規則が勝ちます。最後の規則は常に汎用の規則(正規表現‘`.*`’) でなければなりません。それは他のどんな規則にも合致しない記事が行くべき場所を示します。もしまだフォルダーが無かったならば、それは記事がそこに着地するとすぐに作られるでしょう。ディフォルトでは、メールは規則に合致するすべてのグループに送られます。それを望まないならば(たぶん望まないでしょう)、以下のものを`~/.gnus.el` ファイルに入れてください:

```
(setq nnmail-crosspost nil)
```

たった一つの例は千の言葉を並べるよりも勝ります。そこで、ここでは私の`nnmail-split-methods` を紹介しましょう。私は、私が購読しているいくつかのメーリングリストから来るか、私を直接に目指して来たすべてのメールを濾過器に通すので、私が特別のグループの複写を送ることと、ディフォルトのグループは‘`spam`’ であることに注意してください。これらの規則は、私宛てに届いた spam のおよそ 80% を殺してくれます(電子メールアドレスは spammers がそれらを使用するのを防ぐために変更されます)。

```
(setq
 nnmail-split-methods
 '(("duplicates" "^Gnus-Warning:.*duplicate")
   ("Emacs-devel" "^\\(To:\\|Cc:\\).*localpart@gnu.invalid.*")
   ("Gnus-Tut" "^\\(To:\\|Cc:\\).*localpart@socha.invalid.*")
   ("tcsh" "^\\(To:\\|Cc:\\).*localpart@mx.gw.invalid.*")
   ("BAfH" "^\\(To:\\|Cc:\\).*localpart@.*uni-muenchen.invalid.*")
   ("Hamster-src" "^\\(Cc:\\|To:\\).*hamster-sourcen@yahoogroups.\\(de\\|com\\).*")
   ("Tagesschau" "^From: tagesschau <localpart@www.tagesschau.invalid>$")
   ("Replies" "^\\(Cc:\\|To:\\).*localpart@Frank-Schmitt.invalid.*")
   ("EK" "^From:.*\\(localpart@privateprovider.invalid\\|localpart@workplace.invalid\\
   ("Spam" "^Content-Type:.*\\(ks_c_5601-1987\\|EUC-KR\\|big5\\|iso-2022-jp\\).*")
   ("Spam" "^Subject:.*\\(This really work\\|XINGA\\|ADV:\\|XXX\\|adult\\|sex\\).*")
   ("Spam" "^Subject:.*\\(\=\?ks_c_5601-1987\?\\|\=\?euc-kr\?\\|\=\?big5\?\\).*")
   ("Spam" "^X-Mailer:\\(.*BulkMailer.*\\|.*MIME::Lite.*\\|\\)")
   ("Spam" "^X-Mailer:\\(.*CyberCreek Avalanche\\|.*http\:\/\/GetResponse\.com\\)")
   ("Spam" "^From:.*\\(verizon\.net\\|prontomail\.com\\|money\\|ConsumerDirect\\).*")
   ("Spam" "^Delivered-To: GMX delivery to spamtrap@gmx.invalid$")
   ("Spam" "^Received: from link2buy.com")
   ("Spam" "^Cc: .*azzrael@t-online.invalid")
   ("Spam" "^X-Mailer-Version: 1.50 BETA")
   ("Uni" "^\\(Cc:\\|To:\\).*localpart@uni-koblenz.invalid.*")
   ("Inbox" "^\\(Cc:\\|To:\\).*\\(my\ name\\|address@one.invalid\\|address@two.invalid
   ("Spam" "")))
```

質問 4.16

HTML メールを見るときコントラストをより良くするには、どうすれば良いですか?

回答

Gnus に組み込まれているシンプルな HTML 描画器(`mm-text-html-renderer` が`shr` であるときに使うもの) は、HTML メールの中で指定されている色を使いますが、黒色の背景に対してダークグレイのテキストを表示してしまうような状況を避けるために、それらの色を調整します。しかし、それでもまだコントラストが低過ぎるなら、変数`shr-color-visible-distance-min` および`shr-color-visible-luminance-min` の値を増やしてください。

12.9.6 メッセージの作成

質問 **5.1**

メールを送信したりニュース記事を投稿するために知っている必要がある基本的なコマンドは何ですか?

回答

新しいメールの作成を始めるには、グループバッファーか概略バッファーのどちらかで*m* を叩いてください。ニュース記事を投稿する場合には、グループバッファーで`a` を叩いた後に手で Newsgroups ヘッダーに書き込むか、投稿するグループの概略バッファーで`a` を叩いてください。

メールで返信する場合、元の記事を引用しないか後で手作業で引用するつもりならば*r* を、引用文を最初から取り込んでしまうのならば*R* を使ってください。ニュースグループでフォロー記事を投稿する場合であれば*f* か*F* (*r* と*R* の関係に似ています) を使いましょう。

新たにヘッダーを挿入するならば‘`--text follows this line--`’ の上に書いて、本文はその下に書いてください。書き上がったメッセージを送信するコマンドは*C-c C-c* で、後で仕上げるために‘`drafts`’ グループに保存するためのコマンドは*C-c C-d* です。後者は*D e* で再び編集することができます。

質問 **5.2**

メールを作成するとき、自動的に行を折り返すにはどうすれば良いですか?

回答

No Gnus の場合は自動で行を折り返す機能がディフォルトで ON になります。変数`message-fill-column` を参照してください。

他の Gnus の版では、こんなものを`~/.gnus.el` ファイルに入れてください:

```
(unless (boundp 'message-fill-column)
  (add-hook 'message-mode-hook
            (lambda ()
              (setq fill-column 72)
              (turn-on-auto-fill))))
```

いつもの通りに*M-q* を使うことによって、段落を再整形することができます。

質問 **5.3**

From, Organization, Reply-To などのヘッダーを自動生成したり、定型の署名(signature) を自動的に挿入するには、どうしたら良いですか?

回答

他のやり方もありますが、これのためには投稿様式(posting styles) を使うべきです(理由は後で書きます)。この例ならば、その構文が明確にわかるはずです:

```
(setq gnus-posting-styles
      '((".*"
         (name "Frank Schmitt")
         (address "me@there.invalid")
         (organization "Hamme net, kren mer och nimmi")
         (signature-file "~/.signature")
         ("X-SampleHeader" "foobar")
         (eval (setq some-variable "Foo bar")))))
```

".*" という設定はディフォルトのものです([FAQ 5-4], p. 400, 参照)。それ以降のリストの第一要素に使える値(属性名) はsignature, signature-file, organization, address, name またはbody です。属性名は文字列でも構いません。その場合、属性名はヘッダー名として使われ、その値は記事のヘッダーに挿入されます。ただし値がnil だったら、その名前のヘッダーは削除されます。

(eval (foo bar)) の形式を使うことも可能で、その場合bar を引数に与えられて関数foo が評価され、結果は捨てられます。

質問 5.4

投稿するグループによって異なる From ヘッダーや署名を自動挿入するには、どうすれば良いですか?

回答

これこそが投稿様式(posting styles) の強みです。前の回答(see [FAQ 5-3], p. 399) では、すべてのグループのためのディフォルトを設定するために".*" を使いました。これに"^gmane" のような正規表現を使うと、それ以降の設定を gmane ニュースグループの階層で投稿する記事だけに適用させることができます。代わりに".*binaries" を使うと、名前などが'binary' という文字列を含んでいるグループに投稿する記事だけに、それらの設定が適用されます。

正規表現の代わりに関数を指定することもでき、それが評価されて真を返すときだけ、それに対応する設定が有効になります。これの興味深い二つの候補は、現在のグループがニュースグループだったらt を返すmessage-news-p と、それと対になるmessage-mail-p です。

すべての合致する様式が適用されることに注意してください。以下を例にすれば、'gmane.mail.spam.spamassassin.general' に投稿すると".*" で始まる設定が適用され、message-news-p 以下の設定も適用され、"^gmane" で始まる設定、それに"^gmane\\.mail\\.spam\\.spamassassin\\.general$" 以下の設定のすべてが適用されるということです。このため、一般的な設定は先頭に置き、特定の条件を持つものは下の方に置くのが良いでしょう。

```
(setq
 gnus-posting-styles
 '((".*" ;;default
    (name "Frank Schmitt")
    (organization "Hamme net, kren mer och nimmi")
```

```
 (signature-file "~/.signature"))
((message-news-p) ;;Usenet news?
 (address "mySpamTrap@Frank-Schmitt.invalid")
 (reply-to "hereRealRepliesOnlyPlease@Frank-Schmitt.invalid"))
((message-mail-p) ;;mail?
 (address "usedForMails@Frank-Schmitt.invalid"))
("^gmane" ;;this is mail, too in fact
 (address "usedForMails@Frank-Schmitt.invalid")
 (reply-to nil))
("^gmane\\.mail\\.spam\\.spamassassin\\.general$"
 (eval (set (make-local-variable 'message-sendmail-envelope-from)
            "Azzrael@rz-online.de")))))
```

質問 5.5

スペルチェッカーはありますか? できれば、書いたその場でチェックしてくれるものがあれば良いのですが。

回答

Emacs では単語の綴りをチェックするためにispell.el を、書いたその場でスペルチェックをするためにはflyspell.el を使うことができます。したがって最初に行なうのは、hunspell (https://hunspell.github.io/)、ispell (https://www.cs.hmc.edu/~geoff/ispell.html) またはaspell (http://aspell.net) を適当な path にインストールしておくことです。

ispell.el は外部プログラム'ispell' を使うことを想定しています。'aspell' を選ぶなら、Emacs の設定ファイルで以下の宣言を行なってください(訳注: Emacs 22 以上では、ispell.el がそれらを自動判別します):

```
(setq ispell-program-name "aspell")
```

送出するメッセージがスペルチェックされるようにしたいなら以下の行を

```
(add-hook 'message-send-hook 'ispell-message)
```

単語を書いたその場でスペルチェックすることを好むなら次の行を

```
(add-hook 'message-mode-hook (lambda () (flyspell-mode 1)))
```

~/.gnus.el ファイルに追加してください。

質問 5.6

投稿するグループに基づいて、辞書(スペルチェッカーの) を切り替えることはできますか?

回答

はい、できます。こんなものを~/.gnus.el ファイルに入れてください:

```
(add-hook 'gnus-select-group-hook
          (lambda ()
            (cond
             ((string-match
               "^de\\." (gnus-group-real-name gnus-newsgroup-name))
              (ispell-change-dictionary "deutsch8"))
```

```
                (t
                 (ispell-change-dictionary "english")))))
```

"^de\\." と"deutsch8" は、必要に応じて変更してください。

質問 5.7

全員の電子メールアドレスを思い出さなくても済むようにするための、アドレス帳のようなものはありますか?

回答

そのための非常に基本的な解は「メールの別名」(mail aliases) です。以下のように単純な別名の構文を使って、~/.mailrc ファイルにメールアドレスを登録しておくことができます:

```
alias al        "Al <al@english-heritage.invalid>"
```

そうしておいて、メッセージバッファーの To: または Cc: 行で、別名に続いてスペースか句読点をタイプすることによって、Gnus に完全なアドレスを挿入してもらうことができます。詳細は Message マニュアル(see Section “メールの別名” in *The Message Manual*) を参照してください。

でも、あなたが本当に使いたいのは BBDB (the Insidious Big Brother Database) でしょう。bbdb のウェブサイト (https://bbdb.sourceforge.net/) から入手してください。そして Gnus で BBDB を有効にするために、以下のものを~/.gnus.el ファイルに書き込んでください:

```
(require 'bbdb)
(bbdb-initialize 'gnus 'message)
```

さて、いくつかの一般的な BBDB の設定が必要かもしれません。それらは~/.emacs ファイルに置きましょう。例です:

```
(require 'bbdb)
;; 北アメリカに住んでいるのでなければ、以下によって電話番号の
;; チェックをやめさせるべきです。
(setq bbdb-north-american-phone-numbers-p nil)
;; あなたの電子メールアドレスを BBDB に教えましょう。
(setq bbdb-user-mail-names
      (regexp-opt '("Your.Email@here.invalid"
                    "Your.other@mail.there.invalid")))
;; メールアドレスを補完するときに、候補をエンドレスで出します。
;; cycling while completing email addresses
(setq bbdb-complete-name-allow-cycling t)
;; BBDB のバッファーをポップアップさせません。
(setq bbdb-use-pop-up nil)
```

これで BBDB を使う準備ができたはずです。*M-x bbdb* RET RET で、すべての登録した項目を表示する BBDB のバッファーを開いてください。新たに登録するには*c*、検索には*b*、そして登録してある項目に新しいフィールドを加えるには*C-o* を使いましょう。送信者を BBDB に登録するには、概略バッファーのその記事の場所で単に*:* を叩くだけで、あなたの仕事は終りです。一方、新規にメールを作成しているときに*TAB* を叩くことによって、順繰りに現れる候補の中から受取人を選ぶことができます。

質問 5.8

記事バッファーの上の方で、ときどき小さな画像を目にします。あれは何ですか? また、どうしたら私も投稿するときに付けることができますか?

回答

あの画像は X-Face というものです。ヘッダー行で 48×48 画素の白黒画像がエンコードされています。それを送信する記事に含めたいなら、何かの画像を X-Face に変換する必要があります。そうするには、何らかの画像を加工するためのプログラム(例えば Gimp) に点火して、記事に含めたい画像のファイルを開き、必要な部分を切り抜いて、色の深度を 1-bit まで減らし、48×48 の大きさに縮小または拡大して、bitmap としてファイルに保存してください。次に‘compface’ パッケージをこのサイト (ftp://ftp.cs.indiana.edu/pub/faces/) から入手して、以下を実行することによって実際の X-Face を作りましょう:

```
cat file.xbm | xbm2ikon | compface > file.face
cat file.face | sed 's/["\\]/\\&/g' > file.face.quoted
```

‘compface’ を使うことができなくても、オンラインの X-Face 変換器がhttps://www.dairiki.org/xface/ にあります。MS Windows を使っているのならば、かつて‘http://www.xs4all.nl/~walterln/winface/’ にあった‘WinFace’ プログラムを使うこともできます。後は、送信する記事に X-Face を含めてくれるように Gnus に指示するだけです。それには~/.gnus.el ファイルに以下のようなものを入れてください:

```
(setq message-default-headers
      (with-temp-buffer
        (insert "X-Face: ")
        (insert-file-contents "~/.xface")
        (buffer-string)))
```

あるいは、単に次の項を

```
(x-face-file "~/.xface")
```

gnus-posting-styles に加えてください。

質問 5.9

ときどきニュースグループで、*f* の代わりにうっかり*r* を打ってしまいます。ニュースグループなのにもかかわらずメールで返信しようとしたときに、Gnus に警告してもらうことはできますか?

回答

もう Gnus 5.10 を使っているのであれば、これを~/.gnus.el ファイルに入れてください:

```
(setq gnus-confirm-mail-reply-to-news t)
```

質問 5.10

Gnus が sender ヘッダーを生成しないようにするには、どうしたら良いですか?

回答

ディフォルトでは Gnus は sender ヘッダーを作りません。

質問 5.11

送信したメールやニュースの控えをローカルに残しておきたいのですが、どうすれば良いですか?

回答

それを行なうためには変数gnus-message-archive-group を設定しなければなりません。それには、控えを保存しておくグループの名前を与える文字列を設定することができます。あるいは以下の例のように、評価されるとグループ名を返す関数(訳注: と言うよりは Lisp の式) を設定することもできます。

```
(setq gnus-message-archive-group
      '((if (message-news-p)
            "nnml:Send-News"
          "nnml:Send-Mail")))
```

質問 5.12

送信に成功した後でそのバッファーを"Sent mail to..." として生かしたままにしておく代わりに kill するにはどうしたら良いですか?

回答

~/.gnus.el ファイルにこれを加えてください:

```
(setq message-kill-buffer-on-exit t)
```

質問 5.13

Message-ID が不正だと言われてしまうのですが、それはなぜですか? また、どうやって直したら良いですか?

回答

Message-ID は、送信したメッセージが、それがそれであることを確認するためのものです。それが、あるメッセージにとって固有であるようにするために、Gnus は‘@’ の後に置くマシン名を知る必要があります。Gnus が走っているマシンの名前が適切ではないならば(ほとんどの個人のマシンが該当するでしょう)、Gnus が使う名前を以下のように~/.gnus.el ファイルで設定してください:

```
(setq message-user-fqdn "yourmachine.yourdomain.tld")
```

Gnus 5.9 かそれ以前のものを使っている場合は、代わりにこれを使ってください(新しい版でも動作します):

```
(with-eval-after-load "message"
  (let ((fqdn "yourmachine.yourdomain.tld"));; <-- 変更してね!
    (if (boundp 'message-user-fqdn)
        (setq message-user-fqdn fqdn)
      (gnus-message 1 "Redefining `message-make-fqdn'.")
      (defun message-make-fqdn ()
        "Return user's fully qualified domain name."
        fqdn))))
```

"yourmachine.yourdomain.tld" に何を入れるかを決めることができないならば、複数の選択肢があります。あなたが‘yourUserName.userfqdn.provider.net’ のようなものを使っ

ても良いかどうかをプロバイダーに尋ねてみても良いし、個人でドメイン‘yourdomain.tld’を保有している場合は‘somethingUnique.yourdomain.tld’のようなものを使うことができます。あるいは、ユーザーに無料で FQDN を提供してくれるサービスに登録するのも良いでしょう。

最後に、Gnus にニュース記事では Message-ID をまったく作らせない(そしてその仕事はサーバーに任せる) ことができます。以下の設定を使ってください:

```
(setq message-required-news-headers
      (remove' Message-ID message-required-news-headers))
```

さらに次の設定によって、メールでも Gnus に Message-ID を作らせないことは可能です:

```
(setq message-required-mail-headers
      (remove' Message-ID message-required-mail-headers))
```

しかしながら、メールサーバーのあるものは適切な Message-ID を作ってくれないので、自分でメールを送信して Message-ID を眺めてみることによって、あなたのメールサーバーが正しく振る舞うかどうかをテストしてください。

12.9.7 古いメッセージ

質問 6.1

Gnus を使っていなかったときの古いメールを、Gnus に編入させるにはどうしたら良いですか?

回答

最も楽な方法は、古いメールプログラムにメッセージを‘mbox’形式にまとめてもらうことです。ほとんどの Unix のメイラーはそれをすることができ、あなたが MS Windows の世界の出身であってもhttps://sourceforge.net/projects/mbx2mbox/ で、そのための道具を見つけることができるでしょう。

では、この‘mbox’ファイルを Gnus に編入させましょう。それにはグループバッファーで*G f /path/file.mbox* RET を実行して、‘mbox’ファイルを扱うためのnndoc グループを作ってください。そうすれば、あなたのメールに読み出し専用でアクセスできるようになります。そのメッセージを通常の Gnus のメールグループの階層に編入させたいならば、たった今作ったnndoc グループに*C-u* RET 命令で入って(つまりすべてのメッセージが読めるようにしておいて)、すべてのメッセージに*M P b* で印を付けてから、それらをお望みのグループに*B c name.of.group* RET でコピーするか、またはnnmail-split-methods を使ってそれらを分割(再スプール) するために*B r* を使ってください。

質問 6.2

興味を持った記事を保存するにはどうすれば良いですか?

回答

例えば‘gnu.emacs.gnus’で興味深い記事に出くわして、それを保存しようと思ったときには、複数の解があります。第一の最もやさしい方法は、*O f* 命令を使ってそれをファイルに保存することです。でもそれは、Gnus から保存されたメッセージをより直接にアクセスする手段としては、とても便利だとは言えないのではないでしょうか? それに同意してくれるのならば、Frank Haun pille3003@fhaun.de が書いてくれたこのコードの切れ端を~/.gnus.el ファイルに入れてみてください:

```
(defun my-archive-article (&optional n)
```

```
  "一個以上の記事を、対応する `nnml:' グループにコピーします。
例えば `gnus.ding' の記事は `nnml:1.gnus.ding' グループに行き、
`nnml:List-gnus.ding' の記事は `nnml:1.List-gnus-ding' に行きます。

一個以上の記事を保存するには、概略バッファーでプロセス/接頭引数の
習慣を使ってください。"
  (interactive "P")
  (let ((archive-name
         (format
          "nnml:1.%s"
          (replace-regexp-in-string "^.*:" "" gnus-newsgroup-name))))
    (gnus-summary-copy-article n archive-name)))
```

訳注: See Section “プロセス/接頭引数” in *The Gnus Manual*.

これにより、概略バッファーで*M-x my-archive-article* を実行すれば、指定した記事を`nnml` グループに保存することができます(nnml を好みのバックエンドに変えてください)。

もちろん、以下の設定によってキャッシュを常に有効にすることもできます:

```
(setq gnus-use-cache t)
```

これによって、維持したい記事には可視(ticked) か保留(dormant) の印を付けるだけで済むようになります。また、既読(read) の印を付けることによって、それらの記事はキャッシュから削除されます。

質問 6.3

指定したメッセージを探すにはどうしたら良いですか?

回答

これにも複数の方法があります。Usenet グループに投稿されたものについては、おそらくgroups.google.com (`https://groups.google.com`) に尋ねるのが最も楽な解決方法です。そこで見つかったならば、Google に生の記事を表示してもらって Message-ID を探し、概略バッファーで*M-^ the@message.id* `RET` を実行してください。さらに‘`groups.google.com`’へのインターフェースがあるので、グループバッファーで*G W* を使うこともできます。

メールとニュースの両方のグループで動作するもう一つの方法は、探しているメッセージが存在するグループに入って、Emacs の標準の探索コマンドである*C-s* を使うことです。それは壊れたスレッドにある記事を探すためにも十分に賢いものです。本文の中で探したいのならば、代わりに*M-s* を試してください。さらに付け加えると、これも役に立つgnus-summary-limit-to-foo 関数群があります。

質問 6.4

要らなくなった古いメールを削除するにはどうすれば良いですか?

回答

もはや要らなくなったメールにポイントを置いて`#` で印を付けてから、それらを*B* `DEL` で永久に消してしまうことは、もちろんできます。さらに、それらを実際に消してしまう代わりに*B m nnml:trash-bin* を使ってジャンク・グループ(それは時々消さなければなりません)に送ることもできるのですが、両方とも Gnus が意図するやり方ではありません。

Gnus では、ニュースサーバーでニュース記事が期限切れ消去されるように、メールも消します。そうするためには、概略バッファーのそのメールの上で*E* を叩くことによって、そのメールが期限切れ消去可能であることを(「このメールはもう要らないよ」と) Gnus に伝えれば良いのです。すると、そのグループを抜け出たときに、Gnus は以前に印が付けられたすべてのメッセージを検査して、十分に古くなった(ディフォルトでは一週間より古くなった) メールを消去します。

質問 6.5

読み終わったすべてのメッセージを期限切れ消去したいのですが(少なくともいくつかのグループで)、どうしたら良いですか?

回答

読み終わったすべてのメッセージが期限切れ消去されるようにしたいなら(例えばオンラインのアーカイブが別に存在しているメーリングリストで)、`auto-expire` と`total-expire` という二つの選択肢があります。`auto-expire` というのは、印が付いていないけれども読むために選択されたことがあるすべての記事に期限切れ消去可能の印が付けられることで、メッセージを読むたびに Gnus が*E* を叩いてくれるようなものです。`total-expire` は少し異なるやり方に従っていて、既読の印が付けられたすべての記事が期限切れ消去可能になります。

`auto-expire` を有効にするには、そのグループのグループパラメーターに`auto-expire` を含めてください(グループバッファーでグループパラメーターを変更するグループの上にポイントを置いて*G c* を叩きます)。`total-expire` の方は、グループパラメーターに`total-expire` を加えてください。

どちらの手段を選ぶかは、単に好みの問題です。`auto-expire` は速いのですが、適応スコア付けと共存できないので、この機能を使いたい場合は`total-expire` を使わなければなりません。

`total-expire` か`auto-expire` が設定されているグループで、あるメッセージを期限切れ消去の対象から外したい場合には、*u* で可視(ticked) の印を付けるか`?` で保留(dormant) の印を付けてください。`auto-expire` を使っている場合は、さらに*d* で既読の印を付けることができます。

質問 6.6

期限が来たメールを削除するのではなくて、他のグループに移動させたいのですが。

回答

`~/.gnus.el` ファイルに、このようなものを書き込んでください:

```
(setq nnmail-expiry-target "nnml:expired")
```

(`nnmail-expiry-target` にグループによって異なる値を設定したい場合には、質問「いくつかの(例えばメールの) グループで、スレッド表示をさせなくすることはできますか? あるいは、いくつかのグループに固有の変数を設定することができますか?」(see [FAQ 4-10], p. 395) を参照してください。)

12.9.8 ダイアルアップ環境で Gnus を使う

質問 7.1

ネットに常時接続していないのですが、どうしたら接続する時間を最小限にできますか?

回答

二つのやり方があります。一つは Gnus エージェントを使うこと(see [FAQ 7-2], p. 408) で、もう一方はニュース記事とメールをローカルディスクに取り込むプログラムをインストールして、Gnus にそれらをローカルマシンから読んでもらう方法です。

二つ目のやり方で行きたいのなら、ニュース記事を取り込んでそれらを Gnus に渡すプログラム、同じことをメールに対して行なうプログラム、およびあなたが書いたメールを Gnus から受け取って、オンラインになったらそれらを送信するプログラムが必要です。

最初に Unix システムについて書きましょう。ニュース記事を取り込むための最も安易な解は、Leafnode (`https://www.leafnode.org/`) やsn (`https://patrik.iki.fi/sn/`) のように小規模なNNTP サーバーを使うことです。もちろんinn (`https://www.isc.org/othersoftware/`) のような本格的なニュースサーバーをインストールすることもできます。メールの取り込みに使うもので人気があるのはfetchmail (`https://www.fetchmail.info/`) およびgetmail (`https://pyropus.ca/software/getmail/`) です。それらにメールをディスクに書き込むように指示してください。Gnus はそこから読むことになります。最後ですがおろそかにできないのはメールの送信です。これにはあらゆる MTA、例えばsendmail (`http://www.sendmail.org/`) またはexim (`http://www.exim.org/`) を使うことができます。

Windows 小屋のためにはHamster (`http://www.tglsoft.de/freeware_hamster.html`) を推します。それは小さくてフリーなオープンソースのプログラムで、遠隔サーバーからメールとニュースを取り込んで、NNTP およびPOP3 またはIMAP のそれぞれを介して Gnus (あるいは他のメール/ニュースリーダー) に渡します。さらにそれは Gnus が送信するメールを受け取るためのSMTP サーバーも含んでいます。

質問 **7.2**

ならば、それのエージェントに関係するものは何ですか?

回答

Gnus エージェントは Gnus の一部で、メールとニュースを取り込んでディスクに格納し、後でオフラインのときにそれらを読むことができるようにします。それは、言うならば Forte Agent のようなオフライン・ニュースリーダーの真似をします。ディフォルトで有効になっています。

```
(setq gnus-agent t)
```

数あるグループをローカルに格納することができるサーバーを選ぶ必要があります。そのためには、サーバーバッファーを開いて(グループバッファーで^ を押して) ください。そして、選んだサーバー名の場所にポイントを移動させましょう。最後に`J a` をタイプして、そのサーバーをエージェント化してください。もし間違ってしまったら、あるいは気が変わったら、`J r` をタイプすれば元に戻すことができます。終わったら`q` をタイプして、グループバッファーに戻ってください。次回エージェント化されたサーバーのグループに入るとヘッダーがディスクに格納され、その次の回にグループバッファーに入ると、そこ(ディスク) から読むようになります。

質問 **7.3**

記事の本文もディスクに格納したいのですが、どうすれば良いですか?

回答

ある述語を満足させる記事の本文を自動的に取り込むように、エージェントに指示することができます。それは、グループバッファーで`J c` 命令を使うことによって行くことができる、特別なバッファーで行ないます。どの述語を使うことができるか、およびそれを正しく行なう方法に関する情報については Gnus のマニュアルを参照してください。

さらに、どの記事をディスクに格納するかを、手作業でエージェントに指示することもできます。それには二つのやり方があります: 1 つ目は、概略バッファーで記事の上にカーソルを置いて`#` とタイプし、エージェントに保存する一連の記事にプロセス印を付けた後に`J s` をタイプします。もう 1 つできるのは、概略バッファーで目的の記事にカーソルを置いて`@` をタイプしてから`J u` をタイプして、ダウンロード可能な(%) 印を設定することです。どんな違いがあるのかって? えーと、ダウンロード可能印が永久的なのに対して、プロセス印は概略バッファーを抜け出たとたんに消される点です。実際、複数のグループでダウンロード可能印を設定して、それらすべての記事を(グループバッファーで`J s` 命令を使うことによって) まとめて取り込むことができます。唯一の欠点は、エージェント化されているサーバーの、選択されたすべてのグループのヘッダーをも取り込んでしまうことです。ヘッダーの量にもよりますが、最初の取り込みには何時間もかかるかもしれません。

質問 **7.4**

オフラインのときに Gnus にメールやニュース記事を送信させないようにするには、どうしたら良いですか?

回答

現在の状態がオンライン(plugged) かオフライン(unplugged) かを、Gnus に言ってあげなければなりません。それ以外のもろもろのことは自動的にやってくれます。plugged と unplugged の切り替えは、グループバッファーで`J j` 命令を実行することによって行なうことができます。Gnus を unplugged の状態で起動したいなら、`M-x gnus` の代わりに`M-x gnus-unplugged` を使ってください。ただし、これが働くためにはエージェントを有効にしなければならないことに注意してください。

12.9.9 助けを得る

質問 **8.1**

Emacs の中で情報を探したり助けを求めるには、どうすれば良いですか?

回答

最初に立ち寄るべき場所は Gnus マニュアルです(`C-h i d m Gnus RET` で Gnus マニュアルを開いたら、メニューを眺め回すか`s` でテキスト全体を探してください)。一方 Emacs には標準のヘルプ・コマンドがあって、それには`C-h` ともう一つのキーを使います(`C-h ? ?` をタイプすると、利用可能なヘルプ・コマンドとそれらの意味が現れます)。さらに`M-x apropos-command` で利用可能なすべてのコマンドを、`M-x apropos` で存在する変数を探すことができます。

質問 **8.2**

Gnus のマニュアルの中で、あることがら(例えば添付ファイル、PGP、MIME...) に関する情報を、何も見つけることができません。それらは書かれていないのですか?

回答

Gnus マニュアル以外に message、emacs-mime、および EasyPG アシスタントのマニュアルがあります。それらのパッケージは Gnus とともに配布され、Gnus によって使われますが、真に Gnus の核心部分ではないので別の info ファイルで文書化されています。あなたはそれらのマニュアルも覗いてみるべきでしょう。

質問 8.3

私はどのウェブサイトを知っていなければなりませんか。

回答

最も重要なものは公式 Gnus ウェブサイト (`https://www.gnus.org`) です。

他にも興味深いサイトがあったら教えてください。

質問 8.4

どんなメーリングリストとニュースグループがありますか。

回答

Gnus の一般的な質問を取り扱うニュースグループとして‘`gnu.emacs.gnus`’ があります。開発版の Gnus に対する質問があるのなら、ding メーリングリストで尋ねる方が良いはずです。以下を参照してください。

Big8 (訳注: comp, humanities, misc, news, rec, sci, soc, talk の八大ニュースグループ) にとどまっていたいのならば、‘`news.software.readers`’ もまた、いくらかの Gnus のユーザーによって読まれています(でも、的確な助けを得ることができる見込みという点では、上記のグループの方がはるかに良いでしょう)。ドイツ語を話す人向けには‘`de.comm.software.gnus`’ があります。

ding メーリングリスト(ding@gnus.org) は Gnus の開発を扱います。

質問 8.5

バグはどこに報告すれば良いですか?

回答

`M-x gnus-bug` 命令を使うことによってgnus bug メーリングリスト に送るメッセージを書き始めることができます。それにはあなたを助けやすくするための、あなたの環境に関する情報が添付されます。(訳注: 2005 年の暮れの時点では、このメーリングリストはまともに機能していません。)

質問 8.6

その場で即座に得られる助け(real-time help) が欲しいのですが、どこで見つかりますか?

回答

あなたのIRC を‘`irc.libera.chat`’ の‘`#gnus`’ チャンネルに接続してください。

12.9.10 Gnus をチューンする

質問 9.1

Gnus の起動が本当に遅いのですが、どうしたら速くすることができますか?

回答

この原因は、Gnus が active ファイルを読む方法に関係していることがあり得ます。Gnus マニュアルのsee Section “アクティブファイル” in *The Gnus Manual* を参照してください。そこには、その処理を速くするために試してみる価値があることがらがあります。もう一つの案は、`~/.gnus.el` ファイルをバイトコンパイルすること(*M-x byte-compile-file* RET *~/.gnus.el* RET を実行すること) です。最後に、`~/.gnus.el` ファイルで`require` を使っているならば、それらを`with-eval-after-load` で置き換えることができるでしょう。それは対象のモジュールを起動時ではなく、必要になったときに読み込むようにします。あなたの`~/.gnus.el` ファイルに以下のようなものがある場合には、

```
(require 'message)
(add-to-list 'message-syntax-checks '(sender . disabled))
```

Gnus を起動したとたんに`message.el` が読み込まれます。これを次のように置き換えると、

```
(with-eval-after-load "message"
  (add-to-list 'message-syntax-checks '(sender . disabled)))
```

それは必要になったときに読み込まれるようになります。

訳注: 原典の例は不適切です。現在の Gnus は多くの主要なモジュールを起動時に load するようになっていて、例えば`message.el` モジュールは`~/.gnus.el` ファイルを読み込む時点ですでに load されています。ですから少なくとも`message.el` モジュールに対しては`require` も`with-eval-after-load` も使う必要がありません。実害はありませんが何の効果もありません。変数`message-syntax-checks` に項目を追加する上記の例は、単に以下のように書けば十分です。

```
(add-to-list 'message-syntax-checks '(sender . disabled))
```

質問 **9.2**

グループに入るときの動作を速くするには、どうすれば良いですか?

回答

スピードの殺し屋は、`nil` 以外の値を設定された`gnus-fetch-old-headers` 変数です。ですからスピードが重要なのであれば、それをやらないでください。

`~/.emacs` に以下のようなものを書き込むことによって`gc-cons-threshold` の値を増やすことができます。

```
(setq gc-cons-threshold 3500000)
```

質問 **9.3**

メールの送信が遅くなってしまいました。何が起きたのでしょう?

回答

その理由は、`gnus-message-archive-group` 変数の設定によって、送信メッセージの控えを保存する場所や方法を変更したせいかもしれません。アーカイブグループの代わりに`nnml` グループを使ってみてください。そうすれば正常な速度が戻ってくるはずです。

12.9.11 用語集

~/.gnus.el `~/.gnus.el` という用語が使われるとき、それは Gnus の設定ファイルを指します。それを`~/.gnus` と呼ぶか、あるいは別名を指定しても構いません。

Back End Gnus の用語大系では、バックエンドは Gnus の核心部分と実際のサーバーの間の層に位置を占める仮想サーバーのことです。実際のサーバーとは、NNTP サーバー、POP3 サーバー、IMAP サーバー、または、何であれ「メッセージの取得」や「ヘッダーの取得」を行なうサーバーの機能を持つ関数のことです。

Message この FAQ で「メッセージ」はそれがどんな種類のものでも、メール、または Usenet ニュースグループか他の何らかの特製のバックエンドに送られるニュース記事のどちらかを意味します。

MUA MUA は Mail User Agent の頭字語で、電子メールを読んだり書いたりするためのプログラムのことです。

NUA NUA は News User Agent の頭字語で、Usenet ニュースを読んだり書いたりするためのプログラムのことです。

13 GNU フリー文書利用許諾契約書

訳注: 非公式な日本語訳 (`http://www.opensource.jp/fdl/fdl.ja.html.euc-jp`) があります。

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.
`http://fsf.org/`

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

0. PREAMBLE

   The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

   This License is a kind of ‘‘copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

   We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

   This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The ‘‘Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as ‘‘you”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

   A ‘‘Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

   A ‘‘Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The ‘‘Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The ‘‘Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A ‘‘Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not ‘‘Transparent” is called ‘‘Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The ‘‘Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, ‘‘Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The ‘‘publisher” means any person or entity that distributes copies of the Document to the public.

A section ‘‘Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as ‘‘Acknowledgements”, ‘‘Dedications”, ‘‘Endorsements”, or ‘‘History”.) To ‘‘Preserve the Title” of such a section when you modify the Document means that it remains a section ‘‘Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

   You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

   You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

   If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

   If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

   If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

   It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

   You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled ‘‘History”, Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled ‘‘History” in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the ‘‘History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled ‘‘Acknowledgements” or ‘‘Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled ‘‘Endorsements”. Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled ‘‘Endorsements” or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at

your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled ‘‘Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties---for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

   You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

   The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

   In the combination, you must combine any sections Entitled ‘‘History” in the various original documents, forming one section Entitled ‘‘History”; likewise combine any sections Entitled ‘‘Acknowledgements”, and any sections Entitled ‘‘Dedications”. You must delete all sections Entitled ‘‘Endorsements.”

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an ‘‘aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled ‘‘Acknowledgements”, ‘‘Dedications”, or ‘‘History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

   Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `http://www.gnu.org/copyleft/`.

    Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License ‘‘or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

    ‘‘Massive Multiauthor Collaboration Site” (or ‘‘MMC Site”) means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A ‘‘Massive Multiauthor Collaboration” (or ‘‘MMC”) contained in the site means any set of copyrightable works thus published on the MMC site.

    ‘‘CC-BY-SA” means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

    ‘‘Incorporate” means to publish or republish a Document, in whole or in part, as part of another Document.

    An MMC is ‘‘eligible for relicensing” if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

    The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with. . . Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

14 Index

$

$ 299

%

% 14, 224
%(, %) 275
%<<, %>>, guillemets 275
%{, %} 275

*

* 14, 92

.

.newsrc 8
.newsrc.el 8
.newsrc.eld 8

/

/ 92

<

< 70

>

> 70

A

activating groups 44, 357
active file 10, 357
adapt file group parameter 26
adaptive scoring 244
admin-address 26
adopting articles 70
advertisements 93
agent 216
agent expiry 226
Agent Parameters 218
agent regeneration 227
ANSI control sequences 96
archived messages 137
article 356
article backlog 81
article buffer 125
article buffers, several 133
article caching 79
article customization 129
article emphasis 91
article expiry 181
article hiding 92
article history 55
article marking 62
article pre-fetch 77
article scrolling 56
article series 86
article signature 102
article threading 69
article ticking 62
article washing 94
`article-de-quoted-unreadable` 186
articles 1
asterisk 92
asynchronous article fetching 77
Atom 202
attachments 103
attachments, selection via dired 324
authentication 151
authinfo 151
auto-expire 25
auto-save 9

B

Babyl 204
back end 355
backlog 81
backup files 194
banner 28, 93
batch scoring 236
Bayesian spam filtering, naive 314
BBDB whitelists, spam filtering 306
BBDB, spam filtering 306
binary groups 112
blackholes, spam filtering 307
blacklists, spam filtering 304
BNF 366
body 356
body split 176
bogofilter, spam filtering 308
‘`bogus`’ group 164
bogus groups 34, 357
bookmarks 63
bouncing mail 58
broken-reply-to 25
browsing servers 34
browsing the web 198
bugs 361
`bury-buffer` 81
button levels 100
buttons 98

C

caching ... 79
calendar ... 212
canceling articles ... 60
changing servers ... 7
characters in file names ... 326
charset ... 27
charset, view article with different charset ... 56
charsets ... 106
child ... 4, 358
ClariNet Briefs ... 22
closing servers automatically ... 324
cloud ... 323
cloud, download ... 323
coding system aliases ... 107
colophon ... 354
colors ... 282
comment ... 27
composing mail ... 57
composing messages ... 135
composing news ... 59
configuring search ... 257
contributors ... 330
converting kill files ... 252
copy mail ... 114
creating search groups ... 259
cross-posting ... 121
crosspost ... 165
crosspost mail ... 114
crossposting ... 59
crossposts ... 249
customizing ... 22
customizing nndiary ... 214
customizing threading ... 70

D

daemons ... 284
date ... 241
DCC ... 294
decays ... 255
decoding articles ... 86
dejanews ... 198
delayed sending ... 61
`delete-file` ... 174
deleting headers ... 125
deleting mail ... 114
demons ... 284
describing groups ... 44
desktop notifications ... 285
diary ... 212
diary articles sorting ... 215
diary group parameters ... 216
diary headers generation ... 215
diary summary buffer line ... 215
diary summary line format ... 215
diary summary lines sorting ... 215
digest ... 358
digests ... 60
ding Gnus ... 328
ding mailing list ... 362
direct connection functions ... 154
directory groups ... 202
dired ... 324
disk space ... 360
display ... 26
display-time ... 282
documentation group ... 204
don't panic ... 1
drafts ... 143
dribble file ... 9
duplicate mails ... 186

E

edebug ... 361
elp ... 361
email based diary ... 212
email spam ... 291, 292, 293
emphasis ... 91
ephemeral groups ... 358
Eudora ... 185
excessive crossposting ... 59
exiting Gnus ... 35
exiting groups ... 119
expirable mark ... 63
expiring mail ... 25, 34, 37, 114, 181
expiry, in Gnus agent ... 226
expiry-target ... 26
expiry-wait ... 26
extending the spam elisp package ... 312

F

face ... 288
faces ... 282
fancy mail splitting ... 175
fetching a group ... 286
fetching by Message-ID ... 110
file commands ... 45
file names ... 326
filtering approaches, spam ... 292
finding news ... 3
firewall ... 148
follow up ... 355
followup ... 135
fonts ... 282
foreign ... 356
foreign groups ... 21, 146
foreign servers ... 34
`format-time-string` ... 100
formatting variables ... 272
forwarded messages ... 204
functions ... 363
fuzzy article gathering ... 71
fuzzy matching ... 291

G

gateways ... 207
Gcc ... 138
gcc-self ... 25
generating sieve script ... 46
gereral customization ... 359
git commit messages ... 204
global score files ... 250
Gmail ... 162
gmane ... 22
Gmane, gnus-fetch-old-headers ... 72
Gmane, spam reporting ... 306
gnu.emacs.gnus ... 362
gnus ... 3
Gnus agent ... 216
Gnus agent expiry ... 226
Gnus agent regeneration ... 227
Gnus unplugged ... 216
Gnus utility functions ... 363
Gnus versions ... 328
gnus-activate-all-groups ... 44
gnus-activate-foreign-newsgroups ... 23
gnus-activate-level ... 20
gnus-active ... 363
gnus-adaptive-file-suffix ... 246
gnus-adaptive-pretty-print ... 246
gnus-adaptive-word-length-limit ... 246
gnus-adaptive-word-minimum ... 246
gnus-adaptive-word-no-group-words ... 246
gnus-adaptive-word-syntax-table ... 246
gnus-add-configuration ... 279
gnus-add-current-to-buffer-list ... 363
gnus-add-timestamp-to-message ... 325
gnus-add-to-list ... 24, 135
gnus-after-exiting-gnus-hook ... 35
gnus-after-getting-new-news-hook ... 44
gnus-agent ... 228
gnus-agent-add-group ... 224
gnus-agent-add-server ... 225
gnus-agent-auto-agentize-methods ... 231
gnus-agent-batch ... 232
gnus-agent-cache ... 229
gnus-agent-catchup ... 224
gnus-agent-consider-all-articles ... 229
gnus-agent-directory ... 229
gnus-agent-eagerly-store-articles ... 229
gnus-agent-enable-expiration ... 223
gnus-agent-expire ... 226
gnus-agent-expire-all ... 226
gnus-agent-expire-days ... 223, 226
gnus-agent-expire-group ... 226
gnus-agent-fetch-group ... 224
gnus-agent-fetch-groups ... 224
gnus-agent-fetch-session ... 224
gnus-agent-fetched-hook ... 229
gnus-agent-go-online ... 229
gnus-agent-handle-level ... 229
gnus-agent-high-score ... 223
gnus-agent-long-article ... 223
gnus-agent-low-score ... 223
gnus-agent-mark-article ... 224
gnus-agent-mark-unread-after-downloaded ... 229
gnus-agent-max-fetch-size ... 230
gnus-agent-plugged-hook ... 229
gnus-agent-prompt-send-queue ... 230
gnus-agent-queue-mail ... 230
gnus-agent-regenerate ... 227
gnus-agent-regenerate-group ... 227
gnus-agent-remove-group ... 224
gnus-agent-remove-server ... 225
gnus-agent-short-article ... 223
gnus-agent-summary-fetch-group ... 224
gnus-agent-summary-fetch-series ... 224
gnus-agent-synchronize-flags ... 224, 227, 229
gnus-agent-toggle-mark ... 224
gnus-agent-toggle-plugged ... 224
gnus-agent-unmark-article ... 224
gnus-agent-unplugged-hook ... 229
gnus-alter-articles-to-read-function ... 116
gnus-alter-header-function ... 74
gnus-always-force-window-configuration ... 279
gnus-always-read-dribble-file ... 9
gnus-ancient-mark ... 63
gnus-apply-kill-file ... 252
gnus-apply-kill-file-unless-scored ... 252
gnus-apply-kill-hook ... 252
gnus-article-add-buttons ... 97
gnus-article-add-buttons-to-head ... 97
gnus-article-address-banner-alist ... 93
gnus-article-babel ... 102
gnus-article-banner-alist ... 93
gnus-article-boring-faces ... 56
gnus-article-browse-html-article ... 103
gnus-article-button-face ... 99
gnus-article-capitalize-sentences ... 96
gnus-article-date-english ... 100
gnus-article-date-iso8601 ... 100
gnus-article-date-lapsed ... 101
gnus-article-date-local ... 100
gnus-article-date-original ... 101
gnus-article-date-user ... 100
gnus-article-date-ut ... 100
gnus-article-de-base64-unreadable ... 96
gnus-article-de-quoted-unreadable ... 96
gnus-article-decode-charset ... 104
gnus-article-decode-encoded-words ... 174
gnus-article-decode-hook ... 133
gnus-article-decode-HZ ... 96
gnus-article-decode-mime-words ... 104
gnus-article-describe-briefly ... 132
gnus-article-display-face ... 101, 288
gnus-article-display-x-face ... 101, 287
gnus-article-emojize-symbols ... 102
gnus-article-emphasize ... 91
gnus-article-emulate-mime ... 104
gnus-article-encrypt-body ... 115

gnus-article-encrypt-protocol 115
gnus-article-fill-cited-article 95
gnus-article-fill-long-lines 96
gnus-article-followup-with-original 133
gnus-article-hide 92
gnus-article-hide-boring-headers 92, 125
gnus-article-hide-citation 93
gnus-article-hide-citation-in-followups ... 94
gnus-article-hide-citation-maybe 94
gnus-article-hide-headers 92
gnus-article-hide-list-identifiers 92
gnus-article-hide-pem 92
gnus-article-hide-signature 92
gnus-article-highlight 90
gnus-article-highlight-citation 90
gnus-article-highlight-headers 90
gnus-article-highlight-signature 91
gnus-article-loose-mime 104
gnus-article-mail 132
gnus-article-maybe-highlight 90
gnus-article-menu-hook 283
gnus-article-mime-part-function 105
gnus-article-mode-hook 133
gnus-article-mode-line-format 133
gnus-article-mode-syntax-table 133
gnus-article-mouse-face 99
gnus-article-next-button 132
gnus-article-next-page 132
gnus-article-outlook-deuglify-article 95
gnus-article-outlook-rearrange-citation ... 95
gnus-article-outlook-repair-attribution ... 95
gnus-article-outlook-unwrap-lines 95
gnus-article-over-scroll 133
gnus-article-prepare-hook 133
gnus-article-press-button 127
gnus-article-prev-button 132
gnus-article-prev-page 132
gnus-article-refer-article 132
gnus-article-remove-cr 96
gnus-article-remove-images 102
gnus-article-remove-leading-whitespace 98
gnus-article-remove-trailing-blank-lines . 97
gnus-article-reply-with-original 132
gnus-article-save-directory 84
gnus-article-show-cursor 129
gnus-article-show-summary 132
gnus-article-skip-boring 56
gnus-article-smartquotes-map 95
gnus-article-sort-by-author 77
gnus-article-sort-by-date 77
gnus-article-sort-by-most-recent-date 77
gnus-article-sort-by-most-recent-number ... 77
gnus-article-sort-by-newsgroups 77
gnus-article-sort-by-number 77
gnus-article-sort-by-random 77
gnus-article-sort-by-schedule 215
gnus-article-sort-by-score 77
gnus-article-sort-by-subject 77
gnus-article-sort-functions 77
gnus-article-strip-all-blank-lines 97
gnus-article-strip-banner 93
gnus-article-strip-blank-lines 97
gnus-article-strip-headers-in-body 97
gnus-article-strip-leading-blank-lines 97
gnus-article-strip-leading-space 97
gnus-article-strip-multiple-blank-lines ... 97
gnus-article-strip-trailing-space 97
gnus-article-time-format 100
gnus-article-toggle-fonts 97
gnus-article-treat-ansi-sequences 96
gnus-article-treat-fold-headers 98
gnus-article-treat-fold-newsgroups 98
gnus-article-treat-non-ascii 95
gnus-article-treat-overstrike 95
gnus-article-treat-smartquotes 95
gnus-article-treat-types 130
gnus-article-treat-unfold-headers 98
gnus-article-unsplit-urls 96
gnus-article-verify-x-pgp-sig 97
gnus-article-wash-html 96
gnus-article-wide-reply-with-original 133
gnus-article-x-face-command 287
gnus-article-x-face-too-ugly 287
gnus-async-post-fetch-function 78
gnus-async-prefetch-article-p 78
gnus-async-unread-p 78
gnus-asynchronous 78
gnus-auto-center-summary 54
gnus-auto-expirable-marks 181
gnus-auto-expirable-newsgroups 182
gnus-auto-extend-newsgroup 55
gnus-auto-goto-ignores 230
gnus-auto-select-first 17
gnus-auto-select-next 53
gnus-auto-select-same 53
gnus-auto-select-subject 17
gnus-auto-subscribed-categories 7
gnus-auto-subscribed-groups 7
gnus-backup-startup-file 9
gnus-balloon-face-0 275
gnus-batch-score 236
gnus-before-resume-hook 11
gnus-before-startup-hook 11
gnus-binary-mode 112
gnus-binary-mode-hook 112
gnus-binary-show-article 112
gnus-blocked-images 129
gnus-body-boundary-delimiter 131
gnus-boring-article-headers 125
gnus-break-pages 134
gnus-browse-describe-briefly 34
gnus-browse-describe-group 34
gnus-browse-exit 34
gnus-browse-menu-hook 284
gnus-browse-mode 34
gnus-browse-read-group 34

gnus-browse-select-group 34
gnus-browse-subscribe-newsgroup-method 34
gnus-browse-toggle-subscription 34
gnus-buffer-configuration 276
gnus-bug 361
gnus-build-sparse-threads 73
gnus-button-alist 98
gnus-button-browse-level 100
gnus-button-emacs-level 100
gnus-button-man-handler 98, 99
gnus-button-man-level 100
gnus-button-message-level 100
gnus-button-mid-or-mail-heuristic 99
gnus-button-mid-or-mail-heuristic-alist ... 99
gnus-button-mid-or-mail-regexp 99
gnus-button-prefer-mid-or-mail 99
gnus-button-url-regexp 99
gnus-buttonized-mime-types 105
gnus-cache-active-file 79
gnus-cache-directory 79
gnus-cache-enter-article 80
gnus-cache-enter-articles 79
gnus-cache-generate-active 79
gnus-cache-generate-nov-databases 79
gnus-cache-move-cache 80
gnus-cache-remove-article 80
gnus-cache-remove-articles 79
gnus-cacheable-groups 79
gnus-cached-mark 63
gnus-canceled-mark 63
gnus-catchup-mark 63
gnus-category-add 223
gnus-category-copy 223
gnus-category-customize-category 222
gnus-category-edit-groups 223
gnus-category-edit-predicate 223
gnus-category-edit-score 223
gnus-category-exit 222
gnus-category-kill 223
gnus-category-line-format 223
gnus-category-list 223
gnus-category-mode-hook 223
gnus-category-mode-line-format 223
gnus-check-backend-function 364
gnus-check-bogus-newsgroups 11
gnus-check-new-newsgroups 5
gnus-child 4
gnus-cite-attribution-face 91
gnus-cite-attribution-prefix 91
gnus-cite-attribution-suffix 91
gnus-cite-face-list 91
gnus-cite-hide-absolute 94
gnus-cite-hide-percentage 94
gnus-cite-ignore-quoted-from 91
gnus-cite-max-prefix 90
gnus-cite-minimum-match-count 91
gnus-cite-parse-max-size 90
gnus-cited-closed-text-button-line-format . 93
gnus-cited-lines-visible 93
gnus-cited-opened-text-button-line-format . 93
gnus-cloud 323
gnus-cloud-download-all-data 323
gnus-cloud-interactive 324
gnus-cloud-method 324
gnus-cloud-storage-method 324
gnus-cloud-synced-files 324
gnus-cloud-upload-all-data 323
gnus-configure-frame 278
gnus-confirm-mail-reply-to-news 135
gnus-confirm-treat-mail-like-news 135
gnus-continuum-version 363
gnus-convert-image-to-face-command 288
gnus-convert-image-to-x-face-command 288
gnus-convert-pbm-to-x-face-command 288
gnus-convert-png-to-face 288
gnus-crosspost-complaint 59
gnus-current-home-score-file 247
gnus-current-prefix-symbol 382
gnus-current-prefix-symbols 382
gnus-dead-summary-mode 120
gnus-decay-score 255
gnus-decay-score-function 255
gnus-decay-scores 255
gnus-declare-backend 376
gnus-default-adaptive-score-alist 245
gnus-default-adaptive-word-score-alist ... 246
gnus-default-article-saver 82
gnus-default-directory 325
gnus-default-ignored-adaptive-words 246
gnus-del-mark 63
gnus-delay-article 61
gnus-delay-default-delay 61
gnus-delay-default-hour 61
gnus-delay-group 61
gnus-delay-header 61
gnus-delay-initialize 62
gnus-delay-send-queue 62
gnus-demon-add-disconnection 284
gnus-demon-add-handler 284
gnus-demon-add-rescan 284
gnus-demon-add-scan-timestamps 284
gnus-demon-add-scanmail 284
gnus-demon-cancel 284
gnus-demon-handlers 284
gnus-demon-init 284
gnus-demon-timestep 284
gnus-diary 214
gnus-diary-check-message 215
gnus-diary-delay-format-function 215
gnus-diary-summary-line-format 215
gnus-diary-time-format 215
gnus-directory 325
gnus-dired-attach 324
gnus-dired-attach-at-end 324
gnus-dired-find-file-mailcap 324
gnus-dired-print 325

gnus-display-mime 126
gnus-display-mime-function 126
gnus-dormant-mark 62
gnus-downloadable-mark 64
gnus-downloaded-mark 64
gnus-draft-edit-message 143
gnus-draft-send-all-messages 143
gnus-draft-send-message 143
gnus-draft-toggle-sending 143
gnus-dribble-directory 9
gnus-duplicate-file 122
gnus-duplicate-list-length 122
gnus-duplicate-mark 63
gnus-emphasis-alist 91
gnus-emphasis-bold 92
gnus-emphasis-bold-italic 92
gnus-emphasis-italic 92
gnus-emphasis-underline 92
gnus-emphasis-underline-bold 92
gnus-emphasis-underline-bold-italic 92
gnus-emphasis-underline-italic 92
gnus-empty-thread-mark 64
gnus-enter-category-buffer 224
gnus-ephemeral-group-p 364
gnus-exit-gnus-hook 35
gnus-exit-group-hook 120
gnus-expert-user 271
gnus-expirable-mark 63
gnus-extra-header 51
gnus-extra-headers 50
gnus-extract-address-components 47
gnus-face-0 275
gnus-face-from-file 288
gnus-face-properties-alist 287
gnus-fetch-group 286
gnus-fetch-old-ephemeral-headers 73
gnus-fetch-old-headers 72
gnus-fetch-partial-articles 108
gnus-file-save-name 83
gnus-find-method-for-group 363
gnus-find-subscribed-addresses 24
gnus-folder-save-name 83
gnus-Folder-save-name 83
gnus-forwarded-mark 63
gnus-gather-threads-by-references 72
gnus-gather-threads-by-subject 72
gnus-gcc-externalize-attachments 140
gnus-gcc-mark-as-read 140
gnus-gcc-post-body-encode-hook 140
gnus-gcc-pre-body-encode-hook 140
gnus-gcc-self-resent-messages 140
gnus-generate-horizontal-tree 113
gnus-generate-tree-function 113
gnus-generate-vertical-tree 113
gnus-get-info 363
gnus-get-new-news-hook 44
gnus-global-groups 129
gnus-global-score-files 250
gnus-goto-colon 276
gnus-goto-next-group-when-activating 44
gnus-gravatar-properties 291
gnus-gravatar-size 290
gnus-gravatar-too-ugly 291
gnus-group-add-to-virtual 23
gnus-group-apropos 31
gnus-group-best-unread-group 16
gnus-group-browse-foreign-server 4, 34
gnus-group-catchup-current 18
gnus-group-catchup-current-all 18
gnus-group-catchup-group-hook 18
gnus-group-charset-alist 106
gnus-group-check-bogus-groups 34
gnus-group-clear-data 8, 19
gnus-group-clear-data-on-native-groups . 8, 19
gnus-group-compact-group 43
gnus-group-customize 22
gnus-group-default-list-level 20
gnus-group-delete-group 23
gnus-group-describe-all-groups 44
gnus-group-describe-briefly 44
gnus-group-describe-group 44
gnus-group-description-apropos 31
gnus-group-edit-global-kill 252
gnus-group-edit-group 22
gnus-group-edit-group-method 22
gnus-group-edit-group-parameters 22
gnus-group-edit-local-kill 252
gnus-group-enter-directory 22
gnus-group-enter-server-mode 43
gnus-group-exit 35
gnus-group-expire-all-groups 34
gnus-group-expire-articles 34
gnus-group-find-new-groups 34
gnus-group-find-parameter 364
gnus-group-first-unread-group 16
gnus-group-foreign-p 364
gnus-group-get-new-news 44
gnus-group-get-new-news-this-group 44
gnus-group-goto-unread 16
gnus-group-ham-exit-processor-BBDB 306
gnus-group-ham-exit-processor-bogofilter . 308
gnus-group-ham-exit-processor-spamoracle . 312
gnus-group-ham-exit-processor-stat 310
gnus-group-ham-exit-processor-whitelist . 305
gnus-group-highlight 14
gnus-group-highlight-line 15
gnus-group-highlight-words-alist 92
gnus-group-jump-to-group 16
gnus-group-kill-all-zombies 18
gnus-group-kill-group 18
gnus-group-kill-level 18
gnus-group-kill-region 18
gnus-group-line-format 12
gnus-group-list-active 31
gnus-group-list-all-groups 31
gnus-group-list-all-matching 31

gnus-group-list-cached 31
gnus-group-list-dormant 32
gnus-group-list-flush 32
gnus-group-list-groups 31
gnus-group-list-inactive-groups 20
gnus-group-list-killed 31
gnus-group-list-level 31
gnus-group-list-limit 32
gnus-group-list-matching 31
gnus-group-list-plus 32
gnus-group-list-ticked 32
gnus-group-list-zombies 31
gnus-group-mail 43
gnus-group-make-directory-group 22
gnus-group-make-doc-group 22
gnus-group-make-empty-virtual 23
gnus-group-make-group 21
gnus-group-make-help-group 22
gnus-group-make-rss-group 22
gnus-group-make-useful-group 22
gnus-group-make-web-group 22
gnus-group-mark-buffer 21
gnus-group-mark-group 21
gnus-group-mark-regexp 21
gnus-group-mark-region 21
gnus-group-menu-hook 283
gnus-group-mode-hook 44
gnus-group-mode-line-format 14
gnus-group-name-charset-group-alist 42
gnus-group-name-charset-method-alist 41
gnus-group-native-p 364
gnus-group-news 43
gnus-group-next-group 15, 34
gnus-group-next-unread-group 15
gnus-group-next-unread-group-same-level ... 15
gnus-group-no-more-groups-hook 119
gnus-group-post-news 43
gnus-group-posting-charset-alist 107
gnus-group-prefixed-name 363
gnus-group-prepare-hook 44
gnus-group-prepared-hook 44
gnus-group-prev-group 15, 34
gnus-group-prev-unread-group 15
gnus-group-prev-unread-group-same-level ... 15
gnus-group-quick-select-group 16
gnus-group-quit 35
gnus-group-read-ephemeral-group 22
gnus-group-read-group 16
gnus-group-read-init-file 45
gnus-group-read-only-p 363
gnus-group-real-name 363
gnus-group-rename-group 22
gnus-group-restart 44
gnus-group-save-newsrc 45
gnus-group-secondary-p 364
gnus-group-select-group 16
gnus-group-select-group-ephemerally 17
gnus-group-send-queue 224
gnus-group-set-current-level 19
gnus-group-set-parameter 364
gnus-group-sort-by-alphabet 32
gnus-group-sort-by-level 32
gnus-group-sort-by-method 32
gnus-group-sort-by-rank 32
gnus-group-sort-by-real-name 32
gnus-group-sort-by-score 32
gnus-group-sort-by-server 32
gnus-group-sort-by-unread 32
gnus-group-sort-function 32
gnus-group-sort-groups 32
gnus-group-sort-groups-by-alphabet 33
gnus-group-sort-groups-by-level 33
gnus-group-sort-groups-by-method 33
gnus-group-sort-groups-by-rank 33
gnus-group-sort-groups-by-real-name 33
gnus-group-sort-groups-by-score 33
gnus-group-sort-groups-by-unread 33
gnus-group-sort-selected-groups 33
gnus-group-sort-selected-groups-by-alphabet 33
gnus-group-sort-selected-groups-by-level . 33
gnus-group-sort-selected-groups-by-method . 33
gnus-group-sort-selected-groups-by-rank ... 33
gnus-group-sort-selected-groups-by-real-name 33
gnus-group-sort-selected-groups-by-score . 33
gnus-group-sort-selected-groups-by-unread . 33
gnus-group-spam-exit-processor-blacklist . 305
gnus-group-spam-exit-processor-bogofilter 308
gnus-group-spam-exit-processor-report-gmane 306
gnus-group-spam-exit-processor-spamoracle 312
gnus-group-spam-exit-processor-stat 310
gnus-group-split 178
gnus-group-split-default-catch-all-group . 179
gnus-group-split-fancy 180
gnus-group-split-setup 180
gnus-group-split-update 180
gnus-group-split-updated-hook 180
gnus-group-suspend 35
gnus-group-toggle-subscription 18
gnus-group-toggle-subscription-at-point ... 18
gnus-group-tool-bar 291
gnus-group-transpose-groups 18
gnus-group-uncollapsed-levels 13
gnus-group-universal-argument 21
gnus-group-unmark-all-groups 21
gnus-group-unmark-group 21
gnus-group-unread 363
gnus-group-update-hook 15
gnus-group-use-permanent-levels 20
gnus-group-visible-select-group 17
gnus-group-yank-group 18
gnus-ham-process-destinations 296, 300

gnus-header-button-alist 99
gnus-header-face-alist 90
gnus-hidden-properties 326
gnus-hierarchial-home-score-file 247
gnus-home-adapt-file 248
gnus-home-directory 325
gnus-home-score-file 247
gnus-html-frame-width 129
gnus-html-prefetch-images 78
gnus-html-show-images 102
gnus-ignored-adaptive-words 246
gnus-ignored-from-addresses 51
gnus-ignored-headers 125
gnus-ignored-mime-types 104
gnus-ignored-newsgroups 10
gnus-info-find-node 44, 118
gnus-info-group 381
gnus-info-level 381
gnus-info-marks 381
gnus-info-method 381
gnus-info-params 381
gnus-info-rank 381
gnus-info-read 381
gnus-info-score 381
gnus-info-set-group 381
gnus-info-set-level 381
gnus-info-set-marks 381
gnus-info-set-method 381
gnus-info-set-params 381
gnus-info-set-rank 381
gnus-info-set-read 381
gnus-info-set-score 381
gnus-inhibit-images 134
gnus-inhibit-mime-unbuttonizing 105
gnus-inhibit-slow-scoring 248
gnus-inhibit-startup-message 11
gnus-inhibit-user-auto-expire 184
gnus-init-file 9, 45
gnus-insert-pseudo-articles 90
gnus-insert-random-x-face-header 288
gnus-interactive 382
gnus-interactive-catchup 272
gnus-interactive-exit 272
gnus-invalid-group-regexp 326
gnus-jog-cache 79
gnus-keep-backlog 81
gnus-keep-same-level 20
gnus-kill-file-mark 63
gnus-kill-file-mode-hook 252
gnus-kill-file-name 252
gnus-kill-files-directory 236
gnus-kill-killed 236
gnus-kill-save-kill-file 252
gnus-kill-sticky-article-buffer 81
gnus-kill-sticky-article-buffers 81
gnus-kill-summary-on-exit 120
gnus-killed-mark 63
gnus-large-ephemeral-newsgroup 17
gnus-large-newsgroup 17
gnus-level-default-subscribed 20
gnus-level-default-unsubscribed 20
gnus-level-killed 19
gnus-level-subscribed 19
gnus-level-unsubscribed 19
gnus-level-zombie 19
gnus-list-groups-with-ticked-articles 32
gnus-list-identifiers 29, 92
gnus-low-score-mark 63
gnus-mail-save-name 83
gnus-mailing-list-archive 124
gnus-mailing-list-groups 137
gnus-mailing-list-help 124
gnus-mailing-list-insinuate 124
gnus-mailing-list-mode 24
gnus-mailing-list-owner 124
gnus-mailing-list-post 124
gnus-mailing-list-subscribe 124
gnus-mailing-list-unsubscribe 124
gnus-make-predicate 286
gnus-mark-article-hook 55, 182
gnus-mark-copied-or-moved-articles-as-expirable 184
gnus-mark-unpicked-articles-as-read 111
gnus-max-image-proportion 129
gnus-message-archive-group 138
gnus-message-archive-method 138
gnus-message-highlight-citation 137
gnus-message-replyencrypt 144
gnus-message-replysign 144
gnus-message-replysignencrypted 144
gnus-mime-action-on-part 128
gnus-mime-buttonize-attachments-in-header 103
gnus-mime-copy-part 127
gnus-mime-delete-part 127
gnus-mime-display-attachment-buttons-in-header 103
gnus-mime-display-multipart-alternative-as-mixed 105
gnus-mime-display-multipart-as-mixed 106
gnus-mime-display-multipart-related-as-mixed 105
gnus-mime-inline-part 127
gnus-mime-multipart-functions 105
gnus-mime-pipe-part 128
gnus-mime-print-part 127
gnus-mime-replace-part 127
gnus-mime-save-part 127
gnus-mime-save-part-and-strip 127
gnus-mime-view-all-parts 104
gnus-mime-view-part 127
gnus-mime-view-part-as-charset 127
gnus-mime-view-part-as-type 127
gnus-mime-view-part-externally 128
gnus-mime-view-part-internally 128
gnus-mode-non-string-length 282

gnus-mouse-face 275, 283
gnus-move-split-methods 115
gnus-narrow-to-body 364
gnus-new-mail-mark 14
gnus-news-group-p 364
gnus-newsgroup-ignored-charsets 106
gnus-newsgroup-maximum-articles 17
gnus-newsgroup-name 363
gnus-newsgroup-variables 117
gnus-nntpserver-file 3
gnus-no-groups-message 11
gnus-no-server 4
gnus-not-empty-thread-mark 64
gnus-nov-is-evil 359
gnus-novice-user 271
gnus-numeric-save-name 83, 84
gnus-Numeric-save-name 84
gnus-options-not-subscribe 7
gnus-options-subscribe 7
gnus-other-frame 3
gnus-outlook-deuglify-unwrap-max 95
gnus-outlook-deuglify-unwrap-min 95
gnus-page-delimiter 134
gnus-paging-select-next 54
gnus-parameters 29
gnus-parameters-case-fold-search 30
gnus-parse-headers-hook 74, 326
gnus-part-display-hook 132
gnus-permanently-visible-groups 32, 44
gnus-pick-article-or-thread 111
gnus-pick-display-summary 111
gnus-pick-mode 111
gnus-pick-mode-hook 111
gnus-pick-next-page 111
gnus-pick-start-reading 111
gnus-pick-unmark-article-or-thread 111
gnus-picon-databases 289, 290
gnus-picon-domain-directories 290
gnus-picon-file-types 290
gnus-picon-inhibit-top-level-domains 290
gnus-picon-news-directories 290
gnus-picon-properties 290
gnus-picon-style 290
gnus-picon-user-directories 290
gnus-Plain-save-name 84
gnus-plain-save-name 82, 84
gnus-post-method 135
gnus-posting-styles 141
gnus-prefetched-article-deletion-strategy 78
gnus-preserve-marks 114
gnus-process-mark 64
gnus-prompt-before-saving 82
gnus-ps-print-hook 107
gnus-random-x-face 288
gnus-read-active-file 10
gnus-read-all-available-headers 73
gnus-read-ephemeral-debian-bug-group 23
gnus-read-ephemeral-emacs-bug-group 23
gnus-read-mark 63
gnus-read-method 364
gnus-read-newsrc-file 8
gnus-refer-article-method 110, 320
gnus-refer-thread-limit 109
gnus-refer-thread-limit-to-thread 110
gnus-refer-thread-use-search 109
gnus-registry-cache-file 320
gnus-registry-default-sort-function 320
gnus-registry-extra-entries-precious 323
gnus-registry-get-id-key 323
gnus-registry-marks 322
gnus-registry-max-entries 320
gnus-registry-prune-factor 320
gnus-registry-register-all 320
gnus-registry-set-article-mark 322
gnus-registry-set-id-key 322
gnus-registry-split-strategy 322
gnus-registry-track-extra 321
gnus-registry-unfollowed-groups 320
gnus-replied-mark 63
gnus-rmail-save-name 82
gnus-safe-html-newsgroups 326
gnus-save-all-headers 81
gnus-save-duplicate-list 122
gnus-save-killed-list 8
gnus-save-newsrc-file 8
gnus-save-newsrc-hook 9
gnus-save-quick-newsrc-hook 9
gnus-save-score 237
gnus-save-standard-newsrc-hook 9
gnus-saved-headers 81
gnus-saved-mark 63
gnus-score-after-write-file-function 238
gnus-score-below-mark 237
gnus-score-change-score-file 234
gnus-score-customize 234
gnus-score-decay-constant 255
gnus-score-decay-scale 255
gnus-score-edit-all-score 236
gnus-score-edit-current-scores 234
gnus-score-edit-exit 244
gnus-score-edit-file 234
gnus-score-edit-insert-date 244
gnus-score-exact-adapt-limit 246
gnus-score-expiry-days 238
gnus-score-file-suffix 236
gnus-score-find-bnews 237
gnus-score-find-favourite-words 233
gnus-score-find-hierarchical 238
gnus-score-find-score-files-function 237
gnus-score-find-single 237
gnus-score-find-trace 233
gnus-score-flush-cache 234, 236
gnus-score-followup-article 248
gnus-score-followup-thread 248
gnus-score-interactive-default-score 237
gnus-score-menu-hook 284

gnus-score-mimic-keymap 236
gnus-score-mode-hook 244
gnus-score-over-mark 237
gnus-score-pretty-print 244
gnus-score-search-global-directories 251
gnus-score-set-expunge-below 234
gnus-score-set-mark-below 234
gnus-score-thread-simplify 238
gnus-score-uncacheable-files 237
gnus-search-contact-tables 261
gnus-search-date-keys 261
gnus-search-default-engines 257
gnus-search-ignored-newsgroups 259
gnus-search-use-parsed-queries 257, 260
gnus-secondary-select-methods 4
gnus-select-article-hook 55
gnus-select-group-hook 17
gnus-select-method 3
gnus-selected-tree-face 112
gnus-sender-save-name 84
gnus-server-add-server 147
gnus-server-close-all-servers 151
gnus-server-close-server 150
gnus-server-compact-server 148
gnus-server-copy-server 147, 151
gnus-server-deny-server 150
gnus-server-edit-server 147
gnus-server-equal 364
gnus-server-exit 147
gnus-server-kill-server 147
gnus-server-line-format 146
gnus-server-list-servers 147
gnus-server-menu-hook 283
gnus-server-mode-hook 146
gnus-server-mode-line-format 147
gnus-server-offline-server 151
gnus-server-open-all-servers 150
gnus-server-open-server 150
gnus-server-read-server 147
gnus-server-regenerate-server 147
gnus-server-remove-denials 151
gnus-server-scan-server 147
gnus-server-show-server 147
gnus-server-to-method 364
gnus-server-unopen-status 230
gnus-server-yank-server 147
gnus-set-active 363
gnus-setup-news-hook 11
gnus-shell-command-separator 326
gnus-show-all-headers 125
gnus-show-threads 73
gnus-sieve-crosspost 46
gnus-sieve-file 45, 46
gnus-sieve-generate 46
gnus-sieve-region-end 45
gnus-sieve-region-start 45
gnus-sieve-update 46
gnus-signature-face 91
gnus-signature-limit 102
gnus-signature-separator 91, 102
gnus-simplify-all-whitespace 71
gnus-simplify-ignored-prefixes 71
gnus-simplify-subject-functions 71
gnus-simplify-subject-fuzzy 71
gnus-simplify-subject-fuzzy-regexp 71
gnus-simplify-subject-re 71
gnus-simplify-whitespace 71
gnus-single-article-buffer 133
gnus-site-init-file 9
gnus-smiley-file-types 289
gnus-sort-gathered-threads-function 74
gnus-sorted-header-list 125
gnus-spam-mark 299
gnus-spam-newsgroup-contents 299
gnus-spam-process-destinations 296, 300
gnus-spam-process-newsgroups 299
gnus-sparse-mark 63
gnus-split-methods 84
gnus-start-date-timer 101
gnus-started-hook 11
gnus-startup-file 9
gnus-startup-hook 11
gnus-sticky-article 80
gnus-stop-date-timer 101
gnus-subscribe-alphabetically 6
gnus-subscribe-hierarchical-interactive 7
gnus-subscribe-hierarchically 6
gnus-subscribe-interactively 6
gnus-subscribe-killed 6
gnus-subscribe-newsgroup-method 6
gnus-subscribe-options-newsgroup-method 7
gnus-subscribe-randomly 6
gnus-subscribe-topics 6
gnus-subscribe-zombies 6
gnus-sum-closing-bracket 49
gnus-sum-opening-bracket 49
gnus-sum-opening-bracket-adopted 49
gnus-sum-thread-tree-false-root 48
gnus-sum-thread-tree-indent 49
gnus-sum-thread-tree-leaf-with-other 49
gnus-sum-thread-tree-root 48
gnus-sum-thread-tree-single-indent 49
gnus-sum-thread-tree-single-leaf 49
gnus-sum-thread-tree-vertical 49
gnus-summary-article-posted-p 115
gnus-summary-attach-article 60
gnus-summary-beginning-of-article 56
gnus-summary-best-unread-article 55
gnus-summary-browse-url 108
gnus-summary-bubble-group 20
gnus-summary-caesar-message 94
gnus-summary-cancel-article 60
gnus-summary-catchup 65
gnus-summary-catchup-all 65
gnus-summary-catchup-all-and-exit 120
gnus-summary-catchup-and-exit 120

gnus-summary-catchup-and-goto-next-group . 120
gnus-summary-catchup-and-goto-prev-group . 120
gnus-summary-catchup-from-here 65
gnus-summary-catchup-to-here 65
gnus-summary-check-current 54
gnus-summary-clear-above 65
gnus-summary-clear-mark-forward 64
gnus-summary-copy-article 114
gnus-summary-create-article 114
gnus-summary-crosspost-article 114
gnus-summary-current-score 233
gnus-summary-customize-parameters 119
gnus-summary-default-score 237
gnus-summary-delete-article 114
gnus-summary-describe-briefly 118
gnus-summary-describe-group 118
gnus-summary-display-arrow 116
gnus-summary-display-while-building 116
gnus-summary-down-thread 76
gnus-summary-dummy-line-format 70
gnus-summary-edit-article 115
gnus-summary-edit-article-done 115
gnus-summary-edit-global-kill 252
gnus-summary-edit-local-kill 252
gnus-summary-edit-parameters 119
gnus-summary-end-of-article 56
gnus-summary-enter-digest-group 118
gnus-summary-execute-command 118
gnus-summary-exit 119
gnus-summary-exit-hook 119
gnus-summary-exit-no-update 120
gnus-summary-expand-window 119
gnus-summary-expire-articles 114
gnus-summary-expire-articles-now 114
gnus-summary-expunge-below 237
gnus-summary-first-unread-article 55
gnus-summary-followup 59
gnus-summary-followup-to-mail 59
gnus-summary-followup-to-mail-with-original 59
gnus-summary-followup-with-original 59
gnus-summary-force-verify-and-decrypt 97
gnus-summary-gather-exclude-subject 71
gnus-summary-gather-subject-limit 70
gnus-summary-generate-hook 116
gnus-summary-goto-article 55
gnus-summary-goto-last-article 55
gnus-summary-goto-subject 53
gnus-summary-goto-unread 65, 271
gnus-summary-hide-all-threads 75
gnus-summary-hide-thread 75
gnus-summary-highlight 53
gnus-summary-idna-message 95
gnus-summary-ignore-duplicates 116
gnus-summary-import-article 114
gnus-summary-increase-score 234
gnus-summary-insert-cached-articles 118
gnus-summary-insert-dormant-articles 118
gnus-summary-insert-new-articles 69
gnus-summary-insert-old-articles 69
gnus-summary-insert-ticked-articles 118
gnus-summary-isearch-article 56
gnus-summary-kill-below 65
gnus-summary-kill-process-mark 67
gnus-summary-kill-same-subject 65
gnus-summary-kill-same-subject-and-select . 65
gnus-summary-kill-thread 75
gnus-summary-limit-exclude-childless-dormant 68
gnus-summary-limit-exclude-dormant 68
gnus-summary-limit-exclude-marks 68
gnus-summary-limit-include-cached 68
gnus-summary-limit-include-dormant 68
gnus-summary-limit-include-expunged 68
gnus-summary-limit-include-thread 68
gnus-summary-limit-mark-excluded-as-read . 69
gnus-summary-limit-to-address 67
gnus-summary-limit-to-age 68
gnus-summary-limit-to-articles 68
gnus-summary-limit-to-author 67
gnus-summary-limit-to-bodies 69
gnus-summary-limit-to-display-predicate . . . 68
gnus-summary-limit-to-extra 68
gnus-summary-limit-to-headers 69
gnus-summary-limit-to-marks 68
gnus-summary-limit-to-recipient 67
gnus-summary-limit-to-replied 68
gnus-summary-limit-to-score 68
gnus-summary-limit-to-singletons 67
gnus-summary-limit-to-subject 67
gnus-summary-limit-to-unread 68
gnus-summary-limit-to-unseen 68
gnus-summary-line-format 47, 51
gnus-summary-lower-score 234
gnus-summary-lower-thread 75
gnus-summary-mail-crosspost-complaint 59
gnus-summary-mail-forward 57
gnus-summary-mail-other-window 58
gnus-summary-make-false-root 70
gnus-summary-make-false-root-always 70
gnus-summary-mark-above 65
gnus-summary-mark-as-dormant 64
gnus-summary-mark-as-expirable 65
gnus-summary-mark-as-processable 66
gnus-summary-mark-as-read-backward 65
gnus-summary-mark-as-read-forward 64
gnus-summary-mark-as-spam 296
gnus-summary-mark-below 233
gnus-summary-mark-read-and-unread-as-read . 55
gnus-summary-mark-region-as-read 65
gnus-summary-mark-unread-as-read 55
gnus-summary-menu-hook 283
gnus-summary-mode-hook 116
gnus-summary-mode-line-format 52
gnus-summary-morse-message 95
gnus-summary-move-article 114

gnus-summary-muttprint 82
gnus-summary-muttprint-program 82
gnus-summary-news-other-window 58
gnus-summary-next-article 54
gnus-summary-next-group 120
gnus-summary-next-group-on-exit 16
gnus-summary-next-page 54, 56
gnus-summary-next-same-subject 55
gnus-summary-next-thread 76
gnus-summary-next-unread-article 54
gnus-summary-next-unread-subject 53
gnus-summary-next-unseen-article 55
gnus-summary-pick-line-format 111
gnus-summary-pipe-output 82
gnus-summary-pipe-output-default-command . 82
gnus-summary-pop-article 55
gnus-summary-pop-limit 68
gnus-summary-post-forward 59
gnus-summary-post-news 59
gnus-summary-prepare 118
gnus-summary-prepare-exit-hook 119
gnus-summary-prepare-hook 116
gnus-summary-prepared-hook 116
gnus-summary-prev-article 55
gnus-summary-prev-group 120
gnus-summary-prev-page 56
gnus-summary-prev-same-subject 55
gnus-summary-prev-thread 76
gnus-summary-prev-unread-article 54
gnus-summary-prev-unread-subject 53
gnus-summary-prev-unseen-article 55
gnus-summary-print-article 107
gnus-summary-raise-thread 75
gnus-summary-read-document 119
gnus-summary-refer-article 110
gnus-summary-refer-parent-article 109
gnus-summary-refer-references 109
gnus-summary-refer-thread 109
gnus-summary-remove-bookmark 65
gnus-summary-repair-multipart 103
gnus-summary-reparent-children 76
gnus-summary-reparent-thread 75
gnus-summary-repeat-search-article-backward 118
gnus-summary-repeat-search-article-forward 118
gnus-summary-reply 57
gnus-summary-reply-broken-reply-to 57
gnus-summary-reply-broken-reply-to-with-original 57
gnus-summary-reply-to-list-with-original . 57
gnus-summary-reply-with-original 57
gnus-summary-rescan-group 120
gnus-summary-rescore 233
gnus-summary-reselect-current-group 120
gnus-summary-resend-bounced-mail 58
gnus-summary-resend-message 58
gnus-summary-resend-message-edit 59
gnus-summary-respool-article 115
gnus-summary-respool-default-method 115
gnus-summary-respool-query 115
gnus-summary-respool-trace 115
gnus-summary-rethread-current 75
gnus-summary-same-subject 47
gnus-summary-save-article 81
gnus-summary-save-article-body-file 82
gnus-summary-save-article-file 82
gnus-summary-save-article-folder 82
gnus-summary-save-article-mail 81
gnus-summary-save-article-rmail 81
gnus-summary-save-article-vm 82
gnus-summary-save-body-in-file 83
gnus-summary-save-in-file 83
gnus-summary-save-in-folder 83
gnus-summary-save-in-mail 83
gnus-summary-save-in-pipe 83
gnus-summary-save-in-rmail 82
gnus-summary-save-in-vm 83
gnus-summary-save-newsrc 120
gnus-summary-save-parts 104
gnus-summary-save-process-mark 67
gnus-summary-scroll-down 56
gnus-summary-scroll-up 56
gnus-summary-search-article-backward 118
gnus-summary-search-article-forward 118
gnus-summary-select-article-buffer 56
gnus-summary-selected-face 52
gnus-summary-set-bookmark 65
gnus-summary-set-score 233
gnus-summary-show-all-threads 75
gnus-summary-show-article 56
gnus-summary-show-article-charset-alist ... 56
gnus-summary-show-complete-article 108
gnus-summary-show-thread 75
gnus-summary-sort-by-author 108
gnus-summary-sort-by-chars 108
gnus-summary-sort-by-date 108
gnus-summary-sort-by-extra 109
gnus-summary-sort-by-lines 108
gnus-summary-sort-by-marks 108
gnus-summary-sort-by-most-recent-date 108
gnus-summary-sort-by-most-recent-number . 108
gnus-summary-sort-by-newsgroups 108
gnus-summary-sort-by-number 108
gnus-summary-sort-by-original 109
gnus-summary-sort-by-random 109
gnus-summary-sort-by-recipient 108
gnus-summary-sort-by-schedule 215
gnus-summary-sort-by-score 108
gnus-summary-sort-by-subject 108
gnus-summary-stop-at-end-of-message 54
gnus-summary-stop-page-breaking 94
gnus-summary-supersede-article 60
gnus-summary-thread-gathering-function 72
gnus-summary-tick-above 65
gnus-summary-tick-article-forward 64

gnus-summary-toggle-display-buttonized... 104
gnus-summary-toggle-header... 95
gnus-summary-toggle-threads... 75
gnus-summary-toggle-truncation... 119
gnus-summary-tool-bar... 291
gnus-summary-top-thread... 76
gnus-summary-universal-argument... 118
gnus-summary-unmark-all-processable... 66
gnus-summary-unmark-as-processable... 66
gnus-summary-up-thread... 76
gnus-summary-update-hook... 52
gnus-summary-verbose-headers... 95
gnus-summary-very-wide-reply... 57
gnus-summary-very-wide-reply-with-original... 57
gnus-summary-wake-up-the-dead... 120
gnus-summary-wide-reply... 57
gnus-summary-wide-reply-with-original... 57
gnus-summary-write-article-file... 82
gnus-summary-write-body-to-file... 83
gnus-summary-write-to-file... 83
gnus-summary-yank-message... 60
gnus-summary-yank-process-mark... 67
gnus-summary-zcore-fuzz... 49
gnus-supercite-regexp... 91
gnus-supercite-secondary-regexp... 91
gnus-suppress-duplicates... 122
gnus-suspend-gnus-hook... 35
gnus-symbolic-argument... 272
gnus-thread-expunge-below... 74
gnus-thread-hide-killed... 74
gnus-thread-hide-subtree... 73
gnus-thread-ignore-subject... 74
gnus-thread-indent-level... 74
gnus-thread-operation-ignore-subject... 76
gnus-thread-score-function... 77
gnus-thread-sort-by-author... 76
gnus-thread-sort-by-date... 76
gnus-thread-sort-by-most-recent-date... 76
gnus-thread-sort-by-most-recent-number... 76
gnus-thread-sort-by-newsgroups... 76
gnus-thread-sort-by-number... 76
gnus-thread-sort-by-random... 76
gnus-thread-sort-by-recipient... 76
gnus-thread-sort-by-schedule... 215
gnus-thread-sort-by-score... 76
gnus-thread-sort-by-subject... 76
gnus-thread-sort-by-total-score... 76
gnus-thread-sort-functions... 76
gnus-ticked-mark... 62
gnus-topic-copy-group... 37
gnus-topic-copy-matching... 37
gnus-topic-create-topic... 36
gnus-topic-delete... 37
gnus-topic-display-empty-topics... 38
gnus-topic-display-predicate... 38
gnus-topic-edit-parameters... 38
gnus-topic-expire-articles... 37
gnus-topic-goto-next-topic... 38
gnus-topic-goto-previous-topic... 38
gnus-topic-hide-topic... 37
gnus-topic-indent... 36
gnus-topic-indent-level... 38
gnus-topic-jump-to-topic... 37
gnus-topic-kill-group... 36
gnus-topic-line-format... 38
gnus-topic-list-active... 38
gnus-topic-mark-topic... 37
gnus-topic-mode... 35
gnus-topic-mode-hook... 38
gnus-topic-move-group... 37
gnus-topic-move-matching... 37
gnus-topic-remove-group... 37
gnus-topic-rename... 37
gnus-topic-select-group... 36
gnus-topic-show-topic... 37
gnus-topic-sort-groups... 39
gnus-topic-sort-groups-by-alphabet... 39
gnus-topic-sort-groups-by-level... 39
gnus-topic-sort-groups-by-method... 39
gnus-topic-sort-groups-by-rank... 39
gnus-topic-sort-groups-by-score... 39
gnus-topic-sort-groups-by-server... 39
gnus-topic-sort-groups-by-unread... 39
gnus-topic-toggle-display-empty-topics... 37
gnus-topic-topology... 39
gnus-topic-unindent... 36
gnus-topic-unmark-topic... 37
gnus-topic-yank-group... 36
gnus-total-expirable-newsgroups... 183
gnus-treat-display-face... 131
gnus-treat-display-x-face... 131
gnus-treat-emphasize... 131
gnus-treat-fill-article... 131
gnus-treat-fill-long-lines... 131
gnus-treat-fold-headers... 132
gnus-treat-fold-newsgroups... 132
gnus-treat-from-gravatar... 101
gnus-treat-from-picon... 101
gnus-treat-hide-boring-headers... 131
gnus-treat-hide-citation... 131
gnus-treat-hide-citation-maybe... 131
gnus-treat-hide-headers... 131
gnus-treat-hide-signature... 131
gnus-treat-highlight-citation... 131
gnus-treat-highlight-headers... 132
gnus-treat-highlight-signature... 132
gnus-treat-leading-whitespace... 132
gnus-treat-mail-gravatar... 101
gnus-treat-mail-picon... 101
gnus-treat-newsgroups-picon... 101
gnus-treat-smiley... 101
gnus-treat-strip-banner... 131
gnus-treat-strip-list-identifiers... 131
gnus-treat-unfold-headers... 132
gnus-treat-x-pgp-sig... 132

gnus-tree-brackets ... 112
gnus-tree-line-format ... 112
gnus-tree-minimize-window ... 113
gnus-tree-mode-hook ... 112
gnus-tree-mode-line-format ... 112
gnus-tree-parent-child-edges ... 113
gnus-unbuttonized-mime-types ... 104
gnus-uncacheable-groups ... 79
gnus-undo ... 285
gnus-undo-mode ... 285
gnus-undownloaded-mark ... 64
gnus-unplugged ... 217
gnus-unread-mark ... 55, 62
gnus-unseen-mark ... 64
gnus-update-format ... 273
gnus-update-message-archive-method ... 138
gnus-update-score-entry-dates ... 238
gnus-updated-mode-lines ... 282
gnus-use-adaptive-scoring ... 244
gnus-use-article-prefetch ... 78
gnus-use-cache ... 79
gnus-use-cross-reference ... 120
gnus-use-dribble-file ... 9
gnus-use-full-window ... 276
gnus-use-idna ... 134
gnus-use-long-file-name ... 79, 85
gnus-use-scoring ... 236
gnus-use-trees ... 112
gnus-use-undo ... 285
gnus-useful-groups ... 22
gnus-user-agent ... 137
gnus-uu-be-dangerous ... 88
gnus-uu-correct-stripped-uucode ... 89
gnus-uu-decode-binhex ... 87
gnus-uu-decode-postscript ... 87
gnus-uu-decode-postscript-and-save ... 87
gnus-uu-decode-postscript-and-save-view ... 87
gnus-uu-decode-postscript-view ... 87
gnus-uu-decode-save ... 87
gnus-uu-decode-unshar ... 87
gnus-uu-decode-unshar-and-save ... 87
gnus-uu-decode-unshar-and-save-view ... 87
gnus-uu-decode-unshar-view ... 87
gnus-uu-decode-uu ... 86
gnus-uu-decode-uu-and-save ... 86
gnus-uu-decode-uu-and-save-view ... 86
gnus-uu-decode-uu-view ... 86
gnus-uu-decode-yenc ... 87
gnus-uu-digest-headers ... 135
gnus-uu-digest-mail-forward ... 59
gnus-uu-digest-post-forward ... 60
gnus-uu-do-not-unpack-archives ... 88
gnus-uu-grab-move ... 88
gnus-uu-grab-view ... 88
gnus-uu-grabbed-file-functions ... 88
gnus-uu-ignore-default-archive-rules ... 88
gnus-uu-ignore-default-view-rules ... 88
gnus-uu-ignore-files-by-name ... 88
gnus-uu-ignore-files-by-type ... 88
gnus-uu-invert-processable ... 66
gnus-uu-kill-carriage-return ... 88
gnus-uu-mark-all ... 67
gnus-uu-mark-buffer ... 67
gnus-uu-mark-by-regexp ... 66
gnus-uu-mark-over ... 67
gnus-uu-mark-region ... 66
gnus-uu-mark-series ... 67
gnus-uu-mark-sparse ... 67
gnus-uu-mark-thread ... 66, 75
gnus-uu-notify-files ... 86
gnus-uu-post-include-before-composing ... 89
gnus-uu-post-length ... 89
gnus-uu-post-news ... 60
gnus-uu-post-separate-description ... 89
gnus-uu-post-threaded ... 89
gnus-uu-pre-uudecode-hook ... 89
gnus-uu-save-in-digest ... 89
gnus-uu-tmp-dir ... 88
gnus-uu-unmark-articles-not-decoded ... 89
gnus-uu-unmark-by-regexp ... 66
gnus-uu-unmark-region ... 66
gnus-uu-unmark-thread ... 66, 75
gnus-uu-user-archive-rules ... 88
gnus-uu-user-view-rules ... 87
gnus-uu-user-view-rules-end ... 87
gnus-uu-view-and-save ... 88
gnus-valid-select-methods ... 376
gnus-verbose ... 325
gnus-verbose-backends ... 325
gnus-version ... 44
gnus-view-pseudo-asynchronously ... 90
gnus-view-pseudos ... 90
gnus-view-pseudos-separately ... 90
gnus-visible-headers ... 125
gnus-visual ... 282
gnus-visual-mark-article-hook ... 52
gnus-widen-article-window ... 133
gnus-window-min-height ... 278
gnus-window-min-width ... 278
gnus-x-face ... 287
gnus-x-face-directory ... 288
gnus-x-face-from-file ... 288
Google ... 22, 198
Graham, Paul ... 314
gravatars ... 101
group buffer ... 12
group buffer format ... 12
group description ... 44
group highlighting ... 14
group information ... 44
group level ... 19
group listing ... 31
group mail splitting ... 178
group mode line ... 14
group movement ... 15
group parameters ... 23, 38

group rank ... 20
group score ... 20
group score commands ... 236
group selection ... 16
group sieve commands ... 45
group timestamps ... 45
groups ... 1

H

`ham-marks` ... 300
head ... 356
header ... 356
headers ... 356
help group ... 22, 204
hiding headers ... 125
highlighting ... 14, 90, 282
highlights ... 325
history ... 55, 328
HTML ... 128
http ... 198
hung connections ... 324

I

IDNA ... 134
ifile, spam filtering ... 310
ignored groups ... 10
ignored-charsets ... 27
imap ... 159
IMAP labels ... 162
import old mail ... 180
importing PGP keys ... 123
incoming mail treatment ... 184
Incoming* ... 172
incorporating old mail ... 180
indirect connection functions ... 156
info ... 44
information on groups ... 44
interaction ... 271
interactive ... 382
internal variables ... 363
internationalized domain names ... 134
introduction to Gnus ... 1
invalid characters in file names ... 326
iso-8859-5 ... 107
ISO 8601 ... 100
ISO8601 ... 241
ispell ... 137
`ispell-message` ... 137

K

keys, reserved for users (Article) ... 132
keys, reserved for users (Group) ... 43
keys, reserved for users (Server) ... 147
keys, reserved for users (Summary) ... 47
kill files ... 251, 252
killed groups ... 357
koi8-r ... 107
koi8-u ... 107

L

Latin 1 ... 95
level ... 19
levels ... 357
limiting ... 67
links ... 165
list server brain damage ... 184
LIST overview.fmt ... 121
local variables ... 244
loose threads ... 70

M

M****s*** sm*rtq**t*s ... 95
Ma Gnus ... 328, 353
mail ... 57, 135, 162, 355
mail filtering (splitting) ... 164, 358
mail folders ... 194
mail group commands ... 114
mail list groups ... 24
mail message ... 356
mail NOV spool ... 188
mail server ... 166
mail sorting ... 358
mail source ... 166
mail splitting ... 164, 175, 178
mail spool ... 166
mail washing ... 184
`mail-extract-address-components` ... 47
Mail-Followup-To ... 24
mail-source ... 27
`mail-source-crash-box` ... 172
`mail-source-default-file-modes` ... 173
`mail-source-delete-incoming` ... 172
`mail-source-delete-old-incoming-confirm` ... 172
`mail-source-directory` ... 173
`mail-source-incoming-file-prefix` ... 173
`mail-source-movemail-program` ... 173
`mail-source-touch-pop` ... 136
`mail-sources` ... 173
mail-to-news gateways ... 207
maildir ... 190
mailing list ... 124
mailing lists ... 137
mairix ... 261
making digests ... 60
making groups ... 21

manual 44, 354
mark as unread 64
marking groups 21
marks 62
match-list 28
max-lisp-eval-depth 361
mbox 204
mbox folders 194
menus 282
merging groups 210
message 356
Message-ID 110
message-mail-p 142
message-news-p 142
message-send-mail-function 136
message-sent-hook 248
message-smtpmail-send-it 136
messages 135
mh-e mail spool 189
MH folders 83
MIME 126, 133, 330
MIME decoding 103
mm-decrypt-option 123
mm-encrypt-option 123
mm-file-name-collapse-whitespace 106
mm-file-name-delete-whitespace 106
mm-file-name-replace-whitespace 106
mm-file-name-rewrite-functions 106
mm-file-name-trim-whitespace 106
mm-sign-option 123
mm-text-html-renderer 128
mm-verify-option 123
MMDF mail box 204
mml-secure-message-encrypt-pgp 144
mml-secure-message-encrypt-pgpmime 144
mml-secure-message-encrypt-smime 144
mml-secure-message-sign-pgp 144
mml-secure-message-sign-smime 144
mml-unsecure-message 145
mml1991-use 123
mml2015-use 123
mode lines 282, 325
MODE READER 151
moderation 286
move mail 114
moving articles 115

N

naive Bayesian spam filtering 314
native 356
new features 332
new groups 5
new messages 44
news 355
news back ends 151
news spool 158
newsgroup 25
Newsgroups 50
nnatom 202
nnbabyl 187
nnbabyl-active-file 187, 188
nnbabyl-get-new-mail 187, 188
nnbabyl-mbox-file 187
nnchoke 366
nndiary 212
nndiary customization 214
nndiary mails 212
nndiary messages 212
nndiary operation modes 213
nndiary-mail-sources 214
nndiary-reminders 214
nndiary-split-methods 214
nndiary-week-starts-on-monday 214
nndir 22, 202
nndoc 22, 204
nndoc-article-type 205
nndoc-post-type 205
nndraft 143
nndraft-directory 143
nneething 22, 203
nneething-exclude-files 203
nneething-include-files 203
nneething-map-file 203
nneething-map-file-directory 203
nnfolder 194
nnfolder-active-file 194
nnfolder-delete-mail-hook 195
nnfolder-directory 194
nnfolder-generate-active-file 195
nnfolder-get-new-mail 187, 194
nnfolder-newsgroups-file 194
nnfolder-nov-directory 195
nnfolder-nov-file-suffix 195
nnfolder-nov-is-evil 195
nnfolder-save-buffer-hook 194
nngateway-address 208
nngateway-header-transformation 208
nngateway-mail2news-header-transformation 208
nngateway-simple-header-transformation 208
nnheader-file-name-translation-alist 326
nnheader-get-report 372
nnheader-head-chop-length 326
nnheader-max-head-length 325
nnheader-ms-strip-cr 185
nnheader-report 372
nnimap-address 159
nnimap-authenticator 160
nnimap-expunge 160
nnimap-fetch-partial-articles 161
nnimap-keepalive-intervals 161
nnimap-record-commands 161
nnimap-server-port 159
nnimap-split-download-body 162, 297
nnimap-stream 160

nnimap-streaming 160
nnimap-use-namespaces 161
nnimap-user 159
nnir 270
nnmail-cache-accepted-message-ids 174, 178
nnmail-cache-ignore-groups 174
nnmail-crosspost 165
nnmail-crosspost-link-function 165
nnmail-delete-file-function 174
nnmail-expiry-target 183
nnmail-expiry-wait 182
nnmail-expiry-wait-function 26, 182
nnmail-extra-headers 51
nnmail-fancy-expiry-target 183
nnmail-fancy-expiry-targets 183
nnmail-ignore-broken-references 185
nnmail-keep-last-article 183
nnmail-mail-splitting-charset 165
nnmail-mail-splitting-decodes 165
nnmail-message-id-cache-file 186
nnmail-message-id-cache-length 186
nnmail-pathname-coding-system 42
nnmail-post-get-new-mail-hook 174
nnmail-pre-get-new-mail-hook 174
nnmail-prepare-incoming-header-hook 185
nnmail-prepare-incoming-hook 185
nnmail-prepare-incoming-message-hook 186
nnmail-read-incoming-hook 174
nnmail-remove-leading-whitespace 185
nnmail-remove-list-identifiers 185
nnmail-remove-tabs 185
nnmail-resplit-incoming 165, 167
nnmail-scan-directory-mail-source-once ... 167
nnmail-split-abbrev-alist 177
nnmail-split-fancy 175
nnmail-split-fancy-match-partial-words ... 176
nnmail-split-fancy-syntax-table 177
nnmail-split-fancy-with-parent 177
nnmail-split-header-length-limit 165
nnmail-split-history 165
nnmail-split-hook 174
nnmail-split-lowercase-expanded 177
nnmail-split-methods 164
nnmail-treat-duplicates 186
nnmail-use-long-file-names 174
nnmaildir 190
nnmairix 261
nnmairix-create-search-group 265
nnmairix-create-search-group-from-message 266
nnmairix-create-server-and-default-group . 265
nnmairix-goto-original-article 266
nnmairix-group-change-query-this-group ... 265
nnmairix-group-delete-recreate-this-group 265
nnmairix-group-prefix 263
nnmairix-group-toggle-allowfast-this-group 265
nnmairix-group-toggle-propmarks-this-group 266
nnmairix-group-toggle-readmarks-this-group 265
nnmairix-group-toggle-threads-this-group . 265
nnmairix-mairix-search-options 264
nnmairix-mairix-update-options 265
nnmairix-only-use-registry 267
nnmairix-propagate-marks 266
nnmairix-propagate-marks-to-nnmairix-groups 268
nnmairix-propagate-marks-upon-close 267
nnmairix-purge-old-groups 270
nnmairix-remove-tick-mark-original-article 266
nnmairix-search 265
nnmairix-search-from-this-article 266
nnmairix-search-interactive 265
nnmairix-search-thread-this-article 266
nnmairix-update-database 265
nnmairix-update-groups 268
nnmairix-widget-search 265
nnmairix-widget-search-from-this-article . 266
nnmbox 187
nnmbox-active-file 187
nnmbox-get-new-mail 187
nnmbox-mbox-file 187
nnmh 189
nnmh-be-safe 189
nnmh-directory 189
nnmh-get-new-mail 187, 189
nnml 188
nnml-active-file 188
nnml-compressed-files-size-threshold 189
nnml-directory 188
nnml-generate-nov-databases 189
nnml-get-new-mail 187, 188
nnml-newsgroups-file 188
nnml-nov-file-name 189
nnml-nov-is-evil 188
nnml-prepare-save-mail-hook 189
nnml-use-compressed-files 189
nnnil 208
nnregistry 320
nnrss 199
nnrss-directory 200
nnrss-file-coding-system 200
nnrss-generate-download-script 200
nnrss-ignore-article-fields 200
nnrss-opml-export 200
nnrss-opml-import 200
nnrss-use-local 200
nnselect 209
nnspool 158
nnspool-active-file 158
nnspool-active-times-file 159
nnspool-history-file 159
nnspool-inews-program 158

nnspool-inews-switches 158
nnspool-lib-dir 158
nnspool-newsgroups-file 158
nnspool-nov-directory 158
nnspool-nov-is-evil 159
nnspool-sift-nov-with-sed 159
nnspool-spool-directory 158
nntp 151
nntp authentication 151
nntp-address 157
nntp-authinfo-file 151
nntp-authinfo-function 151
nntp-connection-timeout 152
nntp-end-of-line 158
nntp-maximum-request 152
nntp-netcat-command 158
nntp-netcat-switches 158
nntp-never-echoes-commands 154
nntp-nov-gap 153
nntp-nov-is-evil 153
nntp-open-connection-function 154
nntp-open-connection-functions-never-echo-commands 154
nntp-open-netcat-stream 155
nntp-open-network-stream 155
nntp-open-ssl-stream 155
nntp-open-telnet-stream 155
nntp-open-tls-stream 155
nntp-open-via-rlogin-and-netcat 156
nntp-open-via-rlogin-and-telnet 156
nntp-open-via-telnet-and-telnet 157
nntp-port-number 158
nntp-pre-command 157
nntp-prepare-post-hook 154
nntp-prepare-server-hook 153
nntp-record-commands 154
nntp-send-authinfo 151
nntp-send-mode-reader 151
nntp-server-action-alist 152
nntp-server-opened-hook 151
nntp-telnet-command 156
nntp-telnet-switches 156
nntp-via-address 157
nntp-via-envuser 157
nntp-via-rlogin-command 156
nntp-via-rlogin-command-switches 156
nntp-via-shell-prompt 157
nntp-via-telnet-command 157
nntp-via-telnet-switches 157
nntp-via-user-name 157
nntp-via-user-password 157
nntp-xover-commands 153
nntp-xref-number-is-evil 153
NNTP server 3
nntps 157
NNTPSERVER 3
nnvirtual 210
nnvirtual-always-rescan 211
nnweb 22, 198
nnweb-max-hits 199
nnweb-search 199
nnweb-type 199
nnweb-type-definition 199
No Gnus 328, 348
non-ascii group names 41
Non-ASCII 95
notifications, desktop 285
NOV 121, 153, 356

O

offline 216
OneList 93
Oort Gnus 328, 340
OPML 200
Outlook Express 95
overview.fmt 121

P

parameters 38
parent 358
parent articles 109
patches 361
Paul Graham 314
Pegasus 185
persistent articles 80
PGP key ring import 123
pick and read 111
picons 101
pop before smtp 136
pop3-leave-mail-on-server 169
pop3-movemail 169
pop3-uidl-file 169
POP 166
post 59, 135
post-method 27
posting styles 140
posting-style 27
PostScript 87, 107
pre-fetch 77
predicate specifiers 286
preferred charset 107
printing 107
process mark 63
process/prefix convention 271
procmail 166
profile 361
proxy 148
pseudo-articles 89
Pterodactyl Gnus 328

Q

Quassia Gnus 328

R

rank ... 20
rcvstore ... 83
reading init file ... 45
reading mail ... 162
reading news ... 151
Red Gnus ... 328
referring articles ... 109
regeneration ... 227
registry ... 318
regular expressions header matching, spam filtering ... 307
rejected articles ... 144
renaming groups ... 22
reply ... 135, 355
reporting bugs ... 361
restarting ... 44
reverse scoring ... 250
RFC 1036 ... 329
RFC 1522 decoding ... 174
RFC 1991 ... 330
RFC 2047 decoding ... 174
RFC 2396 ... 124
RFC 2440 ... 330
RFC 2822 ... 329
RFC 5322 ... 329
RFC 5536 ... 329
RFC 822 ... 329
rnews batch files ... 204
root ... 358
RSS ... 199
rule variables ... 87
running nndiary ... 213
Russian ... 107

S

saving .newsrc ... 45
saving articles ... 81
scanning new news ... 44
score cache ... 237
score commands ... 233
score decays ... 255
score file atoms ... 242
score file format ... 239
score file group parameter ... 26
score variables ... 236
scoring ... 233
scoring crossposts ... 249
scoring on other headers ... 248
scoring tips ... 249
search engines ... 257
search queries ... 260
search syntax ... 260
searching ... 257
searching the Usenet ... 198
secondary ... 356
sed ... 159
select groups ... 209
select method ... 357
select methods ... 146
selecting articles ... 54, 209
send delayed ... 61
sending mail ... 135
sent messages ... 137
September Gnus ... 328
series ... 86
server ... 357
server buffer format ... 146
server commands ... 147
server errors ... 4
server parameters ... 149
server variables ... 149
servers ... 1
setting marks ... 64
setting process marks ... 66
shared articles ... 87
shell archives ... 87
sieve ... 28
signatures ... 102
slash ... 92
slow ... 361
slow machine ... 360
Smartquotes ... 95
`smiley-data-directory` ... 289
`smiley-regexp-alist` ... 289
`smiley-style` ... 289
smileys ... 101, 289
snarfing keys ... 123
snews ... 157
solid groups ... 358
sorting groups ... 32
sox ... 87
spam ... 291, 293, 295, 297, 299, 301, 304, 306, 307, 308, 309, 310, 311
spam back ends ... 296, 304
spam configuration examples ... 301
spam elisp package, extending ... 312
spam filtering ... 295, 297, 299, 301, 304, 306, 307, 308, 309, 310, 311, 312
spam filtering approaches ... 292
spam filtering configuration examples ... 301
spam filtering incoming mail ... 297
spam filtering sequence of events ... 295
spam filtering variables ... 299
spam filtering, naive Bayesian ... 314
spam reporting ... 306
spam variables ... 299
`spam-autodetect-recheck-messages` ... 301
`spam-blackhole-good-server-regex` ... 307
`spam-blackhole-servers` ... 307
`spam-bogofilter-database-directory` ... 309
`spam-bogofilter-score` ... 308
`spam-ifile-all-categories` ... 310
`spam-ifile-database` ... 310
`spam-ifile-spam-category` ... 310

spam-initialize 295
spam-log-to-registry 301
spam-mark-ham-unread-before-move-from-spam-group 301
spam-mark-only-unseen-as-spam 301
spam-marks 300
spam-process-ham-in-nonham-groups 300
spam-process-ham-in-spam-groups 300
spam-regex-headers-ham 307
spam-regex-headers-spam 307
spam-report-gmane-use-article-number 306
spam-report-user-mail-address 307
spam-spamassassin-program 309
spam-spamoracle-binary 311
spam-spamoracle-database 311
spam-split-group 297
spam-stat 310, 315
spam-stat, spam filtering 310
spam-stat-buffer-change-to-non-spam 317
spam-stat-buffer-change-to-spam 317
spam-stat-buffer-is-no-spam 317
spam-stat-buffer-is-spam 317
spam-stat-file 316
spam-stat-load 317
spam-stat-process-directory-age 315
spam-stat-process-non-spam-directory 315
spam-stat-process-spam-directory 315
spam-stat-reduce-size 316
spam-stat-reset 315
spam-stat-save 316, 317
spam-stat-score-buffer 317
spam-stat-score-word 317
spam-stat-split-fancy 318
spam-stat-split-fancy-spam-group 316
spam-use-BBDB 306
spam-use-BBDB-exclusive 306
spam-use-blackholes 307
spam-use-blacklist 304
spam-use-bogofilter 308
spam-use-bogofilter-headers 308
spam-use-dig 307
spam-use-ifile 310
spam-use-regex-headers 307
spam-use-spamassassin 309
spam-use-spamassassin-headers 309
spam-use-spamoracle 311
spam-use-stat 295, 310
spam-use-whitelist 304
spam-use-whitelist-exclusive 305
SpamAssassin 294
spamassassin, spam filtering 309
spamming 121
SpamOracle 311
sparse articles 358
split 318
splitting mail 164
splitting, terminology 358
spool 357
starting up 3
startup files 8
sticky articles 80
stripping advertisements 93
styles 140
subscribed 24
subscription 5, 18
summary buffer 47
summary buffer format 47
summary exit 119
summary movement 53
summary sorting 108
superseding articles 60
symbolic prefixes 272
sync 323
synch 323
synchronization 323
system sleep 324

T

tabbed interface 281
tabs 281
temporary groups 358
terminology 355
the Gnus diary library 214
the nndiary back end 212
thread commands 75
thread root 358
threading 69, 358
timestamps 45
To 50
to-address 24
to-group 25
to-list 24
topic commands 35
topic parameters 38, 40
topic sorting 39
topic topology 39
topic variables 38
topics 35
topology 39
total-expire 25
track 318
transient-mark-mode 271
trees 112
troubleshooting 361

U

UCE ... 291, 292, 293
underline ... 92
undo ... 285
Unicode ... 95
unix mail box ... 187
Unix mbox ... 204
unplugged ... 216
unshar ... 87
unsolicited commercial email ... 291, 292, 293
updating sieve script ... 46
url ... 108
USEFOR ... 329
Usenet searches ... 198
User-Agent ... 137
using gpg ... 135, 144
using s/mime ... 135, 144
using smime ... 135, 144
UTF-8 group names ... 42
utility functions ... 363
uudecode ... 86
uuencode ... 104
uuencoded articles ... 86

V

velveeta ... 121
version ... 44
`version-control` ... 9
viewing attachments ... 103
viewing files ... 89
Vipul's Razor ... 294
virtual groups ... 210
virtual server ... 357
visible ... 25
visible group parameter ... 32
visual ... 282

W

washing ... 94, 358
web ... 108, 198
whitelists, spam filtering ... 304
window height ... 278
window layout ... 276
window width ... 278
www ... 198

X

x-face ... 101, 287
X-GM-LABELS ... 162
XOVER ... 153
Xref ... 121

Y

yEnc ... 104

Z

zombie groups ... 19, 357

15 Key Index

!

! (概略) 64

#

(グループ) 21
(概略) 66

$

$ (概略) 296

&

& (概略) 118

*

* (概略) 80

,

, (グループ) 16
, (概略) 55

.

. (グループ) 16
. (概略) 55
. (記事) 128
. (選択) 111

/

/ * (概略) 68
/ . (概略) 68
/ / (概略) 67
/ a (概略) 67
/ A (概略) 67
/ b (概略) 69
/ c (概略) 68
/ C (概略) 69
/ d (概略) 68
/ D (概略) 68
/ E (概略) 68
/ h (概略) 69
/ m (概略) 68
/ M (概略) 68
/ n (概略) 68
/ N (概略) 69
/ o (概略) 69
/ p (概略) 68
/ r (概略) 68
/ R (概略) 67
/ S (概略) 67
/ t (概略) 68
/ T (概略) 68
/ u (概略) 68
/ v (概略) 68
/ w (概略) 68
/ x (概略) 68

<

< (概略) 56

=

= (概略) 119

>

> (概略) 56

?

? (グループ) 44
? (概略) 64
? (閲覧) 34
? (記事) 132

[

[(概略) 55

]

] (概略) 55

^

^ (グループ) 43
^ (概略) 109

@

@ (エージェント概略) 224

|

| (概略) 82
| (記事) 128

A

a (グループ) 43
a (サーバー) 147
a (分類) 223
a (概略) 59
A ! (グループ) 32
A / (グループ) 32
A < (概略) 56
A > (概略) 56
A ? (グループ) 32
A a (グループ) 31
A A (グループ) 31
A c (グループ) 31
A d (グループ) 31
A D (概略) 118
A f (グループ) 32
A g (概略) 56
A k (グループ) 31
A l (グループ) 31
A m (グループ) 31
A M (グループ) 31
A M (概略) 124
A p (グループ) 32
A P (概略) 107
A R (概略) 109
A s (グループ) 31
A s (概略) 56
A S (概略) 80
A t (概略) 102
A T (トピック) 38
A T (概略) 109
A u (グループ) 31
A w (概略) 108
A z (グループ) 31

B

b (グループ) 34
b (概略) 103
B (グループ) 4, 34
B B (概略) 114
B c (概略) 114
B C-M-e (概略) 114
B DEL (概略) 114
B e (概略) 114
B i (概略) 114
B I (概略) 114
B m (概略) 114
B p (概略) 115
B q (概略) 115
B r (概略) 115
B t (概略) 115
B w (概略) 115

C

c (サーバー) 147, 151
c (グループ) 18
c (概略) 120
c (分類) 223
c (記事) 127
C (サーバー) 150
C (グループ) 18
C (記事) 127
C (概略) 60
C-c ^ (記事) 132
C-c C-c (スコア) 244
C-c C-c (投稿) 135
C-c C-c (記事) 115
C-c C-d (グループ) 44
C-c C-d (スコア) 244
C-c C-f (概略) 57
C-c C-i (グループ) 44
C-c C-m (記事) 132
C-c C-m c o (Message) 144
C-c C-m c p (Message) 144
C-c C-m c s (Message) 144
C-c C-m C-n (Message) 145
C-c C-m s o (Message) 144
C-c C-m s p (Message) 144
C-c C-m s s (Message) 144
C-c C-M-x (グループ) 34
C-c C-n a (概略) 124
C-c C-n h (概略) 124
C-c C-n o (概略) 124
C-c C-n p (概略) 124
C-c C-n s (概略) 124
C-c C-n u (概略) 124
C-c C-p (スコア) 244
C-c C-s (グループ) 32
C-c C-s C-a (概略) 108
C-c C-s C-c (概略) 108
C-c C-s C-d (概略) 108
C-c C-s C-i (概略) 108
C-c C-s C-l (概略) 108
C-c C-s C-m C-d (概略) 108
C-c C-s C-m C-m (概略) 108
C-c C-s C-n (概略) 108
C-c C-s C-o (概略) 109
C-c C-s C-r (概略) 109
C-c C-s C-s (概略) 108
C-c C-s C-t (概略) 108
C-c C-s C-u (概略) 108
C-c C-s C-x (概略 109
C-c C-x (トピック) 37
C-c C-x (グループ) 34
C-c M-g (グループ) 44
C-d (概略) 118
C-k (グループ) 18
C-k (トピック) 36
C-k (概略) 65
C-M-a (概略) 119
C-M-b (概略) 76

C-M-d (概略) ... 119
C-M-e (概略) ... 119
C-M-f (概略) ... 76
C-M-k (概略) ... 75
C-M-l (概略) ... 75
C-M-RET (グループ) ... 17
C-o (記事) ... 127
C-t (概略) ... 119
C-w (グループ) ... 18
C-w (概略) ... 65
C-x C-s (概略) ... 120
C-x C-t (グループ) ... 18
C-y (グループ) ... 18
C-y (トピック) ... 36

D

d (閲覧) ... 34
d (概略) ... 64
d (記事) ... 127
D (サーバー) ... 150
D (概略) ... 65
D e (下書き) ... 143
D g (グループ) ... 46
D s (下書き) ... 143
D S (下書き) ... 143
D t (下書き) ... 143
D u (グループ) ... 46
DEL (グループ) ... 15
DEL (記事) ... 132
DEL (概略) ... 56

E

e (サーバー) ... 147
e (記事) ... 128
e (分類) ... 222
e (概略) ... 115
E (概略) ... 65
E (記事) ... 128

F

f (概略) ... 59
F (グループ) ... 34
F (記事) ... 133
F (概略) ... 59

G

g (バイナリー) ... 112
g (サーバー) ... 147
g (グループ) ... 44
g (概略) ... 56
g (分類) ... 223
G b (概略) ... 55
G b a (グループ) ... 265
G b c (グループ) ... 265
G b d (グループ) ... 265
G b g (グループ) ... 265
G b i (グループ) ... 265
G b m (グループ) ... 265
G b o (グループ) ... 266
G b p (グループ) ... 266
G b q (グループ) ... 265
G b r (グループ) ... 265
G b s (グループ) ... 265
G b t (グループ) ... 265
G b u (グループ) ... 265
G c (グループ) ... 22
G C-n (概略) ... 55
G C-p (概略) ... 55
G d (グループ) ... 22
G D (グループ) ... 22
G DEL (グループ) ... 23
G e (グループ) ... 22
G E (グループ) ... 22
G f (グループ) ... 22
G f (概略) ... 55
G g (概略) ... 53
G G f (概略) ... 266
G G g (概略) ... 266
G G m (概略) ... 266
G G o (概略) ... 266
G G t (概略) ... 266
G G u (概略) ... 266
G h (グループ) ... 22
G j (概略) ... 55
G l (概略) ... 55
G m (グループ) ... 21
G M (グループ) ... 22
G M-n (概略) ... 53
G M-p (概略) ... 53
G n (概略) ... 54
G N (概略) ... 54
G o (概略) ... 55
G p (グループ) ... 22
G p (トピック) ... 38
G P (概略) ... 55
G P a (グループ) ... 33
G P l (グループ) ... 33
G P m (グループ) ... 33
G P n (グループ) ... 33
G P r (グループ) ... 33
G P s (グループ) ... 33
G P u (グループ) ... 33
G P v (グループ) ... 33

G r (グループ) 22
G R (グループ) 22, 200
G S a (グループ) 33
G S l (グループ) 33
G S m (グループ) 33
G S n (グループ) 33
G S r (グループ) 33
G S u (グループ) 33
G S v (グループ) 33
G u (グループ) 22
G u (概略) 55
G U (概略) 55
G v (グループ) 23
G V (グループ) 23
G w (グループ) 22
G z (グループ) 43

H

h (概略) 56
H d (グループ) 44
H d (概略) 118
H h (概略) 118
H i (概略) 118
H v (グループ) 44

I

i (グループ) 43
i (記事) 127

J

j (グループ) 16
j (概略) 55
J # (エージェント概略) 224
J a (エージェントグループ) 224
J a (エージェントサーバー) 225
J c (エージェントグループ) 224
J c (エージェント概略) 224
J j (エージェント) 224
J M-# (エージェント概略) 224
J r (エージェントグループ) 224
J r (エージェントサーバー) 225
J s (エージェントグループ) 224
J s (エージェント概略) 224
J S (エージェントグループ) 224
J S (エージェント概略) 224
J u (エージェントグループ) 224
J u (エージェント概略) 224
J Y (エージェントグループ) 224

K

k (サーバー) 147
k (記事) 81
k (概略) 65
k (分類) 223
K | (概略) 103
K b (概略) 103
K c (概略) 103
K d (概略) 103
K e (概略) 103
K E (概略) 115
K H (概略) 103
K i (概略) 103
K m (概略) 103
K o (概略) 103
K O (概略) 103
K r (概略) 103
K v (概略) 103

L

l (サーバー) 147
l (グループ) 31
l (分類) 223
l (閲覧) 34
l (概略) 55
L (グループ) 31
L (サーバー) 151

M

m (グループ) 43
m (概略) 58
M ? (概略) 64
M b (グループ) 21
M b (概略) 65
M B (概略) 65
M c (概略) 64
M C (概略) 65
M C-c (概略) 65
M d (概略) 64
M e (概略) 65
M h (概略) 65
M H (概略) 65
M k (概略) 65
M K (概略) 65
M m (グループ) 21
M P a (概略) 67
M P b (概略) 67
M P g (概略) 66
M P G (概略) 66
M P i (概略) 66
M P k (概略) 67
M P p (概略) 66
M P r (概略) 66
M P R (概略) 66
M P s (概略) 67
M P S (概略) 67

M P t (概略) 66
M P T (概略) 66
M P u (概略) 66
M P U (概略) 66
M P v (概略) 67
M P w (概略) 67
M P y (概略) 67
M r (グループ) 21
M s t 308
M s x (概略) 296
M S (概略) 68
M t (概略) 64
M u (グループ) 21
M U (グループ) 21
M V c (概略) 65
M V k (概略) 65
M V m (概略) 65
M V u (概略) 65
M w (グループ) 21
M-# (グループ) 21
M-# (概略) 66
M-& (概略) 118
M-* (概略) 80
M-^ (概略) 110
M-c (グループ) 19
M-c (サーバー) 151
M-d (グループ) 44
M-d (概略) 296
M-down (概略) 76
M-g (グループ) 44
M-g (概略) 120
M-i (概略) 272
M-k (グループ) 252
M-k (概略) 252
M-K (グループ) 252
M-K (概略) 252
M-n (グループ) 15
M-n (概略) 53
M-o (サーバー) 150
M-p (グループ) 15
M-p (概略) 53
M-R (概略) 118
M-RET (グループ) 16
M-RET (記事) 127
M-RET (概略) 56
M-s M-r (概略) 118
M-s M-s (概略) 118
M-S (概略) 118
M-SPC (グループ) 17
M-t (概略) 104
M-TAB (トピック) 36
M-TAB (記事) 132
M-u (概略) 64
M-up (概略) 76

N

n (グループ) 15
n (概略) 54
n (閲覧) 34
N (グループ) 15
N (概略) 54

O

o (概略) 81
o (記事) 127
O (サーバー) 150
O b (概略) 82
O f (概略) 82
O F (概略) 82
O h (概略) 82
O m (概略) 81
O o (概略) 81
O p (概略) 82
O P (概略) 82
O r (概略) 81
O v (概略) 82

P

p (グループ) 15
p (記事) 127
p (分類) 223
p (概略) 54
p (閲覧) 34
P (グループ) 15
P (概略) 55

Q

q (サーバー) 147
q (グループ) 35
q (記事) 81
q (閲覧) 34
q (分類) 222
q (概略) 119
Q (グループ) 35
Q (概略) 120

R

r (グループ) 45
r (記事) 127
r (概略) 57
R (グループ) 44
R (サーバー) 151
R (概略) 57
R (記事) 132
RET (トピック) 36
RET (グループ) 16
RET (選択) 111
RET (閲覧) 34

RET (概略) 56
RET (記事) 127

S

s (グループ) 45
s (サーバー) 147
s (記事) 132
s (概略) 56
s (分類) 223
S (サーバー) 147
S (概略) 60
S A (概略) 60
S B r (概略) 57
S B R (概略) 57
S C-k (グループ) 18
S D b (概略) 58
S D e (概略) 59
S D r (概略) 58
S f (概略) 59
S F (概略) 59
S i (概略) 58
S k (グループ) 18
S l (グループ) 19
S L (概略) 57
S m (概略) 58
S M-c (概略) 59
S n (概略) 59
S N (概略) 59
S o m (概略) 57
S o p (概略) 59
S O m (概略) 59
S O p (概略) 60
S p (概略) 59
S r (概略) 57
S R (概略) 57
S s (グループ) 18
S t 308
S t (グループ) 18
S u (概略) 60
S v (概略) 57
S V (概略) 57
S w (グループ) 18
S w (概略) 57
S W (記事) 133
S W (概略) 57
S x (概略) 296
S y (グループ) 18
S y (概略) 60
S z (グループ) 18
SPC (グループ) 16
SPC (サーバー) 147
SPC (閲覧) 34
SPC (記事) 132
SPC (選択) 111
SPC (概略) 54, 56

T

t (グループ) 35
t (概略) 95
t (記事) 127
T # (トピック) 37
T # (概略) 75
T ^ (概略) 75
T c (トピック) 37
T C (トピック) 37
T d (概略) 76
T D (トピック) 37
T DEL (トピック) 37
T h (トピック) 37
T h (概略) 75
T H (トピック) 37
T H (概略) 75
T i (概略) 75
T j (トピック) 37
T k (概略) 75
T l (概略) 75
T m (トピック) 37
T M (トピック) 37
T M-# (トピック) 37
T M-# (概略) 75
T M-^ (概略) 76
T M-n (トピック) 38
T M-p (トピック) 38
T n (トピック) 36
T n (概略) 76
T o (概略) 76
T p (概略) 76
T r (トピック) 37
T s (トピック) 37
T s (概略) 75
T S (概略) 75
T S a (トピック) 39
T S e (トピック) 39
T S l (トピック) 39
T S m (トピック) 39
T S r (トピック) 39
T S s (トピック) 39
T S u (トピック) 39
T S v (トピック) 39
T t (概略) 75
T T (概略) 75
T TAB (トピック) 36
T u (概略) 76
TAB (トピック) 36
TAB (記事) 132

U

u (グループ) 18
u (選択) 111
u (閲覧) 34
U (グループ) 18

V

v (サーバー) 147
v (グループ) 43
v (記事) 132
v (概略) 47
V (グループ) 44
V c (概略) 234
V C (概略) 234
V e (概略) 234
V f (概略) 234
V F (概略) 234
V m (概略) 234
V R (概略) 233
V s (概略) 233
V S (概略) 233
V t (概略) 233
V w (概略) 233
V x (概略) 234

W

w (概略) 108
W 6 (概略) 96
W a (概略) 97
W A (概略) 96
W b (概略) 97
W B (概略) 97
W c (概略) 96
W C (概略) 96
W d (概略) 95
W D d (概略) 101
W D D (概略) 102
W D e (概略) 102
W D f (概略) 101
W D F (概略) 97
W D g (概略) 101
W D h (概略) 101
W D m (概略) 101
W D n (概略) 101
W D s (概略) 101
W D W (概略) 102
W D x (概略) 101
W e (グループ) 236
W e (概略) 91
W E a (概略) 97
W E A (概略) 97
W E e (概略) 97
W E l (概略) 97
W E m (概略) 97
W E s (概略) 97
W E t (概略) 97
W E w (概略) 98
W f (グループ) 236
W G f (概略) 98
W G n (概略) 98
W G u (概略) 98
W h (概略) 96
W H a (概略) 90
W H c (概略) 90
W H h (概略) 90
W H s (概略) 91
W i (概略) 95
W l (概略) 94
W m (概略) 95
W M c (概略) 104
W M h (概略) 103
W M v (概略) 104
W M w (概略) 104
W o (概略) 95
W p (概略) 97
W q (概略) 96
W Q (概略) 96
W r (概略) 94
W s (概略) 97
W t (概略) 95
W T e (概略) 101
W T i (概略) 100
W T l (概略) 100
W T o (概略) 101
W T p (概略) 100
W T s (概略) 100
W T u (概略) 100
W u (概略) 96
W U (概略) 95
W v (概略) 95
W w (概略) 95
W W a (概略) 92
W W b (概略) 92
W W B (概略) 93
W W c (概略) 93
W W C (概略) 94
W W C-c (概略) 94
W W h (概略) 92
W W l (概略) 92
W W P (概略) 92
W W s (概略) 92
W Y a (概略) 95
W Y c (概略) 95
W Y f (概略) 95
W Y u (概略) 95
W Z (概略) 96

X

x (概略) 68
X b (概略) 87
X m (概略) 104
X o (概略) 87
X p (概略) 87
X P (概略) 87
X s (概略) 87
X S (概略) 87
X u (概略) 86
X U (概略) 86
X v p (概略) 87
X v P (概略) 87

X v s (概略) 87
X v S (概略) 87
X v u (概略) 86
X v U (概略) 86
X Y (概略) 87

Y

y (サーバー) 147
Y c (概略) 118
Y d (概略) 118
Y g (概略) 118
Y t (概略) 118

Z

z (グループ) 35
z (サーバー) 148
Z c (概略) 120
Z C (概略) 120
Z E (概略) 120
Z G (概略) 120
Z n (概略) 120
Z N (概略) 120
Z p (概略) 120
Z P (概略) 120
Z Q (概略) 119
Z R (概略) 120
Z s (概略) 120
Z Z (概略) 119