

debian

Debian リファレンス

[FAMILY Given]

製作著作 © 2013-2026 Osamu Aoki (青木　修)

本 Debian リファレンス (第 2.153 版) (2026-11-29 05:38:00 UTC) はシステムインストール後のユーザー案内書として、Debian システムの広範な概論を提供します。本書は非開発者を対象にシェルコマンド例を通してシステム管理の多くの局面を説明します。

Contents

1 GNU/Linux チュートリアル **1**
1.1 コンソールの基礎 . . . 1
1.1.1 シェルプロンプト . . . 1
1.1.2 GUI の下でのシェルプロンプト . . . 2
1.1.3 root アカウント . . . 2
1.1.4 root シェルプロンプト . . . 3
1.1.5 GUI のシステム管理ツール . . . 3
1.1.6 仮想コンソール . . . 3
1.1.7 コマンドプロンプトからの退出方法 . . . 4
1.1.8 システムをシャットダウンする方法 . . . 4
1.1.9 合理的なコンソールの復元 . . . 4
1.1.10 初心者向け追加パッケージの提案 . . . 4
1.1.11 追加のユーザーアカウント . . . 6
1.1.12 sudo の設定 . . . 6
1.1.13 お遊びの時間 . . . 7
1.2 Unix-like ファイルシステム . . . 7
1.2.1 Unix ファイルの基礎 . . . 7
1.2.2 ファイルシステムの内側 . . . 8
1.2.3 ファイルシステムのパーミッション . . . 9
1.2.4 新規作成ファイルのパーミッションのコントロール: umask . . . 11
1.2.5 ユーザーのグループ (group) のパーミッション . . . 12
1.2.6 タイムスタンプ . . . 13
1.2.7 リンク . . . 14
1.2.8 名前付きパイプ (FIFO) . . . 15
1.2.9 ソケット . . . 16
1.2.10 デバイスファイル . . . 16
1.2.11 特別なデバイスファイル . . . 17
1.2.12 procfs と sysfs . . . 18
1.2.13 tmpfs . . . 18
1.3 ミッドナイトコマンダー (MC) . . . 18

1.3.1 MC のカスタム化 . . . 19
1.3.2 MC の始動 . . . 19
1.3.3 MC のファイルマネージャー . . . 19
1.3.4 MC のコマンドライントリック . . . 20
1.3.5 MC の内部エディター . . . 20
1.3.6 MC の内部ビューワー . . . 20
1.3.7 MC の自動起動機能 . . . 21
1.3.8 MC の仮想ファイルシステム . . . 21
1.4 基本の Unix 的作業環境 . . . 21
1.4.1 login シェル . . . 21
1.4.2 Bash のカスタム化 . . . 21
1.4.3 特別のキーストローク . . . 23
1.4.4 マウス操作 . . . 23
1.4.5 ページャー . . . 24
1.4.6 テキストエディター . . . 24
1.4.7 デフォールトのテキストエディターの設定 . . . 25
1.4.8 Vim 利用法 . . . 25
1.4.9 シェル活動の記録 . . . 25
1.4.10 基本 Unix コマンド . . . 27
1.5 シェルプロンプト . . . 29
1.5.1 コマンド実行と環境変数 . . . 29
1.5.2 "$LANG" 変数 . . . 29
1.5.3 "$PATH" 変数 . . . 30
1.5.4 "$HOME" 変数 . . . 31
1.5.5 コマンドラインオプション . . . 31
1.5.6 シェルグロブ . . . 31
1.5.7 コマンドの戻り値 . . . 32
1.5.8 典型的なコマンドシーケンスとシェルリディレクション . . . 33
1.5.9 コマンドエイリアス . . . 34
1.6 Unix 的テキスト処理 . . . 34
1.6.1 Unix テキストツール . . . 35
1.6.2 正規表現 . . . 36
1.6.3 置換式 . . . 36
1.6.4 正規表現を使ったグローバル置換 . . . 37
1.6.5 テキストファイルからのデーター抽出 . . . 38
1.6.6 コマンドをパイプするためのスクリプト断片 . . . 39

2 Debian パッケージ管理 41
2.1 Debian パッケージ管理の前提条件 . . . 41
2.1.1 Debian パッケージ管理システム . . . 41
2.1.2 パッケージ設定 . . . 41
2.1.3 基本的な注意事項 . . . 43
2.1.4 永遠のアップグレード人生 . . . 44
2.1.5 Debian アーカイブの基本 . . . 44
2.1.6 Debian は 100% フリーソフトウェアーです . . . 48
2.1.7 パッケージ依存関係 . . . 50
2.1.8 パッケージ管理のイベントの流れ . . . 51
2.1.9 パッケージ管理のトラブルへの応急対処法 . . . 52
2.1.10 Debian パッケージの選択方法 . . . 52
2.1.11 矛盾した要求への対処法 . . . 53
2.2 基本的パッケージ管理操作 . . . 53
2.2.1 `apt` と `apt-get`/`apt-cache` と `aptitude` の比較 . . . 53
2.2.2 コマンドラインによる基本的なパッケージ管理操作 . . . 54
2.2.3 aptitude のインタラクティブな使用 . . . 56
2.2.4 aptitude のキーバインディング . . . 56
2.2.5 aptitude の下でのパッケージの表示 . . . 57
2.2.6 aptitude を使った探索方法 . . . 58
2.2.7 aptitude の regex 式 . . . 58
2.2.8 aptitude による依存関係の解決 . . . 60
2.2.9 パッケージ活動ログ . . . 60
2.3 aptitude 操作例 . . . 60
2.3.1 興味あるパッケージの探索 . . . 61
2.3.2 regex にマッチするパッケージ名のパッケージをリスト . . . 61
2.3.3 regex マッチをしての閲覧 . . . 61
2.3.4 パッケージの完全削除 . . . 61
2.3.5 自動 / 手動インストール状態の整理 . . . 61
2.3.6 システム全体のアップグレード . . . 62
2.4 高度なパッケージ管理操作 . . . 63
2.4.1 コマンドラインによる高度なパッケージ管理操作 . . . 63
2.4.2 インストール済みパッケージファイルの検証 . . . 65
2.4.3 パッケージ問題からの防御 . . . 65
2.4.4 パッケージメタデーターの検索 . . . 66
2.5 Debian パッケージ管理の内部 . . . 66
2.5.1 アーカイブのメタデーター . . . 66
2.5.2 トップレベルの”Release” ファイルと信憑性 . . . 67
2.5.3 アーカイブレベルの”Release” ファイル . . . 68

2.5.4 パッケージメタデーターの取得 . . . 68
2.5.5 APT に関するパッケージ状態 . . . 69
2.5.6 aptitude に関するパッケージ状態 . . . 69
2.5.7 取得したパッケージのローカルコピー . . . 69
2.5.8 Debian パッケージファイル名 . . . 69
2.5.9 dpkg コマンド . . . 70
2.5.10 update-alternative コマンド . . . 72
2.5.11 dpkg-statoverride コマンド . . . 72
2.5.12 dpkg-divert コマンド . . . 72
2.6 壊れたシステムからの復元 . . . 72
2.6.1 古いユーザーの設定との非互換性 . . . 73
2.6.2 パッケージデーターのキャッシングエラー . . . 73
2.6.3 dpkg コマンドを使っての救済 . . . 73
2.6.4 依存関係の欠落により失敗したインストール . . . 73
2.6.5 重複するファイルを持つ相異なるパッケージ . . . 74
2.6.6 壊れたパッケージスクリプトの修正 . . . 74
2.6.7 パッケージセレクションの復元 . . . 75
2.7 パッケージ管理のヒント . . . 75
2.7.1 誰がパッケージをアップロードしたのか? . . . 75
2.7.2 APT のよるダウンロードバンド幅の制限 . . . 75
2.7.3 パッケージの自動ダウンロードとアップグレード . . . 76
2.7.4 Updates と Backports . . . 76
2.7.5 外部のパッケージアーカイブ . . . 77
2.7.6 **apt-pinning** を使わない混合のアーカイブソースからのパッケージ . . . 77
2.7.7 **apt-pinning** で候補バージョンを調整 . . . 78
2.7.8 ” 推奨 (Recommends)” によりパッケージがインストールされるのを阻止 . . . 79
2.7.9 `unstable` からのパッケージと共に、`testing` を追いかける . . . 80
2.7.10 `experimental` からのパッケージと共に、`unstable` を追いかける . . . 81
2.7.11 緊急ダウングレード . . . 82
2.7.12 equivs パッケージ . . . 82
2.7.13 安定版システムへのパッケージ移植 . . . 83
2.7.14 APT のためのプロキシサーバー . . . 83
2.7.15 パッケージ管理の追加参考文書 . . . 84

3 システムの初期化 85
3.1 ブートストラッププロセスの概要 . . . 85
3.1.1 1 段目: UEFI . . . 86
3.1.2 2 段目: ブートローダー . . . 86
3.1.3 3 段目: ミニ Debian システム . . . 87
3.1.4 4 段目: 普通の Debian システム . . . 88
3.2 レスキューシステム . . . 90
3.2.1 USB 上の GRUB UEFI レスキューシステム . . . 90
3.2.2 USB 上の Linux live レスキューシステム . . . 91
3.2.3 GRUB からの Linux live レスキューシステム . . . 92
3.3 Systemd . . . 92
3.3.1 Systemd init . . . 92
3.3.2 Systemd login . . . 93
3.4 カーネルメッセージ . . . 94
3.5 システムメッセージ . . . 94
3.6 システム管理 . . . 94
3.7 他のシステムモニター . . . 96
3.8 システム設定 . . . 96
3.8.1 ホスト名 . . . 96
3.8.2 ファイルシステム . . . 97
3.8.3 ネットワークインターフェースの初期化 . . . 97
3.8.4 クラウドシステムの初期化 . . . 97
3.8.5 sshd サービスを調整するカスタム化例 . . . 97
3.9 udev システム . . . 98
3.10 カーネルモジュール初期化 . . . 98

4 認証とアクセスの制御 100
4.1 普通の Unix 認証 . . . 100
4.2 アカウントとパスワードの情報管理 . . . 102
4.3 良好なパスワード . . . 102
4.4 暗号化されたパスワード作成 . . . 103
4.5 PAM と NSS . . . 103
4.5.1 PAM と NSS によってアクセスされる設定ファイル . . . 104
4.5.2 集中システム管理 . . . 105
4.5.3 「どうして GNU の su は wheel グループをサポートしないのか」 . . . 105
4.5.4 パスワード規則強化 . . . 105
4.6 認証のセキュリティー . . . 106
4.6.1 インターネット上でセキュアーなパスワード . . . 106
4.6.2 セキュアーシェル . . . 107

4.6.3 インターネットのためのセキュリティー強化策 . . . 107
4.6.4 root パスワードのセキュリティー確保 . . . 107
4.7 他のアクセスコントロール . . . 108
4.7.1 アクセス制御リスト (ACLs) . . . 108
4.7.2 sudo . . . 109
4.7.3 PolicyKit . . . 109
4.7.4 サーバーのサービスへのアクセスの制限 . . . 109
4.7.5 Linux のセキュリティ機能 . . . 110

5 ネットワークの設定 111
5.1 基本的ネットワークインフラ . . . 111
5.1.1 ホスト名の解決 . . . 111
5.1.2 ネットワークインターフェース名 . . . 113
5.1.3 LAN のためのネットワークアドレス範囲 . . . 114
5.1.4 ネットワークデバイスサポート . . . 114
5.2 デスクトップのための現代的なネットワーク設定 . . . 114
5.2.1 GUI のネットワーク設定ツール . . . 115
5.3 GUI 無しの現代的なネットワーク設定 . . . 115
5.4 クラウドのための現代的なネットワーク設定 . . . 116
5.4.1 クラウドのための現代的なネットワーク設定 . . . 116
5.4.2 クラウドのための静的 IP を使う現代的なネットワーク設定 . . . 116
5.4.3 クラウドのための Network Manager を使う現代的なネットワーク設定 . . . 116
5.5 低水準ネットワーク設定 . . . 117
5.5.1 Iproute2 コマンド . . . 117
5.5.2 安全な低レベルネットワーク操作 . . . 117
5.6 ネットワークの最適化 . . . 118
5.6.1 最適 MTU の発見 . . . 118
5.6.2 WAN TCP の最適化 . . . 119
5.7 Netfilter インフラ . . . 119

6 ネットワークアプリケーション 122
6.1 ウェッブブラウザー . . . 122
6.1.1 User-Agent 文字列をスプーフィングする . . . 122
6.1.2 ブラウザー拡張 . . . 123
6.2 メールシステム . . . 123
6.2.1 E メールの基本 . . . 123
6.2.2 現代的なメールサービスの制約 . . . 124
6.2.3 歴史的なメールサービスへの期待 . . . 125
6.2.4 メール転送エージェント (MTA) . . . 125

6.2.4.1 exim4 設定 . . . 125
6.2.4.2 SASL を使う postfix の設定 . . . 127
6.2.4.3 メールアドレス設定 . . . 128
6.2.4.4 基本的な MTA の操作 . . . 129
6.3 リーモートアクセスサーバーとユーティリティー (SSH) . . . 129
6.3.1 SSH の基本 . . . 130
6.3.2 リモートホストでのユーザ名 . . . 130
6.3.3 リモートパスワード無しでの接続 . . . 131
6.3.4 外部 SSH クライアントへの対処法 . . . 131
6.3.5 ssh-agent の設定 . . . 131
6.3.6 リモートホストからメールを送信する . . . 132
6.3.7 SMTP/POP3 トンネルをするためのポートフォワーディング . . . 132
6.3.8 SSH 上のリモートシステムをシャットダウンする方法 . . . 132
6.3.9 SSH のトラブルシュート . . . 133
6.4 プリントサーバーとユーティリティー . . . 133
6.5 他のネットワークアプリケーションサーバー . . . 133
6.6 他のネットワークアプリケーションクライアント . . . 134
6.7 システムデーモンの診断 . . . 134

7 GUI システム **137**
7.1 GUI デスクトップ環境 . . . 137
7.2 GUI 通信プロトコル . . . 138
7.3 GUI インフラストラクチャー . . . 139
7.4 GUI アプリケーション . . . 140
7.5 ユーザーディレクトリー . . . 140
7.6 フォント . . . 140
7.6.1 基本的なフォント . . . 140
7.6.2 フォントのラスタ化 . . . 143
7.7 サンドボックス . . . 144
7.8 リモートデスクトップ . . . 144
7.9 X サーバ接続 . . . 144
7.9.1 X サーバローカル接続 . . . 144
7.9.2 X サーバリモート接続 . . . 146
7.9.3 X サーバ chroot 接続 . . . 147
7.10 クリップボード . . . 147

8 I18N と L10N 148
8.1 ロケール . . . 148
8.1.1 UTF-8 ロケールを使う根拠 . . . 148
8.1.2 ロケールの再設定 . . . 149
8.1.3 ファイル名の符号化方式 . . . 150
8.1.4 地域化されたメッセージと翻訳された文書 . . . 150
8.1.5 ロケールの効果 . . . 151
8.2 キーボード入力 . . . 151
8.2.1 Linux コンソールと X Window 用のキーボードインプット . . . 152
8.2.2 Wayland 向けのキーボード入力 . . . 152
8.2.3 IBus を使うインプットメソッドのサポート . . . 152
8.2.4 Fcitx を使うインプットメソッドのサポート . . . 152
8.2.5 日本語の例 . . . 155
8.3 ディスプレー出力 . . . 155
8.3.1 ターミナルの設定 . . . 156
8.3.2 東アジア不明瞭文字幅文字 . . . 156

9 システムに関するティップ 157
9.1 コンソールのティップ . . . 157
9.1.1 シェルの活動を綺麗に記録 . . . 157
9.1.2 screen プログラム . . . 158
9.1.3 ディレクトリー間移動 . . . 159
9.1.4 Readline のラッパー . . . 159
9.1.5 ソースコードツリーのスキャン . . . 159
9.2 Vim のカスタム化 . . . 160
9.2.1 内部機能を使った vim のカスタマイズ . . . 160
9.2.2 外部パッケージを使った vim のカスタマイズ . . . 162
9.3 データーの記録と表現 . . . 163
9.3.1 ログデーモン . . . 163
9.3.2 ログアナライザー . . . 163
9.3.3 テキストデーターのカスタム化表示 . . . 164
9.3.4 時間と日付のカスタム化表示 . . . 164
9.3.5 着色化されたシェル出力 . . . 165
9.3.6 着色化されたコマンド . . . 165
9.3.7 複雑な反復のためにエディターでの活動を記録 . . . 166
9.3.8 X アプリケーションの画像イメージの記録 . . . 166
9.3.9 設定ファイルの変更記録 . . . 166
9.4 プログラム活動の監視と制御と起動 . . . 166
9.4.1 プロセスの時間計測 . . . 168

9.4.2 スケジューリングの優先度 . . . 168
9.4.3 ps コマンド . . . 168
9.4.4 top コマンド . . . 168
9.4.5 プロセスによって開かれているファイルのリスト . . . 169
9.4.6 プログラム活動の追跡 . . . 169
9.4.7 ファイルやソケットを使っているプロセスの識別 . . . 169
9.4.8 一定間隔でコマンドを反復実行 . . . 169
9.4.9 ファイルに関してループしながらコマンドを反復実行 . . . 170
9.4.10 GUI からプログラムをスタート . . . 170
9.4.11 スタートするプログラムのカスタム化 . . . 171
9.4.12 プロセスの停止 . . . 172
9.4.13 タスク 1 回実行のスケジュール . . . 172
9.4.14 タスク定期実行のスケジュール . . . 172
9.4.15 イベントに合わせたタスクのスケジュール . . . 174
9.4.16 Alt-SysRq キー . . . 174
9.5 システム管理ティップ . . . 175
9.5.1 だれがシステムを利用している? . . . 175
9.5.2 全員への警告 . . . 175
9.5.3 ハードウエアーの識別 . . . 175
9.5.4 ハードウエアー設定 . . . 176
9.5.5 システムとハードウエアーの時間 . . . 177
9.5.6 音のインフラ . . . 177
9.5.7 スクリーンセーバーの無効化 . . . 178
9.5.8 ブザー音の無効化 . . . 178
9.5.9 メモリー使用状況 . . . 178
9.5.10 システムのセキュリティーと整合性のチェック . . . 179
9.6 データー保存のティップ . . . 180
9.6.1 ディスク空間の利用状況 . . . 180
9.6.2 ディスクパーティション設定 . . . 180
9.6.3 UUID を使ってパーティションをアクセス . . . 181
9.6.4 LVM2 . . . 182
9.6.5 ファイルシステム設定 . . . 182
9.6.6 ファイルシステムの生成と整合性チェック . . . 184
9.6.7 マウントオプションによるファイルシステムの最適化 . . . 184
9.6.8 スーパーブロックによるファイルシステムの最適化 . . . 184
9.6.9 ハードディスクの最適化 . . . 185
9.6.10 ソリッドステートドライブの最適化 . . . 185
9.6.11 SMART を用いたハードディスクの破壊の予測 . . . 185
9.6.12 $TMPDIR 経由で一時保存ディレクトリーを指定 . . . 186

9.6.13 LVM を使う使用可能なストレージ空間の拡張 . . . 186
9.6.14 他パーティションをマウントする使用可能なストレージ空間の拡張 . . . 186
9.6.15 他ディレクトリーをバインドマウントする使用可能なストレージ空間の拡張 . . . 186
9.6.16 他ディレクトリーをオーバーレーマウントすることで使用可能なストレージ空間を拡張 . . . 187
9.6.17 シムリンクを使う使用可能なストレージ空間の拡張 . . . 187
9.7 ディスクイメージ . . . 187
9.7.1 ディスクイメージの作成 . . . 187
9.7.2 ディスクに直接書込み . . . 188
9.7.3 ディスクイメージファイルをマウント . . . 188
9.7.4 ディスクイメージのクリーニング . . . 190
9.7.5 空のディスクイメージ作成 . . . 190
9.7.6 ISO9660 イメージファイル作成 . . . 191
9.7.7 CD/DVD-R/RW に直接書込み . . . 191
9.7.8 ISO9660 イメージファイルをマウント . . . 192
9.8 バイナリーデーター . . . 192
9.8.1 バイナリーデーターの閲覧と編集 . . . 192
9.8.2 ファイルシステムをマウントせずに操作 . . . 192
9.8.3 データーの冗長性 . . . 193
9.8.4 データーファイルの復元と事故の証拠解析 . . . 193
9.8.5 大きなファイルを小さなファイルに分割 . . . 193
9.8.6 ファイル内容の消去 . . . 194
9.8.7 ダミーファイル . . . 194
9.8.8 ハードディスクの全消去 . . . 195
9.8.9 ハードディスク未使用部分の全消去 . . . 195
9.8.10 削除されたがまだオープン中のファイルの復活法 . . . 196
9.8.11 全てのハードリンクを検索 . . . 196
9.8.12 見えないディスクスペースの消費 . . . 196
9.9 データー暗号化ティップ . . . 197
9.9.1 dm-crypt/LUKS を使ったリムーバブルディスクの暗号化 . . . 197
9.9.2 dm-crypt/LUKS で暗号化されたディスクのマウント . . . 198
9.10 カーネル . . . 198
9.10.1 カーネル変数 . . . 198
9.10.2 カーネルヘッダー . . . 199
9.10.3 カーネルと関連モジュールのコンパイル . . . 199
9.10.4 カーネルソースのコンパイル: Debian カーネルチーム推奨 . . . 200
9.10.5 ハードウエアードライバーとファームウエアー . . . 200
9.11 仮想化システム . . . 201
9.11.1 仮想化やエミュレーションツール . . . 201
9.11.2 仮想化の業務フロー . . . 202
9.11.3 仮想ディスクイメージファイルをマウント。 . . . 204
9.11.4 Chroot システム . . . 204
9.11.5 複数のデスクトップシステム . . . 205

10 データー管理 **206**
10.1 共有とコピーとアーカイブ . . . 206
10.1.1 アーカイブと圧縮ツール . . . 207
10.1.2 コピーと同期ツール . . . 208
10.1.3 アーカイブの慣用句 . . . 208
10.1.4 コピーの慣用句 . . . 209
10.1.5 ファイル選択の慣用句 . . . 210
10.1.6 アーカイブメディア . . . 211
10.1.7 リムーバブルストレージデバイス . . . 212
10.1.8 データー共有用のファイルシステム選択 . . . 213
10.1.9 ネットワーク経由でのデーター共有 . . . 214
10.2 バックアップと復元 . . . 215
10.2.1 バックアップと復元のポリシー . . . 215
10.2.2 バックアップユーティリティーのスイート . . . 216
10.2.3 バックアップのティップ . . . 216
10.2.3.1 GUI バックアップ . . . 218
10.2.3.2 マウントイベントがトリガーするバックアップ . . . 218
10.2.3.3 タイマーイベントがトリガーするバックアップ . . . 219
10.3 データーセキュリティーのインフラ . . . 220
10.3.1 Gnupg のためのキー管理 . . . 220
10.3.2 GnuPG をファイルに使用 . . . 222
10.3.3 Mutt で GnuPG を使用 . . . 222
10.3.4 Vim で GnuPG を使用 . . . 224
10.3.5 MD5 和 . . . 224
10.3.6 パスワードキーリング . . . 224
10.4 ソースコードマージツール . . . 224
10.4.1 ソースファイル間の相違の抽出 . . . 226
10.4.2 ソースファイルに更新をマージ . . . 226
10.4.3 インタラクティブなマージ . . . 226
10.5 Git . . . 226
10.5.1 Git クライアントの設定 . . . 226
10.5.2 基本 Git コマンド . . . 227
10.5.3 Git ティップ . . . 228
10.5.4 Git リファレンス . . . 230
10.5.5 他のバージョンコントロールシステム . . . 230

11 データー変換 **232**
11.1 テキストデーター変換ツール . . . 232
11.1.1 テキストファイルを iconv を使って変換 . . . 232
11.1.2 ファイルが UTF-8 であると iconv を使い確認 . . . 234
11.1.3 iconv を使ってファイル名変換 . . . 234
11.1.4 行末変換 . . . 234
11.1.5 タブ変換 . . . 235
11.1.6 自動変換付きエディター . . . 235
11.1.7 プレーンテキスト抽出 . . . 236
11.1.8 プレーンテキストデーターをハイライトとフォーマット . . . 236
11.2 XML データー . . . 238
11.2.1 XML に関する基本ヒント . . . 238
11.2.2 XML 処理 . . . 239
11.2.3 XML データー抽出 . . . 239
11.2.4 XML データーの静的解析 . . . 241
11.3 タイプセッティング . . . 241
11.3.1 roff タイプセッティング . . . 241
11.3.2 TeX/LaTeX . . . 242
11.3.3 マニュアルページを綺麗に印刷 . . . 242
11.3.4 マニュアルページの作成 . . . 243
11.4 印刷可能データー . . . 243
11.4.1 Ghostscript . . . 243
11.4.2 2 つの PS や PDF ファイルをマージ . . . 243
11.4.3 印刷可能データーユーティリティー . . . 244
11.4.4 CUPS を使って印刷 . . . 244
11.5 メールデーター変換 . . . 244
11.5.1 メールデーターの基本 . . . 246
11.6 画像データーツール . . . 246
11.6.1 画像データーツール (メタパッケージ) . . . 246
11.6.2 画像データーツール (GUI) . . . 247
11.6.3 画像データーツール (CLI) . . . 247
11.7 その他のデーター変換 . . . 247

12 プログラミング **250**
12.1 シェルスクリプト . . . 250
12.1.1 POSIX シェル互換性 . . . 251
12.1.2 シェル変数 . . . 251
12.1.3 シェル条件式 . . . 253
12.1.4 シェルループ . . . 253

12.1.5 シェル環境変数 . . . 254
12.1.6 シェルコマンドライン処理シーケンス . . . 254
12.1.7 シェルスクリプトのためのユーティリティープログラム . . . 255
12.2 インタープリター言語でのスクリプティング . . . 257
12.2.1 インタープリター言語コードのデバグ . . . 257
12.2.2 シェルスクリプトを使った GUI プログラム . . . 257
12.2.3 GUI フィルター用のカスタム動作集 . . . 258
12.2.4 究極の短い Perl スクリプト . . . 258
12.3 コンパイル言語でのコーディング . . . 259
12.3.1 C . . . 259
12.3.2 単純な C プログラム (gcc) . . . 260
12.3.3 Flex —改良版 Lex . . . 260
12.3.4 Bison —改良版 Yacc . . . 260
12.4 静的コード分析ツール . . . 262
12.5 デバグ . . . 264
12.5.1 基本的な gdb 実行 . . . 264
12.5.2 Debian パッケージのデバグ . . . 264
12.5.3 バックトレースの収集 . . . 265
12.5.4 高度な gdb コマンド . . . 266
12.5.5 ライブラリーへの依存の確認 . . . 266
12.5.6 動的呼び出し追跡ツール . . . 266
12.5.7 X エラーのデバグ . . . 266
12.5.8 メモリーリーク検出ツール . . . 267
12.5.9 バイナリーのディスアッセンブリー . . . 267
12.6 ビルドツール . . . 267
12.6.1 Make . . . 267
12.6.2 Autotools . . . 268
12.6.2.1 プログラムをコンパイルとインストール . . . 268
12.6.2.2 プログラムのアンインストール . . . 269
12.6.3 Meson . . . 269
12.7 ウェッブ . . . 269
12.8 ソースコード変換 . . . 270
12.9 Debian パッケージ作成 . . . 270

A 補遺 271

A.1 Debian 迷路 . . . 271
A.2 著作権の経緯 . . . 271
A.3 文書のフォーマット . . . 272

List of Tables

1.1 興味あるテキストモードのプログラムパッケージのリスト 5
1.2 有用な文書パッケージのリスト 5
1.3 重要ディレクトリーの使い方のリスト 8
1.4 "ls -l" の出力の最初の文字のリスト 10
1.5 chmod(1) コマンドで用いられるファイルパーミッションの数字モード 11
1.6 **umask** 値の例 11
1.7 ファイルアクセスのためにシステムが供給する特記すべきグループのリスト 13
1.8 特定コマンド実行のためにシステムが供給する特記すべきグループのリスト 13
1.9 タイムスタンプのタイプのリスト 13
1.10 スペシャルなデバイスファイルのリスト 17
1.11 MC のキーバインディング 19
1.12 enter キー入力への MC の反応 21
1.13 シェルプログラムのリスト 22
1.14 Bash のキーバインディングのリスト 23
1.15 Debian 上でのマウス操作と関連キー操作のリスト 24
1.16 基本の Vim キーストロークのリスト 26
1.17 基本の Unix コマンドのリスト 28
1.18 ロケールの値の 3 つの部分 29
1.19 推奨ロケールのリスト 30
1.20 "$HOME" の値のリスト 31
1.21 シェルグロブパターン 31
1.22 コマンドの終了コード 32
1.23 シェルコマンドの慣用句 33
1.24 事前定義されたファイルデスクリプタ 34
1.25 BRE と ERE のメタ文字 36
1.26 置換式 37
1.27 コマンドをパイプするためのスクリプト断片 40

2.1 Debian のパッケージ管理ツールのリスト 42
2.2 Debian アーカイブサイトのリスト 46

2.3 Debian アーカイブエリアのリスト . . . 47
2.4 スイーツとコード名の関係 . . . 47
2.5 特定パッケージの問題解決のためのキーとなるウェッブサイトのリスト . . . 52
2.6 apt(8) や aptitude(8) や apt-get(8) /apt-cache(8) を使うコマンドラインによる基本パッケージ管理操作 55
2.7 aptitude(8) に関する特記すべきコマンドオプション . . . 56
2.8 aptitude のキーバインディングのリスト . . . 57
2.9 aptitude の表示のリスト . . . 58
2.10 標準パッケージ画面の分類 . . . 58
2.11 aptitude の regex 式のリスト . . . 59
2.12 パッケージ活動のログファイル . . . 60
2.13 高度なパッケージ管理操作 . . . 64
2.14 Debian アーカイブのメタデーターの内容 . . . 66
2.15 Debian パッケージの名前の構造 . . . 70
2.16 Debian パッケージ名の各部分に使用可能な文字 . . . 70
2.17 dpkg が作成する特記すべきファイル . . . 71
2.18 **apt-pinning** テクニックに関する特記すべき Pin-Priority 値をリストします。 . . . 79
2.19 Debian アーカイブ専用のプロキシツールのリスト . . . 84

3.1 ブートローダーのリスト . . . 86
3.2 /boot/grub/grub.cfg の上記部分のメニューエントリーの意味 . . . 87
3.3 Debian システムのブートユーティリティーのリスト . . . 89
3.4 カーネルエラーレベルのリスト . . . 94
3.5 典型的な journalctl コマンド断片の例 . . . 94
3.6 典型的な systemctl コマンド断片の例 . . . 95
3.7 systemd の下での他のモニタリングコマンドのリスト . . . 96

4.1 3 つの pam_unix(8) に関する重要な設定ファイル . . . 100
4.2 "/etc/passwd" の 2 番目のエントリーの内容 . . . 101
4.3 アカウント情報を管理するコマンドのリスト . . . 102
4.4 パスワード生成ツールのリスト . . . 103
4.5 特記すべき PAM と NSS システムのリスト . . . 103
4.6 PAM NSS によりアクセスされる設定ファイルのリスト . . . 104
4.7 インセキュアーとセキュアーのサービスとポートのリスト . . . 106
4.8 追加セキュリティー策を提供するツールのリスト . . . 107

5.1 GUI のネットワーク設定ツール . . . 112
5.2 ネットワークアドレス範囲のリスト . . . 114
5.3 旧式の net-tools コマンドと新しい iproute2 コマンド等との翻訳表 . . . 117
5.4 低レベルネットワークコマンドのリスト . . . 117
5.5 ネットワーク最適化ツールのリスト . . . 118

5.6 最適 MTU 値の基本的なガイドライン . . . 119
5.7 ファイアーウォールツールのリスト . . . 120

6.1 ウェッブブラウザーのリスト . . . 123
6.2 メールユーザーエージェント (MUA) のリスト . . . 124
6.3 基本的なメール転送エージェント関連パッケージのリスト . . . 126
6.4 重要 postfix マニュアルページのリスト . . . 127
6.5 メールアドレス関連のファイルのリスト . . . 128
6.6 基本的 MTA 操作のリスト . . . 129
6.7 リモートアクセスサーバーとユーティリティーのリスト . . . 130
6.8 SSH 設定ファイルのリスト . . . 131
6.9 SSH クライアント起動例のリスト . . . 131
6.10 他のプラットフォーム上で使えるフリーな SSH クライアントのリスト . . . 132
6.11 プリントサーバーとユーティリティーのリスト . . . 133
6.12 他のネットワークアプリケーションサーバー . . . 134
6.13 他のネットワークアプリケーションクライアント . . . 135
6.14 よく使われる RFC のリスト . . . 136

7.1 デスクトップ環境のリスト . . . 137
7.2 データーインフラパッケージのリスト . . . 139
7.3 特筆すべき GUI アプリケーションのリスト . . . 141
7.4 特記すべき TrueType や OpenType フォントのリスト . . . 142
7.5 有用フォント環境と関連パッケージのリスト . . . 143
7.6 特記すべきサンドボックス環境や関連のパッケージのリスト . . . 145
7.7 特記すべきリモートアクセスサーバーのリスト . . . 145
7.8 X サーバーへの接続方法のリスト . . . 146
7.9 文字クリップボードの操作関連プログラムのリスト . . . 147

8.1 IBus とプラグインパッケージのリスト . . . 153
8.2 Fcitx5 とプラグインパッケージのリスト . . . 154

9.1 コンソールの活動をサポートするプログラムのリスト . . . 157
9.2 screen キーバインディングのリスト . . . 159
9.3 vim 初期化に関する情報 . . . 163
9.4 システムログアナライザーのリスト . . . 164
9.5 "ls -l" コマンドを時間スタイル値とともに用いた場合の時間と日付の例 . . . 165
9.6 画像の操作ツールのリスト . . . 166
9.7 設定の履歴を記録するパッケージのリスト . . . 167
9.8 プログラム活動の監視と制御のツールのリスト . . . 167
9.9 スケジューリングの優先度のためのナイス値のリスト . . . 168

9.10 ps コマンドのスタイルのリスト . . . 168
9.11 kill コマンドが良く使うシグナルのリスト . . . 173
9.12 特記すべき SAK コマンドキーのリスト . . . 174
9.13 ハードウエアー識別ツールのリスト . . . 176
9.14 ハードウエアー設定ツールのリスト . . . 176
9.15 サウンドパッケージのリスト . . . 178
9.16 スクリーンセーバーを無効にするコマンドのリスト . . . 178
9.17 報告されるメモリーサイズのリスト . . . 179
9.18 システムセキュリティーや整合性確認のためのツールリスト . . . 180
9.19 ディスクパーティション管理パッケージのリスト . . . 181
9.20 ファイルシステム管理用パッケージのリスト . . . 183
9.21 バイナリーデーターを閲覧や編集するパッケージのリスト . . . 193
9.22 ファイルシステムをマウントせずに操作するパッケージのリスト . . . 193
9.23 ファイルにデーターの冗長性を追加するツールのリスト . . . 193
9.24 データーファイルの復元と事故の証拠解析のリスト . . . 194
9.25 データー暗号化ユーティリティーのリスト . . . 197
9.26 Debian システム上でカーネルの再コンパイルためにインストールする重要パッケージのリスト . . . 199
9.27 仮想化ツールのリスト . . . 203

10.1 アーカイブと圧縮ツールのリスト . . . 207
10.2 コピーと同期ツールのリスト . . . 208
10.3 典型的な使用シナリオに合わせたリムーバブルストレージデバイスのファイルシステムの選択肢のリスト . . . 213
10.4 典型的使用シナリオの場合のネットワークサービスの選択のリスト . . . 214
10.5 バックアップスイートのユーティリティーのリスト . . . 217
10.6 データーセキュリティーインフラツールのリスト . . . 220
10.7 キー管理のための GNU プライバシガードコマンドのリスト . . . 221
10.8 トラストコードの意味のリスト . . . 221
10.9 ファイルに使用する GNU プライバシーガードコマンドのリスト . . . 223
10.10ソースコードマージツールのリスト . . . 225
10.11git 関連のパッケージとコマンドのリスト . . . 227
10.12主要 Git コマンド . . . 228
10.13Git ティップ . . . 229
10.14他のバージョンコントロールシステムツールのリスト . . . 231

11.1 テキストデーター変換ツールのリスト . . . 232
11.2 符号化方式値とその使い方リスト . . . 233
11.3 異なるプラットフォーム上での行末スタイルのリスト . . . 235
11.4 `bsdmainutils` と `coreutils` パッケージ中のタブ変換コマンドのリスト . . . 235
11.5 プレーンテキストデーター抽出ツールのリスト . . . 236

11.6 プレーンテキストデーターをハイライトするツールのリスト . . . 237
11.7 XML で事前定義されているエントリーのリスト . . . 238
11.8 XML ツールのリスト . . . 239
11.9 DSSL ツールのリスト . . . 240
11.10テキストデーター変換ツールのリスト . . . 240
11.11XML 整形印刷ツールのリスト . . . 241
11.12タイプ設定ツールのリスト . . . 241
11.13マンページ作成を補助するパッケージのリスト . . . 243
11.14Ghostscript PostScript インタープリタのリスト . . . 244
11.15プリントできるデーターのユーティリティーのリスト . . . 245
11.16メールデーター変換を補助するパッケージのリスト . . . 245
11.17画像データーツールのリスト (メタパッケージ) . . . 247
11.18画像データーツール (GUI) のリスト . . . 248
11.19画像データーツールのリスト (CLI) . . . 249
11.20その他のデーター変換ツールのリスト . . . 249

12.1 典型的 bashizms のリスト . . . 251
12.2 シェル変数のリスト . . . 252
12.3 シェル変数展開のリスト . . . 252
12.4 重要なシェル変数置換のリスト . . . 252
12.5 条件式中のファイル比較演算子 . . . 253
12.6 条件式中での文字列比較演算子のリスト . . . 254
12.7 シェルスクリプト用の小さなユーティリティープログラムを含むパッケージのリスト . . . 256
12.8 インタープリター関連のパッケージのリスト . . . 256
12.9 ダイアログプログラムのリスト . . . 258
12.10コンパイラ関連のパッケージのリスト . . . 259
12.11Yacc 互換の LALR パーサー生成ソフトのリスト . . . 261
12.12静的コード分析ツールのリスト . . . 263
12.13デバッグパッケージのリスト . . . 264
12.14高度な gdb コマンドのリスト . . . 266
12.15メモリーリーク検出ツールのリスト . . . 267
12.16ビルドツールパッケージのリスト . . . 267
12.17make の自動変数のリスト . . . 268
12.18make 変数の展開のリスト . . . 268
12.19ソースコード変換ツールのリスト . . . 270

Abstract

This book is free; you may redistribute it and/or modify it under the terms of the GNU General Public License of any version compliant to the Debian Free Software Guidelines (DFSG). (日本語による参考説明: 本書はフリーです；Debian フリーソフトウェアーガイドライン (DFSG) に適合するいかなるバージョンの GNU General Public License の条件の下ででも再配布や改変をすることを許可します。)

序章

この Debian リファレンス (第 2.153 版) (2026-11-29 05:38:00 UTC) はシステムインストール後のユーザー向け案内書として Debian のシステム管理に関する概論の提供を目指しています。

本書が対象とする読者は、GNU/Linux システムがどう機能するかを理解するのに、シェルスクリプトぐらいは学ぶ気はあるが、全ての C のソースまで読む気がない人です。

インスト ー ルの方法は、以下を参照下さい:

- 現行安定システム用 Debian GNU/Linux インストールガイド
- 現行テスティング (testing) システム用 Debian GNU/Linux インストールガイド

免責事項

一切保証は致しません。全ての商標はそれぞれの商標の所有者の財産です。

Debian システム自体は動く標的です。このため最新状況を反映した正確な記述は困難です。現行のテスト版 `testing` の Debian システムを用いて本書は記していますが、皆様が読まれる時点ではすでに記載内容が古くなっているでしょう。

本書はあくまで二次的参考文献として扱って下さい。本書は正式の案内書を置き換えません。著者及び本書への貢献者は本書中の誤謬や欠落や曖昧さが引き起こす結果に一切責任を負いません。

Debian とはなにか

Debian プロジェクトはフリーなオペレーティングシステムを創造しようという共通目的を持った個人の集団です。そのディストリビューションは以下の特徴があります。

- ソフトウェアの自由へのコミットメント: Debian 社会契約と Debian フリーソフトウェアーガイドライン (DFSG)
- インターネット上の分散型の無償ボランティア活動: https://www.debian.org
- 多数のプリコンパイルされた高品質のソフトウェアーパッケージ
- セキュリティー更新への平易なアクセス提供による、安定性とセキュリティーの重視
- テスト版 `testing` アーカイブによる、最新のソフトウェアーへの円滑なアップグレードの重視
- 多数のサポートされたハードウエアーアーキテクチャー

Debian の中のフリーソフトウェアー構成要素は、GNU や Linux や BSD や X や ISC や Apache や Ghostscript や Common Unix Printing System や Samba や GNOME や KDE や Mozilla や LibreOffice や Vim や TeX や LaTeX や DocBook や Perl や Python や Tcl や Java や Ruby や PHP や Berkeley DB や MariaDB や PostgreSQL や SQLite や Exim や Postfix や Mutt や FreeBSD や OpenBSD や Plan 9 やその他の多くの独立のフリーソフトウェアーのプロジェクトに由来します。Debian はこの多種多様なフリーソフトウェアーを 1 つのシステムにまとめ上げます。

本書について

編集指針

本書の作成にあたり以下の編集指針を守りました。

- 概論を提供し枝葉末節は省略します。(全体像)
- 簡潔を心がけました。(**KISS**)
- 車輪の再発明をしません。(既存の参考文献へのポインターの利用)
- 非 GUI ツールとコンソールを重視します。(シェル例示を使用)
- 客観的であるようにします。(ポプコン等の利用)

ティップ
私はシステムの階層的側面やシステムの低レベルを明らかにしようとしました。

前提条件



警告
本文書だけに頼らず自分で答えを見出す努力をしっかりすることを期待します。本文書は効率的なスタートポイントを提供するだけです。

一義的情報源から自分自身で解決策を探し出すべきです。

- 一般的情報は https://www.debian.org にある Debian サイト
- ”`/usr/share/doc/`*`package_name`*” ディレクトリー下にある文書
- Unix スタイルのマンページ: ”`dpkg -L` *`package_name`* `|grep '/man/man.*/'`”
- GNU スタイルの **info** ページ: ”`dpkg -L` *`package_name`* `|grep '/info/'`”
- バグレポート: https://bugs.debian.org/*package_name*
- 変化中の事や特定案件に関しては、https://wiki.debian.org/ にある Debian の Wiki
- Open Group の The UNIX System Home Page 中の Single UNIX Specification
- https://www.wikipedia.org/ にある Wikipedia のフリーの百科事典
- The Debian Administrator’s Handbook
- The Linux Documentation Project (TLDP) の HOWTO 集

注意
詳細な文書を読むには、”`-doc`” をサフィクスとする対応する文書パッケージをインストールする必要があるかもしれません。

文書様式

bash(1) シェルコマンドの例示をする以下のような簡略化した表現スタイルで本書は情報を提供します。

```
# command-in-root-account
$ command-in-user-account
```

これらのシェルプロンプトは使われるアカウントを区別します。これはちょうど環境変数として: "PS1='\$'" と"PS2=' '" を設定した場合に相当します。これらの環境変数値はあくまで本書の読みやすさのためで、実際のインストール済みシステではほとんど見かけません。

すべてのコマンド例は英語ロケール"LANG=en_US.UTF8" 下で実行されます。コマンド例中の *command-in-root-accou* や *command-in-user-account* 等のプレースホルダー文字列が翻訳されるとは期待しないで下さい。これは全ての翻訳された例が最新版であるようするための意識的な選択です。

注意
"PS1='\$'" と"PS2=' '" という環境変数値の意味は bash(1) を参照下さい。

システム管理者が行うべきアクションは命令文で書かれています: 例えば、「シェルに各コマンド文字列をタイプ後毎にエンターキーをタイプします。」(必ずしも「〜しましょう。」とはせず簡潔に訳しています。)

英語では、テーブル中の説明や類似のコラムには、パッケージ説明の慣習に従い、定冠詞抜も不定冠詞も抜きの名詞句が入ります。これらには、マンページのコマンドの短い説明の慣習に従った頭の"to" 抜きの不定詞句が代わりに名詞句として入ることもあります。変だなとお考えの方もあるとは存じますが、これは本文書をできるだけ簡潔にするための著者の恣意的な文体の選択です。以上の短な説明慣行に従い、これら名詞句は大文字で始まらず読点もありません。(対応部分は文切り型の体言止めの日本語表現に訳します。)

注意
コマンド名を含めて固有名詞はその位置によらず大文字・小文字の区別を保持します。

本文中に引用されるコマンドの断片はダブルクォーテションマーク間にタイプライターフォントで書き"`aptitude safe-upgrade`" のように表現されます。

本文中に設定ファイルから引用された文字データーはダブルクォーテションマーク間にタイプライターフォントで書き"`deb-src`" のように表現されます。

コマンドはその名前をタイプライターフォントで書き、場合によってはその後ろにマンページのセクション番号を括弧中に入れて書き bash(1) のように表現されます。読者は以下のようにタイプして情報を得るように心がけて下さい。

```
$ man 1 bash
```

マンページはその名前をタイプライターフォントで書き、その後ろにマンページのセクション番号を括弧中に入れて書き sources.list(5) のように表現されます。読者は以下のようにタイプして情報を得るように心がけて下さい。

```
$ man 5 sources.list
```

info ページはダブルクォーテションマーク間にタイプライターフォントというコマンドの断片形式で書き"`info make`" のように表現されます。読者は以下のようにタイプして情報を得るように心がけて下さい。

```
$ info make
```

ファイル名はダブルクォーテションマーク間にタイプライターフォントで書き"`/etc/passwd`" のように表現されます。読者は以下のようにタイプして情報を得るように心がけて下さい。

```
$ sensible-pager "/etc/passwd"
```

ディレクトリー名はダブルクォーテションマーク間にタイプライターフォントで書き”/etc/apt/” のように表現されます。読者は以下のようにタイプして情報を得るように心がけて下さい。

```
$ mc "/etc/apt/"
```

パッケージ名はその名をタイプライターフォントで書き”vim” のように表現されます。読者は以下のようにタイプして情報を得るように心がけて下さい。

```
$ dpkg -L vim
$ apt-cache show vim
$ aptitude show vim
```

文書は､その場所のファイル名でダブルクォーテションマーク間にタイプライターフォントで書き”/usr/share/doc/bas
や”/usr/share/doc/base-passwd/users-and-groups.html” のように表現されたり、その場所の URL で https://www.debian.org のように表現されます。読者は以下のようにタイプして情報を得るように心がけて下さい。

```
$ zcat "/usr/share/doc/base-passwd/users-and-groups.txt.gz" | sensible-pager
$ sensible-browser "/usr/share/doc/base-passwd/users-and-groups.html"
$ sensible-browser "https://www.debian.org"
```

環境変数は、頭に”$” がついた名前をダブルクォーテションマーク間にタイプライターフォントで書き、”$TERM” のように表現されます。読者は以下のようにタイプして情報を得るように心がけて下さい。

```
$ echo "$TERM"
```

ポプコン

ポプコンのデーターは各パッケージの客観的人気の指標として提示されいます。それがダウンロードされた日付は 2026-11-28 09:48:51 UTC で、229395 を越すバイナリーパッケージ数と 26 のアーキテクチャーにまたがる 299780 つの提出レポートからなります。

注意
amd64 の不安定版 unstable アーカイブは現在高々 78709 つのパッケージしか含まないことを承知下さい。ポプコンデーターは多くの旧式設置システムからのレポートを含みます。

”votes” を意味する”V:” が前についたポプコンの数は”1000 * (PC で最近実行されたパッケージに関するポプコン提出)/(全ポプコン提出)” として計算される。

”installs” を意味する”I:” が前についたポプコンの数は”1000 * (PC にインストールされているパッケージに関するポプコン提出)/(全ポプコン提出)” として計算される。

注意
Popcon の数字はパッケージの重要性の絶対指標と考えるべきでません。統計を曲げる多くの因子があります。例えば、Popcon に参加しているシステムの一部は”/usr/bin” などのディレクトリーをシステム性能向上のために”noatime” オプションでマウントすることで当該システムから”vote” することを実質的に禁止しているかもしれません。

パッケージサイズ

各パッケージの客観的指標としてパッケージサイズデーターも提供されます。それは”apt-cache show” や”aptitude show” コマンドが (現在の amd64 アーキテクチャー上の unstable リリース上で) 表示する”Installed-Size:” です。サイズは KiB (Kibibyte = 1024 バイト単位) で表示されます。

注意
小さなパッケージサイズのパッケージは unstable リリース中の当該パッケージが内容のある他パッケージを依存関係でインストールするためのダミーパッケージだからかもしれません。

注意
”(*)” が後ろについたパッケージのサイズは、unstable リリース中にパッケージが無く experimental リリース中のパッケージサイズが代用されたことを示します。

本書へのバグ報告

何かこの文書に問題を発見した場合には、debian-reference パッケージに対して reportbug(1) を用いてバグ報告をして下さい。プレーンテキストバージョンかソースに対する”diff -u” による修正提案を含めて下さい。

新規ユーザーへのリマインダー

新規ユーザーへのリマインダーを以下に記します:

- データをバックアップします
  - 項 10.2 を参照下さい。
- パスワードとセキュリティーキーを保護します
- KISS (keep it simple stupid、簡潔性尊重原則)
  - システムを過剰にエンジニアリングしてはいけません
- ログファイルを読みます
  - 最初のエラーが大事なエラーです
- RTFM (read the fine manual、良く書かれているマニュアルを読みましょう)
- 質問する前にインターネットを検索しましょう
- 必要もないのに root になってはいけません
- パッケージ管理システムを改変してはいけません
- 自分が理解していないことを入力してはいけません
- (全セキュリティレビューを受ける前に) ファイルのパーミッションを変更してはいけません
- あなたの変更をテストするまで root シェルを離れてはいけません
- 代替ブートメディア (USB フラッシュドライブ、CD-ROM, …) を常に確保します。

新規ユーザーへの引用文

新規ユーザーを啓蒙する Debian のメーリングリストで見つけた興味深い引用文を記します。

- ”This is Unix. It gives you enough rope to hang yourself.” 「これは Unix です。首を括るのに十分なロープをあてがってくれますよ。」--- Miquel van Smoorenburg <miquels at cistron.nl>
- ”Unix IS user friendly... It’s just selective about who its friends are.” 「Unix はユーザーフレンドリー (使う人に優しい) です... 誰にフレンドリー (優しく) にするかの人見知りするだけです。」--- Tollef Fog Heen <tollef at add.no>

ウイキペディアの”Unix philosophy” という記事に、おもしろい格言集があります。

Chapter 1

GNU/Linux チュートリアル

コンピューターシステムを学ぶことは新しい外国語を学ぶことに似ていると考えます。チュートリアルブックは有用ですが、実際に自ら使って学ぶことが必要です。円滑なスタートが出きるように、いくつかの基本的なポイントを説明します。

Debian GNU/Linux の強力なデザインはマルチユーザー、マルチタスクという Unix オペレーティングシステムに由来します。これら Unix と GNU/Linux の特徴や類似点の強力さを活用することを覚えましょう。

Unix 対象の文書を避けたり、GNU/Linux に関する文書だけに頼ることは、有用な情報を見逃すことになるので止めましょう。

注意
Unix 的システムをコマンドラインツールで少々使った経験があれば、私がここで説明することはすべてご存知でしょう。リアリティーチェックと記憶を呼び戻すのにこれを使って下さい。

1.1 コンソールの基礎

1.1.1 シェルプロンプト

GNOME や KDE 等のデスクトップシステム等のような GUI 環境をインストールした場合以外には、システム起動の際に文字の login スクリーンが現れます。あなたのホスト名が `foo` と仮定すると、login プロンプトは次に示すような見えます。

GUI 環境をインストールした場合でも、Ctrl-Alt-F3 で文字ベースのログインプロンプトが出ますし、Ctrl-Alt-F2 で GUI 環境に戻れます (詳細は下記の項 1.1.6 を参照下さい)。

```
foo login:
```

login プロンプトであなたのユーザー名 (例えば `penguin`) を打鍵し Enter キーを押します。さらにあなたのパスワードを打鍵し Enter キーを再び押します。

注意
Unix の伝統に従い、Debian システムではユーザー名とパスワードに関して大文字小文字の区別をします。ユーザー名は通常小文字のみから選ばれます。最初のユーザーアカウントは通常インストールの際に作られます。追加のユーザーアカウントは root によって adduser(8) を用いて作られます。

”`/etc/motd`” (本日のメッセージ: Message Of The Day) に保存されている歓迎メッセージとコマンドプロンプトを表示しシステムが起動されます。

```
Debian GNU/Linux 12 foo tty3

foo login: penguin
Password:

Linux foo 6.5.0-0.deb12.4-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.5.10-1~bpo12+1 (2023-11-23) ↩
     x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

Last login: Wed Dec 20 09:39:00 JST 2023 on tty3
foo:~$
```

これであなたはシェルの中にいます。シェルはあなたからのコマンドを解釈します。

1.1.2 GUI の下でのシェルプロンプト

インストールの際に GUI 環境をインストールした場合、システム起動時にグラフィカルなログイン画面が表示されます。あなたのユーザー名とパスワードを入力することで非特権ユーザーアカウントにログインできます。タブ (tab) を用いたりマウスの第一クリックを用いるとユーザー名とパスワードの間を行き来できます。

gnome-terminal(1) や konsole(1) や xterm(1) のような `x-terminal-emulator` プログラムを GUI 環境下で起動するとシェルプロンプトが得られます。GNOME デスクトップ環境下では、”Applications” → ”Accessories” → ”Terminal” とクリックしてもうまくいきます。

デスクトップ環境 (例えば `fluxbox`) 次第ではメニューの起点がよく分からないことがあります。そんな時はデスクトップスクリーンの背景を (右) クリックしてメニューが表示されることを期待しましょう。

1.1.3 root アカウント

root アカウントはスーパーユーザーとか特権ユーザーとも呼ばれます。このアカウントからは以下のようなシステム管理活動ができます。

- ファイルパーミッションによらずシステム上の任意ファイルに関しての、読出しと書込みと削除
- システム上のいかなるファイルに関して、ファイルの所有者やパーミッション設定
- システム上の非特権ユーザーのパスワードを設定
- パスワード無しに任意アカウントへの login

root アカウントの権限を使うには、この無制限の権限ゆえ配慮と責任ある行動が求められます。



警告
root のパスワードを第三者に決して教えてはいけません。

注意
ファイル (Debian システムにとってはファイルの一種である CD-ROM 等のハードウエアーデバイスも含む) パーミッションは、非 root ユーザーによるそのファイルの使用やアクセスをできなくなくすることがあります。この様な状況の下では root アカウントを使うことが簡便なテスト法ですが、問題解決はファイルパーミッションとユーザーのグループのメンバーシップを適正に設定する必要があります (項 1.2.3 を参照下さい)。

1.1.4 root シェルプロンプト

root のパスワードを使って root のシェルプロンプトを使えるようにする基本的な方法を次に記します。

- 文字ベースのログインプロンプトに `root` と入力します。
- どのユーザーシェルプロンプトからでも”`su -l`” と入力します。
  - 現ユーザーの環境を一切引き継がません。
- どのユーザーシェルプロンプトからでも”su” と入力します。
  - 現ユーザーの環境を一部引き継ぐ。

1.1.5 GUI のシステム管理ツール

デスクトップのメニューが GUI のシステム管理ツールを適切な権限とともに自動的に起動しない場合、gnome-terminal(1) や konsole(1) や xterm(1) のような X ターミナルエミュレーターの root シェルプロンプトから起動できます。項 1.1.4 and 項 7.9 を参照下さい。



警告
gdm3(1) 等のディスプレーマネージャーのプロンプトに `root` と入力して、GUI ディスプレー / セッションマネージャーを root アカウントのもとで決して起動してはいけません。
あなたの X スクリーンを覗き見られるかもしれないので、X Window 下で信頼できないリモートの GUI プログラムを決して実行してはいけません。

1.1.6 仮想コンソール

デフォルトの Debian システムでは、6 つの切り替え可能な VT100 様の文字コンソールが利用でき、Linux ホスト上で直接コマンドシェルを起動できます。GUI 環境下でない場合は、`Left-Alt-key` と `F1` —`F6` の中の一つのキーを同時に押すことで仮想コンソール間の切り替えができます。それぞれの文字コンソールはアカウントに独立してログインすることができ、マルチユーザー環境を提供します。このマルチユーザー環境は Unix の素晴らしい機能で、癖になります。

GUI 環境下では、`Ctrl-Alt-F3` キーを押す、つまり `left-Ctrl-key` と `left-Alt-key` と `F3-key` キーを同時に押すと文字コンソール 3 にアクセスできます。普通仮想コンソール 2 で実行されている GUI 環境には `Alt-F2` を押すことにより戻れます。

これとは別の方法で、例えば仮想ターミナル 3 という今とは違う仮想ターミナルへの変更がコマンドラインから出来ます。

```
# chvt 3
```

1.1.7 コマンドプロンプトからの退出方法

コマンドプロンプトで `Ctrl-D`、つまり `left-Ctrl-key` と `d-key` の同時押しをするとシェルでの活動を終了できます。文字コンソールの場合は、こうすると login プロンプト戻ります。これらのコントロール文字は”control D”と大文字を使って表記されますが、Shift キーを押す必要はありません。また `Ctrl-D` に関する簡略表記 `^D` も使われます。この代わりに”exit” とタイプすることができます。

x-terminal-emulator(1) にあっては、このようにすることで `x-terminal-emulator` のウィンドウが閉じることができます。

1.1.8 システムをシャットダウンする方法

ファイル操作の際にパーフォーマンス向上のためにメモリーへのデーターのキャッシュがされる他の現代的な OS と同様に、Debian システムでも電源を安全に切る前に適正なシャットダウン手順を取る必要があります。これはすべてのメモリー上の変更を強制的にディスクに書き出しすことで、ファイルの完全性を維持するためです。ソフトウェアー電源コントロールが利用できる場合、シャットダウン手続きはシステムの電源を自動的に落とします。(これがうまくいかない時には、シャットダウン手続きの後で数秒間電源ボタンを押す必要があるかもしれません。)

普通のマルチユーザーモードからのシステムのシャットダウンがコマンドラインから出来ます。

```
# shutdown -h now
```

シングルユーザーモードからのシステムのシャットダウンがコマンドラインから出来ます。

```
# poweroff -i -f
```

項 6.3.8 を参照下さい。

1.1.9 合理的なコンソールの復元

例えば”`cat` *`some-binary-file`*” のような変な事をした後でスクリーンが無茶苦茶になった場合、コマンドプロンプトに”`reset`” と入力して下さい。このときコマンドを入力してもスクリーンには読み取れる表示がされないかもしれません。”`clear`” とすればスクリーンが消去できます。

1.1.10 初心者向け追加パッケージの提案

デスクトップ環境タスク抜きの最小限インストール Debian システムですら基本的な Unix 機能は提供されますが、コマンドラインや curses に基づく `mc` や `vim` 等のいくつかの文字ターミナルパッケージを apt-get(8) を使って以下のように追加インストールすることから始めることを初心者にお薦めします。

```
# apt-get update
 ...
# apt-get install mc vim sudo aptitude
 ...
```

既にこれらのパッケージがインストールされている場合には、新しいパッケージはインストールされません。

いくつかの参考資料を読むのも良いことです。

これらのパッケージの一部を以下のようにしてインストールします。

```
# apt-get install package_name
```

パッケージ	ポプコン	サイズ	説明
mc	V:42, I:172	1590	テキストモードの全画面ファイルマネージャー
sudo	V:739, I:870	6817	ユーザーに限定的な root 権限を与えるプログラム
vim	V:73, I:330	4415	Unix テキストエディター Vi IMproved (改良版 Vi)、プログラマーのためのテキストエディター (標準版)
vim-tiny	V:55, I:979	1931	Unix テキストエディター Vi IMproved (改良版 Vi)、プログラマーのためのテキストエディター (軽量版)
emacs-nox	V:3, I:12	46564	GNU プロジェクト Emacs、Lisp に基づく拡張可能なテキストエディター
w3m	V:11, I:125	2853	テキストモード WWW ブラウザー
gpm	V:8.0, I:8.7	524	テキストコンソール上の Unix 式のカットアンドペースト (daemon)

Table 1.1: 興味あるテキストモードのプログラムパッケージのリスト

パッケージ	ポプコン	サイズ	説明
doc-debian	I:887	185	Debian プロジェクトの文書、(Debian FAQ) 他
debian-policy	I:6.4	5147	Debian ポリシーマニュアルと関連文書
developers-reference	V:0.2, I:2.0	2656	Debian 開発者のためのガイドラインと情報
debmake-doc	I:0.32	11807	Debian メンテナ向けガイド
debian-history	I:0.49	6513	Debian プロジェクトの歴史
debian-faq	I:886	791	Debian FAQ (よくある質問集)

Table 1.2: 有用な文書パッケージのリスト

1.1.11 追加のユーザーアカウント

以下の練習のためにあなたのメインのユーザーアカウントを使いたくない場合には、例えば `fish` という追加のユーザーアカウントを作ります。root シェルプロンプトで以下のように入力します。

```
# adduser fish
```

すべての質問に返事をします。

こうすることで `fish` という名前の新規アカウントが作られます。練習の後で、このユーザーとそのホームディレクトリーは以下のようのすれば削除できます。

```
# deluser --remove-home fish
```

非 Debian や特殊な Debian システム上では、より下層の useradd(8) や userdel(8) ユーティリティーで上記の操作を代用する必要があります。

1.1.12 sudo の設定

ラップトップ PC 上のデスクトップの Debian システム等のような典型的単一ユーザーワークステーションでは以下のような単純な sudo(8) の設定をして、非特権ユーザー (例えば `penguin`) に管理者権限を (root パスワードではなく) ユーザー自身のパスワードで与えることがよくあります。

```
# echo "penguin  ALL=(ALL) ALL" >> /etc/sudoers
```

これに代え、以下のようにして非特権ユーザー `penguin` にパスワード一切無しに管理者権限を与えることもよくあります。

```
# echo "penguin  ALL=(ALL) NOPASSWD:ALL" >> /etc/sudoers
```

ワークステーション上であなた自身が管理者で唯一のユーザーである場合にのみ、このトリックを使用して下さい。



警告
システムセキュリティー上非常に悪い事態を招くので、ユーザが複数いるワークステーション上で通常のユーザーアカウントに対してこの様な設定をしてはいけません。



注意
上記例のような `penguin` のパスワードとアカウントは root パスワードや root アカウント同様の保護が必要です。
この文脈上の管理者権限はワークステーションに関するシステム管理業務をする権限を与えられた人に属します。そのような権限と能力を持っていなければ、あなたの会社の管理部門の管理職や上司とはいえこのような権限を与えてはいけません。

注意
特定デバイスや特定ファイルへのアクセスの権限を与えるには、sudo(8) をつかって得た `root` 権限を用いるのではなく、**group** を使って限定的アクセス与えることを考えるべきです。
sudo(8) を使ってもう少し工夫された注意深い設定をすれば、共有システム上の他のユーザーに root パスワードを教えること無く限定的管理権限を許可することができます。こうすることは、誰が何をしたかを明らかにするので、複数の管理者がいるホストにおける責任の所在を明らかにします。ただ、誰にもそんな権限を与えたく無いかもしれません。

1.1.13 お遊びの時間

非特権ユーザーアカウントを使う限り全くリスク無く Debian システムでお遊びをする準備万端です。

Debian システムを Unix 的システムとして次に学びましょう。

- 項 1.2 (基本コンセプト)
- 項 1.3 (サバイバル手法)
- 項 1.4 (基本手法)
- 項 1.5 (シェルのメカニズム)
- 項 1.6 (文字処理手法)

1.2 Unix-like ファイルシステム

GNU/Linux や他の Unix 的オペレーティングシステムでは、ファイルはディレクトリーに整理されています。すべてのファイルやディレクトリーは、”/” を根 (root) に持つ一本の大きな木 (ツリー) のようにアレンジされています。

このようなファイルやディレクトリーはいくつかのデバイスに展開することができます。あるデバイス上にあるファイルシステムを大きなファイルツリーにマウントするのに mount(8) が使われます。その逆に、それを切り離すのに umount(8) が使われます。最近の Linux カーネルでは、mount(8) をオプションとともに用いると、ファイルツリーの一部を別のところと結びつけたり、ファイルシステムを共有か非共有か従属かバインド不可としてマウントもできます。各ファイルシステムごとの利用可能なマウントオプションは”`/usr/share/doc/linux-doc-*/Documentation/filesystems/`” にあります。

Unix システム上のディレクトリーは、一部の他システム上ではフォルダと呼ばれます。Unix システム上では”`A:`” のようなドライブというコンセプトが無いこと覚えておいて下さい。単一のファイルシステムがあって、そこにすべてが含まれています。これは Windows と比べた際の大きな利点です。

1.2.1 Unix ファイルの基礎

Unix ファイルの基礎は以下です。

- ファイル名は大文字と小文字を区別します。”`MYFILE`” と”`MyFile`” は異なるファイルです。
- ルートディレクトリーはフィルシステムの根 (ルート、root) を意味して、単に”/” と記載されます。これを root ユーザーのホームディレクトリー”`/root`” とは混同しないで下さい。
- 全てのディレクトリーには**”/”** 以外の文字か記号からなる名前がついています。ルートディレクトリーは例外で、その名前は”/” (” スラッシュ” とか” ルートディレクトリー” と読まれます) でその名前を変えることはできません。
- 各ファイルやディレクトリーは、たどっていくとファイルに到達するディレクトリーの列が示される、完全に記述したファイル名とか絶対ファイル名とかパスにより指定されます。これらの 3 つの表現は同義語です。
- 全ての完全に記述したファイル名は”/” ディレクトリーで始まり、ファイル名中の各ディレクトリーやファイル名の間には”/” がはさまります。最初の”/” はディレクトリー名です。その他の”/” は、以下のサブディレクトリーとの区別をします。そして最後には実際のファイルの名前がきます。ちょっと混乱しそうですので、以下の完全に記述したファイル名の例をご覧下さい: ”`/usr/share/keytables/us.map.gz`”。一方このベース名である、”`us.map.gz`” だけをファイル名と呼ぶ人もあります。
- ルートファイルシステムは”`/etc/`” や”`/usr/`” のような複数の枝を持ちます。これらのサブディレクトリーもまた”`/etc/systemd/`” や”`/usr/local/`” のように、さらにサブディレクトリーに枝別れします。これらの全体をまとめてディレクトリーツリーと呼びます。絶対ファイル名はツリーの根元 (”/”) から枝の先 (ファイル) までの経路として考えることもできます。また、あたかもディレクトリーツリーをルートディレクトリー (”/”) という単

一人物の全直系に広がる家系図のように人が話すのを聞いたことがあるでしょう。あたかもそれぞれのサブディレクトリーに親があるとし、パスはファイルの完全な祖先の系図のように表現します。ルートディレクトリーではない他の場所から始まる相対パスもあります。ディレクトリー"`../`" は親ディレクトリーを参照していることを覚えておきましょう。このような呼び方はディレクトリーのような構造を持つ他の階層的ツリー状のデーター構造体でもよく使われます。

- ハードディスクのような物理デバイスに対応したパス名の要素は存在しません。ここが、パス名に"`C:\`" のようなデバイス名が含まれる RT-11 や CP/M や OpenVMS や MS-DOS や AmigaOS や Microsoft Windows と違う点です。(但し、普通のファイルシステム中に物理デバイスを示すディレクトリー項目はあります。項 1.2.2 を参照下さい。)

注意
ほとんど全ての文字や記号をファイル名中に使えますが、実際そうすることは賢明ではありません。スペースやタブや改行や他の特殊文字: `{ } ( ) [ ] ' ` " \ / > < | ; ! # & ^ * % @ $` はコマンドラインで特別な意味を持つので避けるべきです。名前の中の単語間には、ピリオドやハイフンや下線を選んで区別します。各語頭を"`LikeThis`" のように語頭を大文字にすることもできます。経験を積んだ Linux のユーザーはファイル名中にスペースが入ることを避けます。

注意
"root" (ルート) と言う言葉は"root ユーザー" と言う意味でも" ルートディレクトリー" と言う意味でも使われます。それがいずれかは使われている文脈から明かです。

注意
パスと言う言葉は上述の完全に記述したファイル名に関して使われるばかりではなくコマンドサーチパスにも使われます。どちらの意味かは文脈から通常明かです。

ファイル階層について詳細に学ぶ最も良い方法は、Filesystem Hierarchy Standard ("`/usr/share/doc/debian-policy/`" や hier(7)) に記述されています。手始めとして以下の事実を覚えるべきです。

ディレクトリー	ディレクトリーの用途
`/`	ルートディレクトリー
`/etc/`	システム全体の設定ファイル
`/var/log/`	システムのログファイル
`/home/`	全ての非特権ユーザーのホームディレクトリー

Table 1.3: 重要ディレクトリーの使い方のリスト

1.2.2 ファイルシステムの内側

Unix の伝統に従い、Debian/Linux システムはハードディスクや他のストレージデバイス上に存在する物理データーを表すファイルシステムを提供し、コンソールスクリーンやリモートのシリアルコンソールなどのハードウェアデバイスとの相互作用が"`/dev/`" の下に統一された形式で表されています。

Debian/Linux システム上の、各々のファイルやディレクトリーや名前付きパイプ (2 つのプログラムがデーターを共有する方法) や物理デバイスは、それぞれの所有者 (owner) やデーターが所属するグループ (group) や最終アクセス時間などの付帯属性 (attribute) を記述する inode と呼ばれるデーター構造を持ちます。ほとんど全てをファイルシステム表現しようというアイデアは Unix の発明でしたし、現代的な Linux カーネルはこのアイデアを一歩進めています。コンピューター上で実行されているプロセス情報さえファイルシステム中に見つけられます。

このような物理的実体と内部プロセスの抽象的かつ統一された表現は非常にパワフルなので、多くの全く異なるデバイスに同じコマンドを使用して同種の操作が行えます。実行中のプロセスに繋がった特殊なファイルにデーターを書き込むことでカーネルが如何に動作するかまで変更できます。

ティップ
ファイルツリーや物理的実体の間の関係を確認する必要がある際には、mount(8) を引数無しで実行して下さい。

1.2.3 ファイルシステムのパーミッション

Unix 的システムのファイルシステムのパーミッションは以下の 3 つの影響されるユーザーのカテゴリーのために定義されています。

- ファイルを所有するユーザー **(user)** (**u**)
- ファイルが所属するグループ **(group)** 中の他ユーザー (**g**)
- ” 世界” や” 全員” とも呼ばれる、全第三者 **(other)** ユーザー (**o**)

ファイルでは、それぞれに対応するパーミッションは以下のようになります。

- 読出し **(read)** (**r**) パーミッションはファイル内容確認を可能にします。
- 書込み **(write)** (**w**) パーミッションはファイル内容変更を可能にします。
- 実行 **(execute)** (**x**) パーミッションはファイルをコマンド実行を可能にします。

ディレクトリーでは、対応するパーミッションはそれぞれ以下のようになります。

- 読出し **(read)** (**r**) パーミッションはディレクトリー内容リストを可能にします。
- 書込み **(write)** (**w**) パーミッションはディレクトリーへのファイルの追加削除を可能にします。
- 実行 **(execute)** (**x**) パーミッションはディレクトリー内のファイルへのアクセスを可能にします。

ここで、ディレクトリーに関する実行 **(execute)** パーミッションとはディレクトリー内のファイルへの読出しを許可するのみならず、サイズや変更時間のようなアトリビュート閲覧を許可します。

ファイルやディレクトリーのパーミッション情報他を表示するには、ls(1) が使われます。”`-l`” オプション付きでこれが起動されると、以下の情報がこの順序で表示されます。

- ファイルのタイプ (最初の文字)
- ファイルのアクセスパーミッション (以下の 9 文字。ユーザーとグループと第三者の順にそれぞれに対して 3 文字から構成されている)
- ファイルへのハードリンク数
- ファイルを所有するユーザー **(user)** の名前
- ファイルが所属するグループ **(group)**
- ファイルのサイズ (文字数、バイト)
- ファイルの日時 (mtime)
- ファイルの名前

root アカウントから chown(1) を使用することでファイルの所有者を変更します。ファイルの所有者又は root アカウントから chgrp(1) を使用することでファイルのグループを変更します。ファイルの所有者又は root アカウントから chmod(1) を使用することでファイルやディレクトリーのアクセスパーミッションを変更します。`foo` ファイルの操作の基本的文法は以下の通り。

文字	意味
-	通常ファイル
d	ディレクトリー
l	シムリンク
c	文字デバイス名
b	ブロックデバイス名
p	名前付きパイプ
s	ソケット

Table 1.4: "ls -l" の出力の最初の文字のリスト

```
# chown newowner foo
# chgrp newgroup foo
# chmod  [ugoa][+-=][rwxXst][,...] foo
```

例えば以下のようにするとディレクトリーツリーの所有者をユーザー foo に変更しグループ bar で共有できます。

```
# cd /some/location/
# chown -R foo:bar .
# chmod -R ug+rwX,o=rX .
```

更に特殊なパーミッションビットが 3 つ存在します。

- セットユーザー **ID** ビット (ユーザーの **x** に代えて **s** か **S**)
- セットグループ **ID** ビット (グループの **x** に代えて **s** か **S**)
- スティッキービット (第三者の **x** に代えて **t** か **T**)

ここで、これらのビットの"ls -l" のアウトプットはこれらの出力によってかくされた実行ビットが非設定 **(unset)** の場合大文字となります。

セットユーザー **ID** を実行ファイルにセットすると、ユーザーはファイルの所有者 ID (例えば、**root**) を使って実行ファイルを実行することを許可されます。同様に、セットグループ **ID** を実行ファイルにセットすると、ユーザーはファイルのグループ ID (例えば、**root**) を使って実行ファイルを実行することを許可されます。これらの設定はセキュリティーを破壊するリスクを引き起こすので、これらのビットを有効にするには特別な注意が必要です。

セットグループ **ID** をディレクトリーに対して設定すると、BSD 的ファイル作成手法が有効になり、ディレクトリーに作成した全ファイルが所属するグループがディレクトリーのものになります。

スティッキビットをディレクトリーに対して有効にすると、ディレクトリーにあるファイルがファイルの所有者以外から削除されるのを防ぎます。"/tmp" のような全員書込み可能ディレクトリーやグループ書込み可能なディレクトリーなどのにあるファイルの内容を安全にするためには、書込みパーミッションを無効にするだけでなく、ディレクトリーにスティッキビットもセットする必要があります。さもなければ、ディレクトリーに書込みアクセスできるユーザーにより、ファイルが削除され、同じ名前で新規ファイルが作成されることを許してしまいます。

ファイルパーミッションの興味ある例を次にいくつか示します。

```
$ ls -l /etc/passwd /etc/shadow /dev/ppp /usr/sbin/exim4
crw------T 1 root root   108, 0 Oct 16 20:57 /dev/ppp
-rw-r--r-- 1 root root     2761 Aug 30 10:38 /etc/passwd
-rw-r----- 1 root shadow   1695 Aug 30 10:38 /etc/shadow
-rwsr-xr-x 1 root root   973824 Sep 23 20:04 /usr/sbin/exim4
$ ls -ld /tmp /var/tmp /usr/local /var/mail /usr/src
drwxrwxrwt 14 root root  20480 Oct 16 21:25 /tmp
drwxrwsr-x 10 root staff  4096 Sep 29 22:50 /usr/local
drwxr-xr-x 10 root root   4096 Oct 11 00:28 /usr/src
drwxrwsr-x  2 root mail   4096 Oct 15 21:40 /var/mail
drwxrwxrwt  3 root root   4096 Oct 16 21:20 /var/tmp
```

数字	意味
1 桁目 (任意)	セットユーザー **ID** (=4) とセットグループ **ID** (=2) とスティキービット (=1) の和
2 桁目	ユーザーに関する、読出し **(read)** (=4) と書込み **(write)** (=2) と実行 **(execute)** (=1) のファイルパーミッションの和
3 桁目	グループに関して、同上
4 桁目	第三者に関して、同上

Table 1.5: chmod(1) コマンドで用いられるファイルパーミッションの数字モード

chmod(1) を用いて、ファイルパーミッションを記述するためのもう一つの数字モードが存在します。この数字モードは 8 進数を使った 3 桁から 4 桁の数字を用います。

これは複雑に聞こえるかもしれませんが、実際は本当にシンプルです。"`ls -l`" コマンドの出力の最初の数列 (2〜10 列) を見て、それをファイルパーミッションのバイナリー表記 (2 進数) ("-" を"0"、"rwx" を"1") として読むと、この数字モードの値はファイルパーミッションの 8 進数表現として意味を持ちます。

例えば、次を試してみて下さい:

```
$ touch foo bar
$ chmod u=rw,go=r foo
$ chmod 644 bar
$ ls -l foo bar
-rw-r--r-- 1 penguin penguin 0 Oct 16 21:39 bar
-rw-r--r-- 1 penguin penguin 0 Oct 16 21:35 foo
```

ティップ
シェルスクリプトから"`ls -l`" で表示される情報にアクセスする必要がある際には、test(1) や stat(1) や readlink(1) のような適切なコマンドの使用を考えるべきです。シェル組込みコマンドの"[" や"`test`" を使うのも手です。

1.2.4 新規作成ファイルのパーミッションのコントロール: umask

新規作成ファイルのやディレクトリーに適用されるパーミッションは `umask` シェル組込みコマンドを使うことにより制限できます。dash(1) か bash(1) か bash-builtins(7) をご覧下さい。

```
(file permissions) = (requested file permissions) & ~(umask value)
```

umask	作成されるファイルパーミッション	作成されるディレクトリーパーミッション	使い方
`0022`	`-rw-r--r--`	`-rwxr-xr-x`	ユーザーのみにより書込み可
`0002`	`-rw-rw-r--`	`-rwxrwxr-x`	グループにより書込み可

Table 1.6: **umask** 値の例

Debian システムはユーザー専用グループ (UPG) 方式をデフォルトで使用します。新規ユーザーがシステムに追加される毎に UPG は作成されます。UPG はそのグループを作成したユーザーと同じ名前を持ち、そのユーザーが UPG の唯一のメンバーです。UPG 方式では全ユーザーが各自専用のグループを持つので、umask を `0002` に設定しても安全です。(一部 Unix 系システムでは全一般ユーザーを 1 つの **users** グループに所属させることがよく行われます。そのような場合には安全のため umask を `0022` に設定した方がよいでしょう。)

ティップ
`~/.bashrc` ファイル中に"`umask 002`" と書いて UPG を有効にしましょう。

1.2.5 ユーザーのグループ (group) のパーミッション



警告
リブートやそれに類する行為をする前に未保存の変更を保存しましょう。

ユーザー penguin をグループ bird に 2 ステップで追加できます:

- 以下のいずれか一つでグループ設定を変更します。
  - "sudo usermod -aG bird penguin" を実行します。
  - "sudo adduser penguin bird" を実行します。(典型的 Debian システム上のみ)
  - /etc/group に"sudo vigr" を、/etc/gshadow に"sudo vigr -s" を実行し bird の行に penguin を追記します。
- 以下のいずれか一つを使い設定を適用します。
  - コールドリブートとログイン。(最適選択肢)
  - GUI メニュー経由でのログアウト。(現代的デスクトップ環境下では、上手く行かないかもしれません。)

ユーザー penguin をグループ bird から 2 ステップで削除できます:

- 以下のいずれか一つでグループ設定を変更します。
  - "sudo usermod -rG bird penguin" を実行します。
  - "sudo deluser penguin bird" を実行します。(典型的 Debian システム上のみ)
  - /etc/group に"sudo vigr" を、/etc/gshadow に"sudo vigr -s" を実行し bird の行の penguin を削除します。
- 以下のいずれか一つを使い設定を適用します。
  - コールドリブートとログイン。(最適選択肢)
  - "kill -TERM -1" を実行し、"systemctl restart NetworkManager.service" 等のような何らかの修正操作をします。
  - GUI 経由でのログアウトは Gnome Desktop では選択肢ではありません。

現代的なデスクトップ環境では、何らかのワームリブートをしようとしても本物のコールドリブートの脆弱な代用品にしかなりません。

注意
もし"auth optional pam_group.so" 行が"/etc/pam.d/common-auth" に書き加えれ、"/etc/security/group.conf" に対応する設定がされていれば、実際のユーザーのグループメンバーシップは動的に割り当てられます。(第 4 章を参照下さい。)

ハードウエアーデバイスは Debian システム上では一種のファイルでしかありません。USB フラッシュドライブや CD-ROM のようなデバイスをユーザーアカウントからアクセスするのに問題があった場合にはそのユーザーを該当するグループのメンバーにします。

いくつかのシステムが供給するグループはそのメンバーに root 権限無しに特定のファイルやデバイスにアクセスすることを可能にします。

グループ	アクセスできるファイルやデバイスの説明
dialout	シリアルポート (”/dev/ttyS[0-3]”) への全面的かつ直接のアクセス
dip	信頼できるピアーにダイヤルアップ **IP** 接続をするためのシリアルポートへの制限付きアクセス
cdrom	CD-ROM や DVD+/-RW のドライバー
audio	音声デバイス
video	映像デバイス
scanner	スキャナー
adm	システムモニターのログ
staff	下級管理業務のためのディレクトリー: ”/usr/local”、”/home”

Table 1.7: ファイルアクセスのためにシステムが供給する特記すべきグループのリスト

ティップ
モデムの設定をしたりどこにでも電話したり等するには dialout グループに所属する必要があります。もし信頼できるピアーに関する事前定義された設定ファイル”/etc/ppp/peers/” が root によって作られていると、dip グループに属するだけで pppd(8) や pon(1) や poff(1) コマンドを用いてダイヤルアップ **IP** 接続が作成できます。

いくつかのシステムが供給するグループはそのメンバーに root 権限無しに特定のコマンドを実行することを可能にします。

グループ	実行可能なコマンド
sudo	いかなるコマンドでもスーパーユーザー特権下で実行
lpadmin	プリンターのデーターベースからプリンターを追加や変更や削除するコマンドを実行

Table 1.8: 特定コマンド実行のためにシステムが供給する特記すべきグループのリスト

システムが供給するユーザーやグループの完全なリストは、base-passwd パッケージが供給する”/usr/share/doc/bas
の中にある最新バージョンの”Users and Groups” 文書を参照下さい。

ユーザーやグループシステムを管理するコマンドは passwd(5) や group(5) や shadow(5) や newgrp(1) や vipw(8) や vigr(8) や pam_group(8) を参照下さい。

1.2.6 タイムスタンプ

GNU/Linux ファイルのタイムスタンプには 3 種類あります。

タイプ	意味 **(**歴史的 **Unix** 定義**)**
mtime	ファイル内容変更時間 (ls -l)
ctime	ファイル状態変更時間 (ls -lc)
atime	ファイル最終アクセス時間 (ls -lu)

Table 1.9: タイムスタンプのタイプのリスト

注意
ctime はファイル作成日時ではありません。

注意
GNU/Linux システム上では、実際の **atime** 値は歴史的 Unix 定義とは異なる場合があります。

- ファイルが上書きされると、ファイルの **mtime** と **ctime** と **atime** の属性すべてが変更されます。
- ファイルの所有者やパーミッションの変更をすると、ファイルの **ctime** や **atime** アトリビュートを変えます。
- 伝統的 Unix システム上ではファイルを読むとファイルの **atime** 属性が変更されます。
- GNU/Linux システム上では、"`strictatime`" でファイルシステムをマウントした場合にファイルを読むとファイルの **atime** が変更されます。
- ファイルを初めて読み込んだときか、1 日空けてアクセスした場合、ファイルの **atime** 属性の更新が GNU/Linux (Linux 2.6.30 以降) の"`relatime`" でマウントされているファイルシステムでは生じます。
- **atime** 属性は `noatime` でマウントされているファイルシステムでは、読み込み時に更新されることはありません。

注意
"`noatime`" や"`relatime`" マウントオプションは普通使用状況下でのファイルシステムの読み出しパーフォーマンスを向上させるために導入されました。"`strictatime`" オプション下の単純なファイル読み出しオペレーションは **atime** 属性を更新する時間のかかる書き込み操作を引き起こします。しかし、**atime** 属性は mbox(5) ファイルを除くとほとんど使われることはありません。mount(8) を参照下さい。

既存ファイルのタイムスタンプを変更するには touch(1) コマンドを使って下さい。

タイムスタンプに関して、`ls` コマンドは非英語ロケール ("`fr_FR.UTF-8`") でローカライズされた文字列を表示します。

```
$ LANG=C  ls -l foo
-rw-rw-r-- 1 penguin penguin 0 Oct 16 21:35 foo
$ LANG=en_US.UTF-8  ls -l foo
-rw-rw-r-- 1 penguin penguin 0 Oct 16 21:35 foo
$ LANG=fr_FR.UTF-8  ls -l foo
-rw-rw-r-- 1 penguin penguin 0 oct. 16 21:35 foo
```

ティップ
"`ls -l`" の出力のカスタム化は項 9.3.4 を参照下さい。

1.2.7 リンク

"`foo`" というファイルを異なるファイル名"`bar`" に結びつけるのには 2 つの方法があります。

- Hard link
  - 既存ファイルの重複名
  - "`ln foo bar`"
- シンボリックリンクもしくはシムリンク
  - 他のファイルをその名前で指す特殊ファイル
  - "`ln -s foo bar`"

リンク数の変化と `rm` コマンドの結果の微妙な違いについての以下の例をご覧下さい。

```
$ umask 002
$ echo "Original Content" > foo
$ ls -li foo
1449840 -rw-rw-r-- 1 penguin penguin 17 Oct 16 21:42 foo
$ ln foo bar     # hard link
$ ln -s foo baz  # symlink
$ ls -li foo bar baz
1449840 -rw-rw-r-- 2 penguin penguin 17 Oct 16 21:42 bar
1450180 lrwxrwxrwx 1 penguin penguin  3 Oct 16 21:47 baz -> foo
1449840 -rw-rw-r-- 2 penguin penguin 17 Oct 16 21:42 foo
$ rm foo
$ echo "New Content" > foo
$ ls -li foo bar baz
1449840 -rw-rw-r-- 1 penguin penguin 17 Oct 16 21:42 bar
1450180 lrwxrwxrwx 1 penguin penguin  3 Oct 16 21:47 baz -> foo
1450183 -rw-rw-r-- 1 penguin penguin 12 Oct 16 21:48 foo
$ cat bar
Original Content
$ cat baz
New Content
```

ハードリンクは同一ファイルシステム内に作ることができ、ls(1) コマンドに”-i” オプションを使って表示される inode 番号が同じです。

シンボリックリンクは上の例に示したように、常にファイルアクセスパーミッション”rwxrwxrwx” を持ちますので、シンボリックリンクが指すファイルのアクセスパーミッションが有効ファイルアクセスパーミッションとなります。



注意
もし特段の理由がないなら複雑なシンボリックリンクやハードリンクを作らない方が一般的には良いでしょう。シンボリックリンクの論理的組み合わせがファイルシステム中でループになっているという悪夢を引き起こすかもしれません。

注意
もしハードリンクを使う特段の理由がなければ、ハードリンクよりシンボリックリンクを使う方が一般的には良いでしょう。

”.” ディレクトリーは、それ自身が中にあるディレクトリーとリンクしていますので、新規ディレクトリーのリンク数は 2 から始まります。”..” ディレクトリーは親ディレクトリーとリンクしているので、ディレクトリーのリンク数は新規サブディレクトリーの増加に伴い増加します。

もし最近あなたが Windows から Linux に移動してきたなら、Unix のファイル名のリンクは Windows 上でもっとも似ている”shortcuts” との比較で如何にうまくできているかにすぐ気づくでしょう。ファイルシステム中に実装されているのでアプリケーションからはリンクされたファイルなのかオリジナルなのかの区別がつきません。ハードリンクの場合は実際全く違いはありません。

1.2.8 名前付きパイプ (FIFO)

名前付きパイプは、パイプのように働くファイルです。何かをファイルに入れると、もう一方の端からそれが出てきます。こうしてこれは FIFO または First-In-First-Out (先入れ先出し) と呼ばれます。つまり、最初にパイプに入れられたものが最初にもう一方の端から出てきます。

名前付きパイプに書き込む場合、パイプに書き込むプロセスは情報がパイプから読出されるまで終了しません。名前付きパイプから読み出す場合、読出すプロセス何か読出すものが無くなるまで終了するのを待ちます。パイプの

サイズは常に 0 です。--- 名前付きパイプはデーターを保存せず、シェルの”|” というシンタクッスが提供する機能ように 2 つのプロセスをリンクするだけです。しかし、このパイプは名前を持つので、2 つのプロセスは同じコマンドラインになくても良いし、同じユーザーにより実行される必要さえありません。パイプは Unix の非常に影響力ある発明でした。

例えば、次を試してみて下さい:

```
$ cd; mkfifo mypipe
$ echo "hello" >mypipe & # put into background
[1] 8022
$ ls -l mypipe
prw-rw-r-- 1 penguin penguin 0 Oct 16 21:49 mypipe
$ cat mypipe
hello
[1]+  Done                    echo "hello" >mypipe
$ ls mypipe
mypipe
$ rm mypipe
```

1.2.9 ソケット

ソケットはインターネットのコミュニケーションやデーターベースやオペレーティングシステム自身によって頻繁に使われます。ソケットは名前つきパイプ (FIFO) に似ており、異なるコンピューター間でさえプロセス間の情報交換を可能にします。ソケットにとって、これらのプロセスは同時に実行する必要も、同じ祖先プロセスの子供として実行する必要もありません。これはプロセス間通信 (IPC) の終端点です。ネットワーク越しで異なるホストの間で情報の交換をすることも可能です。2 つの典型的なソケットは、インターネットソケットと Unix ドメインソケットです。

ティップ
(iproute2 パッケージが提供する) ”`ss -t -u -a`” を実行すると特定のシステム上のソケットの全般状況がよく分かります。

1.2.10 デバイスファイル

デバイスファイルは、システム上のハードディスク、ビデオカード、ディスプレー、キーボードなどの物理デバイス又は仮想デバイス等を意味します。仮想デバイスの例として”`/dev/console`” として表されるコンソールがあります。

2 タイプのデバイスファイルがあります。

- 文字デバイス
  - 1 文字毎にアクセス可能
  - 1 文字 = 1 バイト
  - 例: キーボードデバイス、シリアルポート等
- ブロックデバイス
  - 比較的大きなブロック単位でアクセス可能
  - 1 ブロック > 1 バイト
  - 例: ハードディスク、…

デバイスファイルの読出しと書込みは可能ですが、おそらく人間にとっては意味不明のバイナリーデーターがファイル中に含まれています。このようなファイルにデータ書き込むことは、ハードウェアの接続に関するトラブルシューティングに役立つことがあります。例えば、プリンタデバイス"/dev/lp0" にテキストファイルをダンプしたり、適切なシリアルポート"/dev/ttyS0" にモデムコマンドを送ることができます。しかし、注意深くやらないと、大災害をもたらすことがあります。くれぐれも気をつけて下さい。

注意
普通のプリンターへのアクセスは lp(1) を使います。

以下のように ls(1) を実行するとデバイスノード番号が表示されます。

```
$ ls -l /dev/sda /dev/sr0 /dev/ttyS0 /dev/zero
brw-rw---T  1 root disk     8,  0 Oct 16 20:57 /dev/sda
brw-rw---T+ 1 root cdrom   11,  0 Oct 16 21:53 /dev/sr0
crw-rw---T  1 root dialout  4, 64 Oct 16 20:57 /dev/ttyS0
crw-rw-rw-  1 root root     1,  5 Oct 16 20:57 /dev/zero
```

- "/dev/sda" はメジャーデバイス番号 8 とマイナーデバイス番号 0 を持ちます。これは disk グループに所属するユーザーにより、読出し / 書込みアクセスが可能です。
- "/dev/sr0" はメジャーデバイス番号 11 とマイナーデバイス番号 0 を持ちます。これは cdrom グループに所属するユーザーにより、読出し / 書込みアクセスが可能です。
- "/dev/ttyS0" はメジャーデバイス番号 4 とマイナーデバイス番号 64 を持ちます。これは dialout グループに所属するユーザーにより、読出し / 書込みアクセスが可能です。
- "/dev/zero" はメジャーデバイス番号 1 とマイナーデバイス番号 5 を持ちます。これは誰によっても読出し / 書込みアクセスが可能です。

現代的な Linux システムでは、"/dev/" の下のファイルは udev(7) メカニズムで自動的に充足されます。

1.2.11 特別なデバイスファイル

いくつかの特別なデバイスファイルがあります。

デバイスファイル	アクション	レスポンスの説明
/dev/null	読出し	" 行末 (EOF) 文字" を返す
/dev/null	書込み	何も返さず (底なしのデーターのゴミ捨て場)
/dev/zero	読出し	"\0 (NUL) 文字" を返す (ASCII の数字のゼロとは違う)
/dev/random	読出し	真の乱数発生機から真のエントロピーのあるランダムな文字を返す (遅い)
/dev/urandom	読出し	暗号学的にセキュアーな擬似乱数発生機からランダムな文字を返す
/dev/full	書込み	ディスクフル (ENOSPC) エラーを返す

Table 1.10: スペシャルなデバイスファイルのリスト

以上はシェルのリディレクションとともによく使われます。(項 1.5.8 を参照下さい)。

1.2.12 procfs と sysfs

procfs と sysfs は”/proc” や”/sys” 上にマウントされる仮想ファイルシステムであり、カーネルの内部データー構造をユーザー空間にさらけ出します。言い換えると、オペレーティングシステムのオペレーションへの便利なのぞき窓となると言う意味で仮想といえます。

”/proc” ディレクトリー中には、システム上で実行されている各プロセスに対応したそのプロセス ID (PID) の名前がついたサブディレクトリー他があります。プロセス情報をアクセスする ps(1) のようなシステムユーティリティーはこのディレクトリー構造からその情報を得ています。

”/proc/sys/” の下のディレクトリーには実行時のカーネル変数を変更するインターフェースがあります。(専用の sysctl(8) コマンドもしくはそのプリロード / 設定ファイル”/etc/sysctl.d/*.conf” によっても同様のことができます。)

特にあるファイル - ”/proc/kcore” - に気づくと、パニックになる人がよくいます。これは一般に巨大です。これは (おおよそ) コンピューターのメモリーの内容のコピーです。これは kernel をデバッグするのに用いられます。コンピューターのメモリーを指す仮想ファイルなので、そのサイズに関して心配する必要は全くありません。

”/sys” の下のディレクトリーはカーネルから引き出されたデーター構造、その属性、それらの関連を含んでいます。一部カーネル変数を実行時に変更する機構もまた含まれたりします。

linux-doc-* パッケージで供給される Linux カーネル文書 (”/usr/share/doc/linux-doc-2.6.*/Documentation/ 中の”proc.txt(.gz)” や”sysfs.txt(.gz)” や関連する他の文書を参照下さい。

1.2.13 tmpfs

tmpfs は仮想記憶中にすべてのファイルを保持する一時的なファイルシステムです。メモリー上のページキャッシュ中にある tmpfs のデーターは必要に応じてディスク上の swap 空間へと書き出せます。

”/run” ディレクトリは初期ブートプロセスに tmpfs としてマウントされます。こうすることで”/” が読み取り専用でマウントされていてもそこへの書き込みが可能です。これは過渡的な状態ファイルの保管のための新たな場所で、Filesystem Hierarchy Standard のバージョン 2.3 に規定されたいくつかの場所を置き換えます:

- ”/var/run” → ”/run”
- ”/var/lock” → ”/run/lock”
- ”/dev/shm” → ”/run/shm”

linux-doc-* パッケージで供給される Linux カーネル文書 (”/usr/share/doc/linux-doc-*/Documentation/file 中の”tmpfs.txt(.gz)” を参照下さい。

1.3 ミッドナイトコマンダー (MC)

Midnight Commander (MC) は Linux コンソールや他の端末環境のための GNU 製” スイス軍ナイフ” です。標準 Unix コマンドを習うよりもより簡単なメニューを使ったコンソール経験が初心者にもできます。

”mc” と名づけられた Midnight Commander パッケージを以下のようにしてインストールする必要があります。

```
$ sudo apt-get install mc
```

Debian システムを探検するために mc(1) コマンドを使います。これは学習するための最良の方法です。カーソールキーとエンターキーを使うだけで興味深い場所をちょっと探検します。

- ”/etc” とサブディレクトリー
- ”/var/log” とサブディレクトリー
- ”/usr/share/doc” とサブディレクトリー
- ”/usr/sbin” と”/usr/bin”。

1.3.1 MC のカスタム化

終了時に作業ディレクトリーを MC に変更させそのディレクトリーへ cd させるためには、mc パッケージが提供するスクリプトを”~/.bashrc” が含むように変更します。

```
. /usr/lib/mc/mc.sh
```

この理由は mc(1) (”-P” オプション項目) を参照下さい (今言っていることがよく分からないなら、これは後日しても大丈夫です。)

1.3.2 MC の始動

MC は以下のようにして起動します。

```
$ mc
```

MC を使うとメニューを通じた最小限のユーザーの努力で全てのファイル操作の面倒が見られます。ヘルプ表示を出すには、ただ F1 を押すだけです。カーソールキーとファンクションキーの操作だけで MC を使えます。

注意
gnome-terminal(1) のようなコンソールでは、ファンクションキーのキーストロークがコンソールプログラムに横取りされる事があります。gnome-terminal の場合、”Preferences” → ”General” や”Shortcuts” メニューからこの機能を無効にできます。

もし文字化け表示がされる文字符号化 (エンコーディング) 問題に出会った際には、MC のコマンドラインに”-a” を加えると解消する事があります。

これでも MC の表示の問題が解消しない際には、項 8.3.1 を参照下さい。

1.3.3 MC のファイルマネージャー

2 つのディレクトリーパネルがありそれぞれファイルリストを含むのが標準です。他の便利なモードとしては、右側のウィンドウを”information” とセットしてファイルアクセス権情報などを表示するモードがあります。次にいくつかの不可欠なキーストロークを示します。gpm(8) デーモンを実行すると、Linux の文字ターミナルでマウスも使えます。(MC で普通の挙動のカットアンドペーストをさせるには、shift キーを押して下さい。)

キー	キーバインディング
F1	ヘルプメニュー
F3	内部ファイルビューワー
F4	内部エディター
F9	プルダウンメニュー有効
F10	MC を終了
Tab	二つのウィンドウの間を移動
Insert もしくは Ctrl-T	コピーのような複数ファイル操作のためにファイルをマーク
Del	ファイルの削除 (気をつけましょう -- MC を安全削除モードに設定)
カーソールキー	自明

Table 1.11: MC のキーバインディング

1.3.4 MC のコマンドラインテクニック

- cd コマンドは選択されたスクリーンに表示されたディレクトリーを変更します。
- Ctrl-Enter と Alt-Enter はファイル名をコマンドラインにコピーします。コマンドライン編集と一緒に cp(1) や mv(1) コマンドで御使用下さい。
- Alt-Tab はシェルファイル名の自動展開の選択肢を表示します。
- MC の引数で両ウィンドウのスタートディレクトリーを指定できます。例えば"mc /etc /root"。
- Esc + n-key → Fn (つまり、Esc + 1 → F1、等々、Esc + 0 → F10)
- Esc をキーの前に押すのは Alt をキーと同時に押すのと同様の効果があります。つまり、Esc + c は Alt-C と同じです。Esc はメタキーとよばれ時々"M-" と表記されます。

1.3.5 MC の内部エディター

MC の内部エディターは興味深いカットアンドペースト機構を持ちます。F3 キーを押すと、選択範囲のスタートとマークし、次に F3 を押すと、選択範囲のエンドとマークし、選択範囲を強調します。そしてカーソールを動かすことができます。F6 を押すと、選択範囲はカーソールの位置に移動します。F5 を押すと、選択範囲はコピーされ、カーソールの位置に挿入されます。F2 を押すとファイルを保存します。F10 を押すと選択範囲はなくなります。ほとんどのカーソールキーは直感的に働きます。

このエディターは以下のコマンドの内のひとつを使いファイルに対し直接起動できます。

```
$ mc -e filename_to_edit
```

```
$ mcedit filename_to_edit
```

これはマルチモードエディターではありませんが、複数の Linux コンソール上で使用すると同じ効果を発揮させされます。ウィンドウ間のコピーを行うには、Alt-*n* キーを押して仮想コンソールを切替えて、"File→Insert file" や"File→Copy to file" を用いてファイルの一部を他のファイルに動かします。

この内部エディターはお好きな他の外部エディターと置き換えが可能です。

また、多くのプログラムは使用するエディターを決定するために環境変数"$EDITOR" や"$VISUAL" を使用します。最初 vim(1) や nano(1) が使いにくい場合には"~/.bashrc" に次に示す行を追加してエディターを"mcedit" に設定するのも一計です。

```
export EDITOR=mcedit
export VISUAL=mcedit
```

できればこれは"vim" に設定することを推奨します。

vim(1) が使いにくい場合には、mcedit(1) をほとんどのシステム管理業務のために使い続けられます。

1.3.6 MC の内部ビューワー

MC は非常に賢明なビューワーです。文書内の単語を検索するための素晴らしいツールです。私は"/usr/share/doc" ディレクトリー内のファイルに対していつもこれを使います。これは大量にある Linux 情報を閲覧する最速の方法です。このビューワーは以下のコマンドの内のひとつを使い直接起動できます。

```
$ mc -v path/to/filename_to_view
```

```
$ mcview path/to/filename_to_view
```

ファイルタイプ	**enter** キーへの反応
実行ファイル	コマンド実行
man ファイル	ビューワーソフトに内容をパイプ
html ファイル	ウェッブブラウザーに内容をパイプ
"`*.tar.gz`" や"`*.deb`" ファイル	サブディレクトリーであるかのように内容を表示

Table 1.12: enter キー入力への MC の反応

1.3.7 MC の自動起動機能

ファイルの上で Enter を押すと、適切なプログラムがファイル内容を処理します (項 9.4.11 を参照下さい)。これは非常に便利な MC の機能です。

これらのビューワーや仮想ファイルの機能を有効にするためには、閲覧可能なファイルには実行可能と設定されていてはいけません。chmod(1) コマンドを使うか、MC のファイルメニュー経由で状態を変更して下さい。

1.3.8 MC の仮想ファイルシステム

MC は Internet 経由でのファイルアクセスに使えます。`F9` を押してメニューに行き、"`Enter`" と"h" を押して仮想ファイルシステムをアクティベートします。"`sh://[user@]machine[:options]/[remote-dir]`" の形式で URL を入力すると、`ssh` を利用してあたかもローカルにあるかのようにリモートディレクトリーを取得します。

1.4 基本の Unix 的作業環境

MC はほとんど全てのことを可能にしますが、シェルプロンプトから起動されるコマンドラインツールの使用方法について学び、Unix 的な作業環境に親しむのは非常に重要なことです。

1.4.1 login シェル

login シェルはシステム初期化プログラムによって使われることがあるので、bash(1) のままにし、chsh(1) を使って login シェルを切り替えることを避けることが賢明です。

もし異なるインタラクティブなシェルプロンプトを使いたい場合には、GUI ターミナルエミュレーターのコンフィギュレーションからの設定や、例えば"`exec /usr/bin/zsh -i -l`" や"`exec /usr/bin/fish -i -l`" 等と書き込んだ `~/.bashrc` から起動しましょう。

ティップ
POSIX-ライクなシェルは基本シンタックスはにていますが、シェル変数や glob の展開のような基本事項の挙動が異なることがあります。詳細に関しては個々の文書を確認して下さい。

このチュートリアル章内では、インタラクティブなシェルは常に `bash` です。

1.4.2 Bash のカスタム化

vim(1) の挙動は"`~/.vimrc`" を使ってカスタム化できます。

例えば、次を試してみて下さい。

パッケージ	ポプコン	サイズ	**POSIX** シェル	説明
bash	V:900, I:1000	7309	はい	Bash: GNU Bourne Again SHell (デファクトスタンダード)
bash-completion	V:41, I:964	1952	N/A	bash シェルのプログラム可能なコンプリーション
dash	V:933, I:999	211	はい	Debian の Almquist シェル、シェルスクリプトに好適
zsh	V:38, I:67	2571	はい	Z shell: 多くの拡張された標準シェル
tcsh	V:4, I:14	1366	いいえ	TENEX C Shell: 拡張バージョンの Berkeley csh
mksh	V:5.0, I:7.3	7713	はい	Korn シェルの 1 バージョン
csh	V:1.0, I:5.0	348	いいえ	OpenBSD の C シェル、Berkeley csh の派生
sash	V:0.3, I:4.6	1249	はい	組み込みコマンド付きの独立シェル (標準の"/usr/bin/sh" には不向き)
ksh	V:0.2, I:7.5	65	はい	真の AT&T バージョンの Korn シェル
rc	V:0.08, I:0.69	182	いいえ	AT&T Plan 9 の rc シェルの実装
posh	V:0.01, I:0.21	195	はい	ポリシー準拠の通常のシェル (pdksh の派生)

Table 1.13: シェルプログラムのリスト

```
# enable bash-completion
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

# CD upon exiting MC
. /usr/lib/mc/mc.sh

# set CDPATH to a good one
CDPATH=.:/usr/share/doc:~:~/Desktop:~
export CDPATH

PATH="${PATH+$PATH:}/usr/sbin:/sbin"
# set PATH so it includes user's private bin if it exists
if [ -d ~/bin ] ; then
  PATH="~/bin${PATH+:$PATH}"
fi
export PATH

EDITOR=vim
export EDITOR
```

ティップ
bash に関する更なるカスタム化方法は、第 9 章中の項 9.3.6 等にあります。

ティップ
bash-completion パッケージは bash で入力のプログラムによる補完を可能にします。

1.4.3 特別のキーストローク

Unix 的環境下では、特別の意味を持ったいくつかのキーストロークがあります。普通の Linux の文字ターミナルでは左側の `Ctrl` や `Alt` キーのみが期待にそって機能することを承知下さい。次に特記すべき暗記するべきキーストロークを記します。

キー	キーバインディングの説明
`Ctrl-U`	カーソールの前の 1 行を消去
`Ctrl-H`	カーソールの前の 1 文字を削除
`Ctrl-D`	入力を終了 (シェルを使用中の場合、シェルを終了)
`Ctrl-C`	実行中のプログラムを終了
`Ctrl-Z`	プログラムをバックグラウンドジョブに移動し一時停止
`Ctrl-S`	スクリーンへの出力を停止
`Ctrl-Q`	スクリーンへの出力を再開
`Ctrl-Alt-Del`	システムをリブート / 停止、inittab(5) 参照下さい
`Left-Alt` キー (もしくは、`Windows` キー)	Emacs および同様の UI でのメタキー
`Up-arrow`	`bash` でコマンド履歴検索をスタート
`Ctrl-R`	`bash` でインクリメンタルなコマンド履歴検索をスタート
`Tab`	`bash` のコマンドラインのファイル名入力を完結
`Ctrl-V Tab`	`bash` のコマンドラインで `Tab` を展開することなく入力

Table 1.14: Bash のキーバインディングのリスト

ティップ
ターミナルの `Ctrl-S` 機能は stty(1) で無効にできます。

1.4.4 マウス操作

Debian システム上の文字列へのマウス操作は 2 つのスタイルを混ぜていて少々複雑です:

- 伝統的な Unix 流のマウス操作:
  - 3 ボタンの使用 (クリック)
  - プライマリーを使用
  - `xterm` のような X アプリケーションや Linux コンソール中のテキストアプリケーションが使用
- 現代的な GUI 流のマウス操作:
  - 2 ボタンの使用 (ドラッグ + クリック)
  - プライマリーとクリップボードを使用
  - `gnome-terminal` のような現代的な GUI アプリケーションが使用

ここで、プライマリーの選択はハイライトされたテキスト範囲です。実行中プログラムの停止を避けるため、ターミナルプログラム内では `Shift-Ctrl-C` が代用で使われています。

現代的なホイールマウスの真ん中のホイールは中マウスボタンと見なされ、中クリックに使えます。2 ボタンマウス状況では左右のボタンの同時押しが中クリックとして使えます。

Linux 文字コンソールでマウスを使うには gpm(8) をデーモンで実行する必要があります。

アクション	反応
マウスの左クリックアンドドラッグ	プライマリーの選択範囲の選択
左クリック	プライマリーの選択スタート点の選択
右クリック（伝統的）	プライマリーの選択エンド点の選択
右クリック（現代的）	コンテキスト依存のメニュー (カット/コピー/ペースト)
ミドルクリックまたは Shift-Ins	プライマリーの選択をカーソール位置に挿入
Ctrl-X	プライマリーの選択をクリップボードにカット
Ctrl-C (ターミナルで Shift-Ctrl-C)	プライマリーの選択をクリップボードにコピー
Ctrl-V	クリップボードをカーソール位置に挿入 (ペースト)

Table 1.15: Debian 上でのマウス操作と関連キー操作のリスト

1.4.5 ページャー

less(1) は機能拡張されたページャー (ファイル内容のブラウザー) です。コマンドアーギュメントに指定されたファイル、もしくは標準入力を読みます `less`。コマンドで閲覧中にヘルプが必要なら、"h" を押しましょう。これは、more(1) よりもはるかに高機能で、"`eval $(lesspipe)`" または"`eval $(lessfile)`" をシェルのスタートスクリプト中で実行することで更に機能が拡充されます。詳しくは、"`/usr/share/doc/less/LESSOPEN`" を参照下さい。"`-R`" オプションを用いると、生の文字出力や ANSI カラーエスケープシーケンスが有効になります。less(1) を参照下さい。

ティップ
`less` コマンドでは、ヘルプスクリーンを見るには"h" とタイプし、文字列検索をするには"/" か"?" とタイプし、大文字と小文字の区別・非区別を変更するには"`-i`" とタイプします。

1.4.6 テキストエディター

Unix 的システムで人気のある、Vim か Emacs プログラムのいずれかのバリアントに習熟するべきです。

著者としては Vim コマンドに慣れることは正しいことだと考えています。なぜなら Vi エディターは Linux/Unix の世界では必ず存在するからです。(実際はオリジナルの `vi` か、新しい `nvi` がどこででも見つけられるプログラムです。これにもかかわらず Vim を著者が初心者のために選んだのは、より強力かつ動作が充分似ているのと、`F1` キーを通じてヘルプが表示されるからです。)

これとは違い、Emacs か XEmacs をエディターとして選ぶのも、特にプログラムをするには、非常に良い選択です。Emacs には、ニュースリーダ機能、ディレクトリーの編集機能、メール機能他の、過多な機能があります。プログラミングやシェルスクリプトの編集に使うときは、作業中のフォーマットをインテリジェントに認識し助力をしようとします。Linux 上で必要なプログラムは Emacs だけと考える人もいます。Emacs を今 10 分間学ぶことは将来何時間もの節約になります。Emacs を学ぶ際には GNU の Emacs マニュアルを持っておくことを高く推薦します。

これら全てのプログラムには練習しながら学べるようにチュータリングプログラムが通常同梱されています。Vim を"`vim`" とタイプして起動し、`F1` キーを押します。最初の 35 行を読みます。カーソールを"`|tutor|`" に移動し `Ctrl-]` を押してオンラインの訓練コースを始めます。

注意
Vim や Emacs のような良いエディターは、UTF-8 や他のエギゾチックな符号化方式 (エンコーディング) のテキストを正しく扱えます。それには UTF-8 ロケール中の GUI 環境で、必要なプログラムとフォントをインストールをするのが賢明です。エディタには GUI 環境に依らずにファイルのエンコーディングを設定するオプションがあります。マルチバイトテキストについてはそれぞれの文書を参照下さい。

1.4.7 デフォールトのテキストエディターの設定

Debian にはいくつかの異なったエディターがあります。上述のように vim パッケージをインストールすることを推薦します。

Debian ではシステムのデフォールトのエディターへの統一されたアクセスを”/usr/bin/editor” コマンドを通じて提供しているので、他のプログラム (例えば reportbug(1) 等) がエディターを起動できます。設定変更は次で出来ます。

```
$ sudo update-alternatives --config editor
```

著者が”/usr/bin/vim.tiny” より”/usr/bin/vim.basic” を初心者に推薦するのはシンタクスハイライトをサポートしているからです。

ティップ
多くのプログラムは”$EDITOR” か”$VISUAL” という環境変数を使ってどのエディターを使うかを決めます (項 1.3.5 と項 9.4.11 を参照下さい)。Debian システムの整合性のために、これらを”/usr/bin/editor” と設定しましょう。(歴史的には”$EDITOR” は”ed” で、”$VISUAL” は”vi” でした。)

1.4.8 Vim 利用法

最近の vim(1) は合理的な”nocompatible” モードで起動し NORMAL モードに入ります。[1]

vim をインタラクティブなチュートリアルコースで学習する”vimtutor” プログラムを使いましょう。

vim プログラムはモードに基づきタイプされたキーストロークへの挙動を変えます。バッファーにキーストロークをタイプ入力するのは INSERT モードと REPLACE モード主にされます。カーソール移動は NORMAL モードで主にされます。インタラクティブな選択は VISUAL モードでされます。NORMAL モードで”:” をタイプするとそのモードが Ex モードに代わります。Ex モードはコマンドを受け付けます。

ティップ
Vim には **Netrw** パッケージが同梱されています。Netrw はファイルの読み書きやディレクトリーのネットワーク経由やローカルの閲覧を可能にします！Netrw を”vim .” (アーギュメントとしてピリオド) として試し、”:help netrw” にあるマニュアルを読みましょう。

---

vim の高度な設定は、項 9.2 を参照下さい。

1.4.9 シェル活動の記録

シェルコマンドの出力はスクリーンから押し出されると永久に無くなってしまうかもしれません。シェルでの活動を後で見直せるようにファイルに記録しておくのは良いことです。この種の記録は何らかのシステム管理作業をする際には非常に重要です。

ティップ
新しい Vim (version>=8.2) は TERMINAL-JOB モードを使うときれいにシェル活動を記録できます。項 1.4.8 を参照下さい。

シェル活動の記録の基本方法は script(1) の下で実行することです。

例えば、次を試してみて下さい:

[1] 比較的古い vim ですら起動時に”-N” オプションをつけることで合理的な”nocompatible” モードで起動できます。

モード	キーストローク	アクション
NORMAL	:help|only	ヘルプファイルを表示します
NORMAL	:e filename.ext	filename.ext を編集するために新規バッファーを開きます
NORMAL	:w	現バッファーを元ファイルに上書きします
NORMAL	:w filename.ext	現バッファーを filename.ext に書きます
NORMAL	:q	vim を終了します
NORMAL	:q!	vim を強制終了します
NORMAL	:only	分割し開かれているすべての他ウィンドウを閉じます
NORMAL	:set nocompatible?	vim が合理的な nocompatible モードかを確認します
NORMAL	:set nocompatible	vim が合理的な nocompatible モードに設定します
NORMAL	i	INSERT モードに入ります
NORMAL	R	REPLACE モードに入ります
NORMAL	v	VISUAL モードに入ります
NORMAL	V	行単位の VISUAL モードに入ります
NORMAL	Ctrl-V	ブロック単位の VISUAL モードに入ります
TERMINAL-JOB 以外	ESC-キー	NORMAL モードに入ります
NORMAL	:term	TERMINAL-JOB モードに入ります
TERMINAL-NORMAL	i	TERMINAL-JOB モードに入ります
TERMINAL-JOB	Ctrl-W N (もしくは Ctrl-\ Ctrl-N)	TERMINAL-NORMAL モードに入ります
TERMINAL-JOB	Ctrl-W :	TERMINAL-NORMAL 中の Ex モードに入ります

Table 1.16: 基本の Vim キーストロークのリスト

```
$ script
Script started, file is typescript
```

`script` の下で何なりのシェルコマンドを実行します。

`Ctrl-D` を押して `script` から脱出します。

```
$ vim typescript
```

項 9.1.1 を参照下さい。

1.4.10 基本 Unix コマンド

基本的 Unix コマンドを学びます。ここでは一般的意味で”Unix” を使っています。いかなる Unix クローンの OS も通常等価なコマンドを提供します。Debian システムも例外ではありません。今一部コマンドが思うように機能しなくても心配しないで下さい。エイリアスがシェルで使われた場合は、対応するコマンドの出力は変わります。次は順番に実行すると言う意味の例ではありません。

非特権ユーザーのアカウントから以下のコマンドを全て実行します。

注意
Unix は”`.`” で始まるファイル名を隠す伝統があります。それらは伝統的には特定の設定情報やユーザーの嗜好を含むファイルです。
`cd` コマンドに関しては bash-builtins(7) を参照下さい。
最小限の Debian システムではデフォルトのページャーが more(1) で、スクロールバックができません。`less` パッケージを”`apt-get install less`” というコマンドラインでインストールすると、less(1) がデフォルトのページャーになりカーソールキーでスクロールバック出来るようになります。
上記の”`ps aux | grep -e "[e]xim4*"`” コマンド中に現れる正規表現中の”`[`” と”`]`” によって `grep` が自分自身にマッチするのを避けることができます。正規表現中の”`4*`” は数字”4” の 0 回以上の繰り返しを意味するので、`grep` が”`exim`” と”`exim4`” の両方にマッチします。”`*`” はシェルのファイルネームのグロブと正規表現で使われますが、意味が違います。grep(1) から正規表現を学びましょう。

上記のコマンドを訓練として用いて、ディレクトリーを渡り歩き、システムの中を覗き込んで下さい。コンソールのコマンドに関して質問がある場合は、必ずマニュアルページを読んでみて下さい。

例えば、次を試してみて下さい:

```
$ man man
$ man bash
$ man builtins
$ man grep
$ man ls
```

マンページのスタイルは慣れるのに少々大変かもしれません。なぜなら特に比較的旧式の非常に伝統的なマンページは比較的言葉が少ないからです。しかし一旦慣れるとその簡潔さの良さが分かるでしょう。

GNU や BSD 由来を含む多くの Unix 的なコマンドは以下のように (場合によっては一切の引数無しで) 起動すると簡単なヘルプ情報を表示することを承知下さい。

```
$ commandname --help
$ commandname -h
```

コマンド	説明
pwd	カレント / ワーキングディレクトリーの名前を表示
whoami	現在のユーザー名を表示
id	現在のユーザーのアイデンティティ (名前と uid と gid と関連する group) を表示
file *foo*	"*foo*" ファイルのファイルタイプを表示
type -p *commandname*	"*commandname*" コマンドのファイルの位置を表示
which *commandname*	, ,
type *commandname*	"*commandname*" コマンドに関する情報を表示
apropos *key-word*	"*key-word*" に関連したコマンドを発見
man -k *key-word*	, ,
whatis *commandname*	"*commandname*" コマンドに関する 1 行の説明を表示
man -a *commandname*	"*commandname*" コマンドに関する説明を表示 (Unix スタイル)
info *commandname*	"*commandname*" コマンドに関する比較的長い説明を表示 (GNU スタイル)
ls	ディレクトリーの内容をリスト (非ドットファイルおよびディレクトリー)
ls -a	ディレクトリーの内容をリスト (全ファイルおよびディレクトリー)
ls -A	ディレクトリーの内容をリスト (ほとんど全ファイルおよびディレクトリー、".." と"." をスキップ)
ls -la	ディレクトリーの内容を詳細情報とともにリスト
ls -lai	ディレクトリーの内容を inode 番号と詳細情報とともにリスト
ls -d	現ディレクトリーの中の全ディレクトリーをリスト
tree	ファイルツリーの内容を表示
lsof *foo*	"*foo*" ファイルのオープンの状態をリスト
lsof -p *pid*	プロセス ID: "*pid*" によってオープンされたファイルをリスト
mkdir *foo*	現ディレクトリー中に"*foo*" という新規ディレクトリー作成
rmdir *foo*	現ディレクトリー中の"*foo*" というディレクトリーを削除
cd *foo*	現ディレクトリー中もしくは"$CDPATH" 変数中にリストされたディレクトリー中の"*foo*" というディレクトリーにディレクトリーを変更
cd /	ディレクトリーをルートディレクトリーに変更
cd	現在のユーザーのホームディレクトリーにディレクトリーを変更
cd */foo*	絶対ディレクトリーパス"*/foo*" にディレクトリーを変更
cd ..	親ディレクトリーにディレクトリーを変更
cd ~*foo*	ユーザー"*foo*" のホームディレクトリーにディレクトリーを変更
cd -	一つ前のディレクトリーにディレクトリーを変更
</etc/motd pager	"/etc/motd" の内容をデフォールトのページャーで表示
touch *junkfile*	空ファイル"*junkfile*" を作成
cp *foo bar*	既存のファイル"*foo*" を新規ファイル"*bar*" にコピー
rm *junkfile*	ファイル"*junkfile*" を削除
mv *foo bar*	既存のファイル"*foo*" の名前を新しい名前"*bar*" に変更 (ディレクトリー"*bar*" が存在不可)
mv *foo bar*	既存のファイル"*foo*" を新しい場所"*bar/foo*" に移動 (ディレクトリー"*bar*" が存在しなければいけない)
mv *foo bar/baz*	既存のファイル"*foo*" を新しい場所の新しい名前のファイル"*bar/baz*" に移動 (ディレクトリー"*bar*" が存在しなければいけないが、ディレクトリー"*bar/baz*" は存在してはいけない)
chmod 600 *foo*	既存のファイル"*foo*" を第三者から読出し不可かつ書込み不可 (全員実行不可)
chmod 644 *foo*	既存のファイル"*foo*" を第三者からは読出し可だが書込み不可 (全員実行不可)
chmod 755 *foo*	既存のファイル"*foo*" を第三者からは読出し可だが書込み不可 (全員実行可能)
find . -name *pattern*	シェルで"*pattern*" にマッチするファイル名を探索 (比較的遅い)
locate -d . *pattern*	シェルで"*pattern*" にマッチするファイル名を探索 (定期的に生成されるデーターベースを使い比較的早い)
grep -e "*pattern*" *.html	現ディレクトリーにある".html" で終わる全ファイルから"*pattern*" のパターンを検索し、該当する全ファイルを表示
	フルスクリーンを用いてプロセス情報を表示し、"q" と押して終

1.5 シェルプロンプト

Debian システムの使い方が少し分かったでしょう。Debian システム上でのコマンド実行のメカニズムを掘り下げます。初心者のためにちょっと簡略化してみました。正確な説明は bash(1) を参照下さい。

シンプルなコマンドは、以下の要素のシーケンスとなります。

1. 変数代入 (任意)
2. コマンド名
3. 引数 (任意)
4. リダイレクト (任意: > と >> と < と << 等。)
5. 制御演算子 (任意: && と || と改行と ; と & と (と))

1.5.1 コマンド実行と環境変数

環境変数の値は Unix コマンドの挙動を変えます。

環境変数のデフォールト値は PAM システムにより初期設定されます。その後以下のような何らかのアプリケーションプログラムにより再設定されているかもしれません。

- pam_env のような PAM システムは "/etc/pam.conf" や"/etc/environment" や"/etc/default/locale" によって環境変数を設定できます。
- ユーザーモード systemd ユニット (systemd.unit(5) 参照) から起動される現代的な GUI 環境は"~/.config/environment.d/" ディレクトリー中のファイルによりその環境変数を設定します。
- ユーザー特定のプログラム初期化は"~/.profile" や"~/.bash_profile" や"~/.bashrc" により環境変数を再設定することがあります。

1.5.2 "$LANG" 変数

デフォルトのロケールは"$LANG" 環境変数中に定義され、インストーラーか、GNOME の場合なら"Settings" → "Region & Language" → "Language" / "Formats" と言ったその後の GUI 設定によって"LANG=xx_YY.UTF-8" 等と設定されます。

> 注意
> 本当に必要な場合でなければ、"$LC_*" 変数を避け、とりあえず"$LANG" 変数のみを用いてシステム環境を設定する事をお薦めします。

"$LANG" 変数に与えられる完全なロケール値は 3 つの部分からなります: "xx_YY.ZZZZ"。

ロケールの値	意味
xx	ISO 639 言語コード (小文字)、例えば"en"
YY	ISO 3166 国コード (大文字)、例えば"US"
ZZZZ	コードセット、常に"UTF-8" と設定

Table 1.18: ロケールの値の 3 つの部分

典型的なコマンドの実行は以下のようなシェルの行シーケンスを用います。

推奨ロケール	言語 (地域)
en_US.UTF-8	英語 (米国)
en_GB.UTF-8	英語 (英国)
fr_FR.UTF-8	フランス語 (フランス)
de_DE.UTF-8	ドイツ語 (ドイツ)
it_IT.UTF-8	イタリア語 (イタリア)
es_ES.UTF-8	スペイン語 (スペイン)
ca_ES.UTF-8	カタラン語 (スペイン)
sv_SE.UTF-8	スウェーデン語 (スウェーデン)
pt_BR.UTF-8	ポルトガル語 (ブラジル)
ru_RU.UTF-8	ロシア語 (ロシア)
zh_CN.UTF-8	中国語 (中華人民共和国)
zh_TW.UTF-8	中国語 (台湾 R.O.C.)
ja_JP.UTF-8	日本語 (日本)
ko_KR.UTF-8	韓国語 (大韓民国)
vi_VN.UTF-8	ベトナム語 (ベトナム)

Table 1.19: 推奨ロケールのリスト

```
$ echo $LANG
en_US.UTF-8
$ date -u
Wed 19 May 2021 03:18:43 PM UTC
$ LANG=fr_FR.UTF-8 date -u
mer. 19 mai 2021 15:19:02 UTC
```

以上で、date(1) プログラムは異なるロケール変数で実行されます。

- 最初のコマンドでは、"$LANG" はシステムのデフォルトのロケール値"en_US.UTF-8" に設定されます。
- 二番目のコマンドでは、"$LANG" はフランス語の UTF-8 ロケール値"fr_FR.UTF-8" に設定されます。

ほとんどのコマンド実行は通常頭に環境変数定義をつけません。上記の例の代わりに以下のように実行します。

```
$ LANG=fr_FR.UTF-8
$ date -u
mer. 19 mai 2021 15:19:24 UTC
```

ティップ
バグを報告する場合、もし非英語環境を使っているならば、プログラムを"en_US.UTF-8" ロケールの下で実行し確認することが望ましいです。

ロケールの詳細に関しては、項 8.1 を参照下さい。

1.5.3 "$PATH" 変数

シェルにコマンドを打ち込んだ際に、シェルは"$PATH" 環境変数にリストされたディレクトリーのリストから検索します。"$PATH" 環境変数の値は、シェルの検索パスとも呼ばれます。

標準の Debian インストールでは、ユーザーアカウントの"$PATH" 環境変数には"/usr/sbin" や"/usr/sbin" が含まれないかもしれません。例えば、ifconfig コマンドは"/usr/sbin/ifconfig" とフルパスを使って実行する必要があります。(類似の ip コマンドは"/usr/bin" にあります。)

Bash シェルの"$PATH" 環境変数は、"~/.bash_profile" か"~/.bashrc" ファイルで変更できます。

1.5.4 ”$HOME” 変数

多くのコマンドはユーザー特定の設定をホームディレクトリーに保存し、その内容でコマンドの挙動が変わります。ホームディレクトリーは”$HOME” 環境変数で指定されます。

”$HOME” の値	プログラム実行状況
/	init プロセスが実行するプログラム (デーモン)
/root	普通の root シェルから実行されるプログラム
/home/*normal_user*	普通のユーザーシェルから実行されるプログラム
/home/*normal_user*	普通のユーザーの GUI デスクトップメニューから実行されるプログラム
/home/*normal_user*	”sudo program” を用いて root として実行されるプログラム
/root	”sudo -H program” を用いて root として実行されるプログラム

Table 1.20: ”$HOME” の値のリスト

> ティップ
> シェルは、”~/” を現ユーザーのホームディレクトリーである”$HOME/” へと展開します。シェルは、”~foo/” をユーザー foo のホームディレクトリーである”/home/foo/” へと展開します。

もし $HOME をあなたのプログラムから使えない場合には、項 12.1.5 を参照下さい。

1.5.5 コマンドラインオプション

プログラムコマンドによっては引数があります。引数は”-” か”--” で始まり、オプションと呼ばれ、コマンドの挙動をコントロールします。

```
$ date
Thu 20 May 2021 01:08:08 AM JST
$ date -R
Thu, 20 May 2021 01:08:12 +0900
```

上記で、コマンドライン引数”-R” が date(1) の挙動を RFC2822 準拠の日付文字列出力と変えています。

1.5.6 シェルグロブ

ファイル名を全てタイプせずにファイルのグループをコマンド処理したいことがよくあります。シェルのグロブ (ワイルドカードとも時々呼ばれる) を用いたファイル名のパターン展開を用いるとこのニーズに答えられます。

シェルグロブパターン	マッチルールの説明
*	”.” で始まらないファイル名
.*	”.” で始まるファイル名 (Bash 5.2 以降のデフォルトでは、”.” と”..” は除外)
?	1 文字
[…]	括弧中の 1 文字
[a-z]	”a” と”z” の範囲間の 1 文字
[!…]	括弧内 (”!” 以外) に含まれる文字以外の 1 文字

Table 1.21: シェルグロブパターン

注意
ファイルパス中の”/” は上記テーブル中のシェルグロブパターンではマッチされません。

例えば、次を試してみて下さい:

```
$ mkdir foo; cd foo; touch b1.txt b2.txt b3.c b4.h .b5.txt ..b6.txt
$ mkdir bar; cd bar; touch b1.txt b2.txt b3.c b4.h .b5.txt ..b6.txt ; cd -
$ echo b*.txt
b1.txt b2.txt
$ echo b*
b1.txt b2.txt b3.c b4.h bar
$ echo */*.txt
bar/b1.txt bar/b2.txt
$ echo *.[hc]
b3.c b4.h
$ echo .*
.b5.txt ..b6.txt
$ echo b[!1-3]*
b4.h bar
$ echo b*/b[!1-3]*
bar/b4.h
$ cd ..; rm -rf junk
```

glob(7) を参照下さい。

注意
普通のシェルのファイル名の展開と違い、find(1) が”-name” テスト他でシェルパターン”*” をテストする際にはファイル名先頭の”.” ともマッチします。(新 POSIX 機能)

注意
BASH は shopt 組み込みオプションで”dotglob” や”noglob” や”nocaseglob” や”nullglob” や”extglob” などとすることでグロブ挙動を色々変更できます。bash(1) を参照下さい。

1.5.7 コマンドの戻り値

各コマンドは終了ステータスを戻り値 (変数: ”$?”) として返します。

コマンドの終了状態	戻り値の数値	戻り値の論理値
成功	ゼロ、0	真
失敗	非ゼロ、-1	偽

Table 1.22: コマンドの終了コード

例えば、次を試してみて下さい。

```
$ [ 1 = 1 ] ; echo $?
0
$ [ 1 = 2 ] ; echo $?
1
```

注意
シェルの論理的な観点では、成功は、0 (ゼロ) の値を持つ論理的真として扱われることを承知下さい。少々これは非直感的なのでここで再確認する必要があります。

1.5.8 典型的なコマンドシーケンスとシェルリディレクション

次に挙げるシェルコマンドの一部として一行でタイプするシェルコマンドの慣用句を覚えましょう。

コマンドの慣用句	説明
`command &`	command をサブシェル中でバックグラウンド実行
`command1 | command2`	command1 の標準出力を command2 の標準入力にパイプ (同時並行で実行)
`command1 2>&1 | command2`	command1 の標準出力と標準エラー出力を command2 の標準入力にパイプ (同時進行で実行)
`command1 ; command2`	command1 を実行し、後に続いて command2 を実行
`command1 && command2`	command1 を実行; もし成功したら、後に続いて command2 を実行 (command1 と command2 の両方が成功したら、正常終了を返す)
`command1 || command2`	command1 を実行; もし成功しなかったら、後に続いて command2 を実行 (command1 か command2 のどちらかが成功したら、正常終了を返す)
`command > foo`	command の標準出力を foo ファイルにリダイレクト (上書き)
`command 2> foo`	command の標準エラー出力をファイル foo にリダイレクト (上書き)
`command >> foo`	command の標準出力をファイル foo にリダイレクト (追記)
`command 2>> foo`	command の標準エラー出力を foo ファイルにリダイレクト (追記)
`command > foo 2>&1`	command の標準出力と標準エラー出力を foo ファイルにリダイレクト
`command < foo`	command の標準入力を foo ファイルからリダイレクト
`command << delimiter`	command の標準入力を”delimiter” に出会うまでのこれに続く行からリダイレクト (ヒアドキュメント)
`command <<- delimiter`	command の標準入力を”delimiter” に出会うまでのこれに続く行からリダイレクト (ヒアドキュメント、行頭のタブ文字は入力から削除)

Table 1.23: シェルコマンドの慣用句

Debian システムはマルチタスクシステムです。バックグラウンドジョブを使うと単一シェルの下で複数プログラムを実行可能にします。バックグラウンドジョブの管理にはシェル内部組み込みコマンドの `jobs` や `fg` や `bg` や `kill` を使います。bash(1) マンページ中の”SIGNALS” と”JOB CONTROL” セクションや bash-builtins(7) を参照下さい。

例えば、次を試してみて下さい:

```
$ </etc/motd pager
```

```
$ pager </etc/motd
```

```
$ pager /etc/motd
```

```
$ cat /etc/motd | pager
```

4 つ全ての例が全く同じ表示をしますが、最後の例は余計な `cat` コマンドを実行するので理由なくリソースを無駄に遣います。

シェルでは `exec` 組み込みコマンドを任意のファイルディスクリプタとともに使いファイルをオープンすることができます。

```
$ echo Hello >foo
$ exec 3<foo 4>bar  # open files
```

```
$ cat <&3 >&4       # redirect stdin to 3, stdout to 4
$ exec 3<&- 4>&-    # close files
$ cat bar
Hello
```

ファイルデスクリプタの 0-2 は事前定義されています。

デバイス	説明	ファイルデスクリプタ
stdin	標準出力	0
stdout	標準出力	1
stderr	標準エラー出力	2

Table 1.24: 事前定義されたファイルデスクリプタ

1.5.9 コマンドエイリアス

良く使うコマンドにエイリアスを設定できます。

例えば、次を試してみて下さい:

```
$ alias la='ls -la'
```

こうすると、"la" が"ls -la" の短縮形として機能し、全てのファイルを長いリスト形式でリストします。

既存のエイリアスは alias でリストできます (bash(1) の"SHELL BUILTIN COMMANDS" 参照下さい)。

```
$ alias
...
alias la='ls -la'
```

type 内部コマンドを使うと正確なパスやコマンドの正体を識別できます (bash(1) の"SHELL BUILTIN COMMANDS" 下参照下さい)。

例えば、次を試してみて下さい:

```
$ type ls
ls is hashed (/bin/ls)
$ type la
la is aliased to ls -la
$ type echo
echo is a shell builtin
$ type file
file is /usr/bin/file
```

上記で、ls は最近探索されましたが"file" は最近探索されていませので、"ls" は" ハッシュされた" つまりシェルには"ls" コマンドの場所を高速アクセスのために内部記録していると表示されます。

ティップ
項 9.3.6 を参照下さい。

1.6 Unix 的テキスト処理

Unix 的作業環境では、テキスト処理はテキストを標準テキスト処理ツールの連鎖パイプを通す行います。これは決定的な Unix の発明です。

1.6.1 Unix テキストツール

Unix 的システムでしばしば使われる標準テキスト処理ツールがいくつかあります。

- 正規表現を使わないもの:
  - cat(1) はファイルをつなぎ合わせて内容を全て出力します。
  - tac(1) はファイルをつなぎ合わせ逆順で出力します。
  - cut(1) は行の一部を選択し出力します。
  - head(1) はファイルの最初の部分を選択し出力します。
  - tail(1) はファイルの最後の部分を選択し出力します。
  - sort(1) は行を順番に並び替えます。
  - uniq(1) は順番に並べられたファイルから重複行を削除します。
  - tr(1) は文字を変換削除します。
  - diff(1) は 1 行ごとにファイルを比較します。
- 基本正規表現 (**BRE**) をデフォルトで使用するもの:
  - ed(1) は原始的な行エディターです。
  - sed(1) はストリームエディターです。
  - egrep(1) はテキストのパターンマッチをします。
  - vim(1) はスクリーンエディターです。
  - emacs(1) はスクリーンエディターです。(ちょっと拡張された **BRE**)
- 拡張正規表現 (**ERE**) を使用するもの:
  - awk(1) は単純なテキスト処理をします。
  - egrep(1) はテキストのパターンマッチをします。
  - `tcl`(3tcl) は考え得る全てのテキスト処理をします: re_syntax(3)。時々 `tk`(3tk) とともに使用されます。
  - perl(1) は考え得る全てのテキスト処理をします。perlre(1).
  - `pcre2-utils` パッケージの pcre2grep(1) はテキストのパターンマッチを Perl 互換正規表現 (PCRE) パターンを使ってします。
  - `re` モジュールとともに使うことで python3(1) は考え得る全てのテキスト処理ができます。"`/usr/share/doc/pytho` を参照下さい。

もしこれらのコマンドが正確にどう動作するかを確認したいなら、"`man command`" を使って自分で見つけましょう。

注意
ソート順や範囲表現はロケールに依存します。コマンドの従来の挙動を得たい場合は、**UTF-8** がついた普通のロケール（項 8.1 を参照）ではなく、**C** または **C.UTF-8** ロケールを使います。

注意
Perl 正規表現 (perlre(1)) と Perl 互換正規表現 (PCRE) と `re` モジュールで提供される Python 正規表現は **ERE** に多くの共通の拡張をしています。

BRE	ERE	正規表現の説明
\ . [ ] ^ $ *	\ . [ ] ^ $ *	共通のメタ文字
\+ \? \(\) \{ \} |		”\” でエスケープされた、BRE のみで用いるメタ文字
	+ ? ( ) { } |	”\” でエスケープされ無い、ERE のみで用いるメタ文字
c	c	非メタ文字”c” にマッチ
\c	\c	”c” 自身がメタ文字でも”c” という文字そのものとマッチ
.	.	改行を含む全ての文字とマッチ
^	^	文字列の最初
$	$	文字列の最後
\<	\<	単語の先頭
\>	\>	単語の末尾
[abc…]	[abc…]	”abc…” のいずれかの文字にマッチ
[^abc…]	[^abc…]	”abc…” 以外の文字にマッチ
r*	r*	”r” という正規表現の 0 回以上にマッチ
r\+	r+	”r” という正規表現の 1 回以上にマッチ
r\?	r?	”r” という正規表現の 0 回か 1 回にマッチ
r1|r2	r1|r2	”r1” か”r2” という正規表現のいずれかにマッチ
\(r1|r2\)	(r1|r2)	”r1” か”r2” という正規表現のいずれかにマッチし、それを括弧で囲まれた正規表現と見なす

Table 1.25: BRE と ERE のメタ文字

1.6.2 正規表現

正規表現は多くのテキスト処理ツールで使われています。シェルグロブに類似していますがより複雑で強力です。

正規表現はマッチするパターンを表現し、テキスト文字とメタ文字からなっています。

メタ文字は特別な意味を持った文字です。上記のようにテキストツールによって、**BRE** と **ERE** の 2 つの主要なスタイルがあります。

emacs の正規表現は、**ERE** 同様の”+” と”?” をメタ文字と扱う拡張をしてはありますが、基本的に **BRE** です。これら文字を emacs の正規表現で”\” でエスケープする必要はありません。

grep(1) によって正規表現を使った文字列探索ができます。

例えば、次を試してみて下さい:

```
$ egrep 'GNU.*LICENSE|Yoyodyne' /usr/share/common-licenses/GPL
GNU GENERAL PUBLIC LICENSE
GNU GENERAL PUBLIC LICENSE
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
```

ティップ
項 9.3.6 を参照下さい。

1.6.3 置換式

置換式の場合、一部の文字は特別な意味を持ちます。

置換式	置換式を置換する文字の説明
&	正規表現がマッチしたもの (emacs では \& を使用)
\n	n 番目の括弧で囲まれた正規表現にマッチしたもの (”n” は数字)

Table 1.26: 置換式

Perl の代替文字列には”&” に代えて”$&” が使われ、”\n” に代えて”$n” が使われます。

例えば、次を試してみて下さい:

```
$ echo zzz1abc2efg3hij4 | \
sed -e 's/\(1[a-z]*\)[0-9]*\(.*\)$/=&=/'
zzz=1abc2efg3hij4=
$ echo zzz1abc2efg3hij4 | \
sed -E -e 's/(1[a-z]*)[0-9]*(.*)$/=&=/'
zzz=1abc2efg3hij4=
$ echo zzz1abc2efg3hij4 | \
perl -pe 's/(1[a-z]*)[0-9]*(.*)$/=$&=/'
zzz=1abc2efg3hij4=
$ echo zzz1abc2efg3hij4 | \
sed -e 's/\(1[a-z]*\)[0-9]*\(.*\)$/\2===\1/'
zzzefg3hij4===1abc
$ echo zzz1abc2efg3hij4 | \
sed -E -e 's/(1[a-z]*)[0-9]*(.*)$/\2===\1/'
zzzefg3hij4===1abc
$ echo zzz1abc2efg3hij4 | \
perl -pe 's/(1[a-z]*)[0-9]*(.*)$/$2===$1/'
zzzefg3hij4===1abc
```

ここで、括弧で囲まれた正規表現のスタイルと、マッチした文字列が異なるツール上でテキスト置換処理にどう使われるかとに注目下さい。

これらの正規表現は一部エディター内でカーソールの動きやテキスト置換アクションに対しても使えます。

シェルコマンドラインの行末のバックスラッシュ”\” は改行をホワイトスペース文字としてエスケープするので、シェルコマンドライン入力を次行に継続させます。

これらのコマンドを習うために、関連するマニュアルページを全て読んで下さい。

1.6.4 正規表現を使ったグローバル置換

ed(1) コマンドは以下のようにすると”file” 中に存在する全ての”FROM_REGEX” を”TO_TEXT” に置換できます。

```
$ ed file <<EOF
,s/FROM_REGEX/TO_TEXT/g
w
q
EOF
```

sed(1) コマンドは以下のようにすると”file” 中に存在する全ての”FROM_REGEX” を”TO_TEXT” に置換できます。

```
$ sed -i -e 's/FROM_REGEX/TO_TEXT/g' file
```

vim(1) コマンドは ex(1) コマンドを使い以下のようにすると”file” 中に存在する全ての”FROM_REGEX” を”TO_TEXT” に置換できます。

```
$ vim '+%s/FROM_REGEX/TO_TEXT/gc' '+update' '+q' file
```

ティップ
上記の”c” フラグをにより各置換毎に対話型の確認をします。

複数ファイル (”file1” と”file2” と”file3”) を vim(1) や perl(1) で同様に正規表現を用いて処理できます。

```
$ vim '+argdo %s/FROM_REGEX/TO_TEXT/gce|update' '+q' file1 file2 file3
```

ティップ
上記の”e” フラグをにより”No match” エラーでマッピングが停止することを防ぎます。

```
$ perl -i -p -e 's/FROM_REGEX/TO_TEXT/g;' file1 file2 file3
```

perl(1) の例中で、”-i” はその場で各ターゲットファイルの編集、”-p” は与えられたすべてのファイルに関する暗黙的なループを意味します。

ティップ
”-i” の代わりに”-i.bak” という引数を用いるとオリジナルファイル名に”.bak” をつけたファイル名でオリジナルファイルが保管されます。複雑な置換のエラーからの復元が簡単にできます。

注意
ed(1) や vim(1) は **BRE** です。一方、perl(1) は **ERE** です。

1.6.5 テキストファイルからのデーター抽出

2004 年以前の元 Debian リーダの名前と就任日がスペースで分割されたフォーマットでリストされている”DPL” と呼ばれるファイルを考えてみましょう。

```
Ian     Murdock   August  1993
Bruce   Perens    April   1996
Ian     Jackson   January 1998
Wichert Akkerman  January 1999
Ben     Collins   April   2001
Bdale   Garbee    April   2002
Martin  Michlmayr March   2003
```

ティップ
最新の Debian のリーダーの歴史に関しては、”A Brief History of Debian” を参照下さい。

Awk はこういったタイプのファイルからデーターを抽出するために良く使われます。

例えば、次を試してみて下さい:

```
$ awk '{ print $3 }' <DPL                    # month started
August
April
January
January
April
```

```
April
March
$ awk '($1=="Ian") { print }' <DPL          # DPL called Ian
Ian     Murdock   August  1993
Ian     Jackson   January 1998
$ awk '($2=="Perens") { print $3,$4 }' <DPL # When Perens started
April 1996
```

Bash などのシェルもこれらのファイルを解釈するのに使えます。

例えば、次を試してみて下さい:

```
$ while read first last month year; do
    echo $month
  done <DPL
... same output as the first Awk example
```

ここで、read 組込みコマンドは"$IFS" (内部フィールドセパレータ) を用いて行を単語単位で分割します。

"$IFS" を":" に変更すると、"/etc/passwd" をシェルでうまく解読できます。

```
$ oldIFS="$IFS"   # save old value
$ IFS=':'
$ while read user password uid gid rest_of_line; do
    if [ "$user" = "bozo" ]; then
      echo "$user's ID is $uid"
    fi
  done < /etc/passwd
bozo's ID is 1000
$ IFS="$oldIFS"   # restore old value
```

(同じことを Awk を使って行うには、フィールドセパレータ設定は"FS=':'" とします。)

IFS はパラメーター展開、コマンド置換、数式展開の結果を分割するためにもシェルにより使われます。これはダブルクォートやシングルクォートされた単語内では発生しません。IFS の標準値は *space* と *tab* と *newline* の組合せです。

シェルの IFS トリックを注意深く使って下さい。シェルがスクリプトの一部を入力として解釈した場合に、奇妙なことが起きるかもしれません。

```
$ IFS=":,"                          # use ":" and "," as IFS
$ echo IFS=$IFS,   IFS="$IFS"       # echo is a Bash builtin
IFS=  , IFS=:,
$ date -R                           # just a command output
Sat, 23 Aug 2003 08:30:15 +0200
$ echo $(date -R)                   # sub shell --> input to main shell
Sat  23 Aug 2003 08 30 36 +0200
$ unset IFS                         # reset IFS to the default
$ echo $(date -R)
Sat, 23 Aug 2003 08:30:50 +0200
```

1.6.6 コマンドをパイプするためのスクリプト断片

以下のスクリプトはパイプの一部として素晴らしいことをします。

1 行のシェルスクリプトは find(1) や xargs(1) を使って非常に複雑な操作を多くのファイルに繰り返し実行できます。項 10.1.5 と項 9.4.9 を参照下さい。

シェルの対話モードを使うのが複雑過ぎるようになったときには、シェルのスクリプトを書くのも一計です (項 12.1 を参照下さい)。

スクリプト断片 **(1** 行入力**)**	コマンドの効果
`find /usr -print`	"/usr" の下の全ファイル発見
`seq 1 100`	1 から 100 までプリント
`| xargs -n 1 command`	パイプからの各項目を引数としてコマンドを反復実行
`| xargs -n 1 echo`	パイプからのホワイトスペースで分離された項目を行に分割
`| xargs echo`	パイプからの全ての行を 1 行にマージ
`| grep -e regex_pattern`	*regex_pattern* を含む行を抽出
`| grep -v -e regex_pattern`	*regex_pattern* を含まない行を抽出
`| cut -d: -f3 -`	":" で区切られた 3 番目のフィールドを抽出 (passwd ファイルなど)
`| awk '{ print $3 }'`	ホワイトスペースで区切られた 3 番目のフィールドを抽出
`| awk -F'\t' '{ print $3 }'`	タブで区切られた 3 番目のフィールドを抽出
`| col -bx`	バックスペースを削除し、タブをスベースに変換
`| expand -`	タブをスベースに変換
`| sort| uniq`	入力をソートし重複を削除
`| tr 'A-Z' 'a-z'`	大文字を小文字に変換
`| tr -d '\n'`	複数行を 1 行に連結
`| tr -d '\r'`	キャリッジリターンを削除
`| sed 's/^/# /'`	各行頭に"#" を追加
`| sed 's/\.ext//g'`	".ext" を削除
`| sed -n -e 2p`	2 番目の行を表示
`| head -n 2 -`	最初の 2 行を表示
`| tail -n 2 -`	最後の 2 行を表示

Table 1.27: コマンドをパイプするためのスクリプト断片

Chapter 2

Debian パッケージ管理

注意
本章は最新安定版リリースがコード名: `trixie` と言う前提で書かれています。
APT システムのデータソースは本文書中では集合的にソースリストと表記されます。これは、”`/etc/apt/sources.list`” ファイルか、”`/etc/apt/sources.list.d/*.list`” ファイルか、”`/etc/apt/sources.list.d/*.sources`” ファイルの中の何処かに定義されます。

2.1 Debian パッケージ管理の前提条件

2.1.1 Debian パッケージ管理システム

Debian は、フリーソフトウェアーのコンパイル済みバイナリーパッケージからなる整合性あるディストリビューションを作り、そのアーカイブを通じてそれらを頒布するボランティア組織です。

Debian のアーカイブは、HTTP や FTP 法によるアクセスされるための多くのリモートのミラーサイトとして提供されています。それは、CD-ROM/DVD によっても提供されています。

これら全てのリソースを利用できる現行の Debian パッケージ管理システムは Advanced Packaging Tool (APT) です。

Debian のパッケージ管理システムは、適正に使われれば、バイナリーパッケージの整合性ある組み合わせがアーカイブからシステムにインストールされるようになっています。現在、amd64 アーキテクチャーでは 78709 つのパッケージが利用できます。

Debian のパッケージ管理システムは、多彩な歴史があり、使用されるフロントエンドのユーザープログラムやバックエンドのアーカイブへのアクセス方法に多くの選択肢があります。現在は以下を推薦します。

- パッケージのインストールや削除や dist-upgrade を含む全ての対話的コマンドライン操作を提供する、apt(8)。
- スクリプトから Debian のパッケージ管理をするためによぶ、apt-get(8)。(古い Debian システム等で)`apt` が使えない際の控えのオプション。
- インストール済みパッケージを管理したり、使用可能なパッケージを探索するためのインタラクティブなテキストインターフェースを提供する、aptitude(8)

2.1.2 パッケージ設定

Debian システム上でのパッケージ設定の要点を次に記します。

パッケージ	ポプコン	サイズ	説明
dpkg	V:909, I:1000	5086	Debian のための低水準パッケージ管理システム（ファイルベース）
apt	V:882, I:1000	5025	CLI でパッケージを管理する APT フロントエンド: apt/apt-get/apt-cache
aptitude	V:32, I:154	4837	フルスクリーンコンソール中でインタラクティブにパッケージを管理する APT フロントエンド: aptitude(8)
tasksel	V:42, I:985	352	選択されたタスクをインストールする APT フロントエンド: tasksel(8)
unattended-upgrades	V:142, I:190	374	セキュリティー更新の自動インストールを可能にする APT の拡張パッケージ
gnome-software	V:159, I:260	4732	GNOME 用のソフトウェアーセンター (GUI APT フロントエンド)
synaptic	V:31, I:262	7264	グラフィカルなパッケージマネージャー (GTK APT フロントエンド)
apt-utils	V:383, I:996	1187	APT ユーティリティープログラム: apt-extracttemplates(1) と apt-ftparchive(1) と apt-sortpkgs(1)
apt-listchanges	V:451, I:895	563	パッケージ変更履歴の通知ツール
apt-listbugs	V:4.7, I:6.4	512	APT による各インストール前にクリチカルバグをリストする
apt-file	V:14, I:52	76	APT パッケージ探索ユーティリティー -- コマンドラインインターフェース
apt-rdepends	V:0.4, I:4.2	39	パッケージの依存関係を再帰的にリスト

Table 2.1: Debian のパッケージ管理ツールのリスト

- 現代的デスクトップ環境では、パッケージ設定変更やパッケージアップグレード後にシステムをリブートすることが、システムの適正稼働に好ましい。
- システム管理者による手動の設定は尊重されます。言い換えれば、パッケージ設定システムは利便性のために勝手な設定をしません。
- 各パッケージは、debconf(7) と呼ばれる標準化されたユーザーインターフェースを使用するパッケージの初期インストールプロセス支援のためのパッケージ毎の設定スクリプトが同梱されています。
- Debian の開発者はパッケージの設定スクリプトによりユーザーのアップグレードが滞りなく進むように最大限の努力を行います。
- システム管理者にはパッケージされたソフトウェアーの全機能が利用可能です。ただしセキュリティーリスクのある機能はデフォールトのインストール状態では無効にされています。
- セキュリティーリスクのあるサービスを手動でアクティベートした場合は、リスクの封じ込めはあなたの責任です。
- システム管理者は難解奇異な設定を手動で有効にはできます。ただこんなことをすればポピュラーな一般の補助プログラムと干渉してしまうかもしれません。

2.1.3 基本的な注意事項



警告
ランダムな混合のスイーツからパッケージをインストールしてはいけません。コンパイラーの ABI とかライブラリーのバージョンとかインタープリターの機能等のシステム管理に関する深い知見が必要なパッケージの整合性がきっと破壊されます。

初心者の Debian システム管理者は **Debian** の安定版 **`stable`** リリースをセキュリティー更新を適用しながら使うべきです。Debian システムを非常によく理解するまでは、以下の予防策を守るべきです。

- ソースリスト中にテスト版 **`testing`** とか不安定版 **`unstable`** とかを含めない。
- ソースリスト中に標準の Debian と Debian 以外の Ubuntu のようなアーカイブを混在させない。
- "`/etc/apt/preferences`" を作成しない。
- パッケージ管理ツールのデフォールトを影響を理解せずに変更しない。
- ランダムなパッケージを"`dpkg -i` *`random_package`*" でインストールしない。
- ランダムなパッケージを"`dpkg --force-all -i` *`random_package`*" で絶対インストールしない。
- "`/var/lib/dpkg/`" の中のファイルを消去や改変しない。
- ソースから直接コンパイルしたソフトウェアープログラムをインストールする際にシステムファイルを上書きしない。
  - 必要な場合は"`/usr/local/`" か"`/opt/`" 中にインストールする。

上記予防策に違反するアクションにより起越される Debian パッケージシステムへの非互換効果は、システムを使えなくするかもしれません。

ミッションクリティカルなサーバーを走らせる真剣な Debian システム管理者は更なる用心をすべきです。

- 安全な条件下であなたの特定の設定で徹底的にテストすることなくセキュリティー更新をも含めた如何なるパッケージもインストールをしてはいけません。
  - システム管理者のあなたがシステムに対して最終責任があります。
  - Debian システムの長い安定性の歴史それ自体は何の保証でもありません。

2.1.4 永遠のアップグレード人生



注意
あなたの業務サーバーには、セキュリティー更新をした安定版 stable スイーツを推薦します。管理に限られた時間しか割けないデスクトップ PC に関しても同様の事が言えます。

私が上記のような警告をしても、多くの本文書の読者は、テスト版 testing や不安定版 unstable スイーツを使いたいと考えるのは分かっています。

以下に記すことにより悟りを開けば、アップグレード地獄という果てしない因果応報の葛藤から人は解脱し、Debian の涅槃の境地に到達できます。

本リストは自己管理されたデスクトップ環境を対象とします。

- Debian continuous integration と source only upload practices と library transition tracking 等の Debian アーカイブの QA インフラで自動管理された実質的にローリングリリースだから、testing スイートを使いましょう。testing スイートのパッケージは全ての最新機能を提供するのに十分頻繁に更新されます。
- テスト版 testing スイーツに該当するコードネーム (trixie が安定版 stable であるリリース期間の場合”forky”) をソースリスト中に設定します。
- メジャースイートリリースの約一ヶ月後に自分自身で状況を確認した後でソースリストの中のこのコードネームを新しいコードネームに手動で更新します。Debian user と developer のメーリングリストもこれに関する良好な情報源です。

非安定版 unstable スイーツを使うことは推奨できません。非安定版 unstable スイーツは開発者としてパッケージのデバグには好適ですが、普通のデスクトップ使用ではあなたを不要なリスクに晒してしまいます。Debian システムの非安定版 unstable スイーツは、ほぼいつも非常に安定に見えるとはいえ、過去パッケージ上の問題をいくつか経験して来てるし、その一部は簡単には解決できないものでした。

Debian パッケージのバグからの早急かつ簡単な復元を確実にするいくつかの予防策のアイデアです。

- 項 3.2 に従い、レスキューシステムを準備する。
- Debian システムの安定版 stable スイーツを別のパーティションにインストールし、システムをデュアルブータブル化
- apt-listbugs をインストールしてアップグレードの前に Debian バグトラッキングシステム (BTS) をチェックを考慮
- 問題回避するのに十分なだけのパッケージシステムの基盤を学習



注意
これらの予防策の何れもできないなら、テスト版 testing や不安定版 unstable スイーツを使うのには、きっとあなたは準備不足です。

2.1.5 Debian アーカイブの基本

ティップ
Debian アーカイブの正式のポリシーは Debian ポリシーマニュアル、第 2 章 - Debian アーカイブに規定されています。

Debian アーカイブをシステムユーザーの視点から見てみます。

システムユーザーから見ると、Debian アーカイブは APT システムを用いてアクセスされます。

APT システムは、そのデーターソースをソースリストとして指定し、それは sources.list(5) に説明されています。

典型的な HTTP でのアクセスを使う trixie システムでは、ソースリスト” が”/etc/apt/sources.list.d/debian.so
中に deb822 スタイルで以下で提供されます:

```
Types: deb deb-src
URIs: http://deb.debian.org/debian/
Suites: trixie trixie-updates
Components: main non-free-firmware contrib non-free

Types: deb deb-src
URIs: http://security.debian.org/debian-security/
Suites: trixie-security
Components: main non-free-firmware contrib non-free
```

ティップ
もしソースリストが”/etc/apt/sources.list” とか”/etc/apt/sources.list.d/*.list” という 1 行スタイルのファイルで提供される場合は、以下のようにして更新します。

```
$ sudo apt modernize-sources
```

---

deb822 スタイルのソースリストの要点は以下です。

- その定義ファイルは”/etc/apt/sources.list.d/*.sources” ファイルです。
- 空行で分離されている複数行の各ブロックは APT システムのデーターソースを定義します。
- ”Types:” スタンザは”deb” や”deb-src” といったタイプのリストを定義します。
- The ”URIs:” スタンザは Debian アーカイブのルート URI のリストを定義します。
- ”Suites:” スタンザはスイート名かコードネームのいずれかを用いてディストリビューションのリストを定義します。
- ”Components:” スタンザは Debian アーカイブの中の有効なアーカイブのエリア名のリストを定義します。

ソース関連のメタデーターにアクセスしない aptitude のためだけなら”deb-src” 行は安全に省略することができます。こうするとアーカイブのメタデーターの更新速度が向上します。

URL は”https://”, ”http://”, ”ftp://”, ”file://”, …のいずれも可能です。

”#” で始まる行はコメントで無視されます。

上記で、次期安定版 stable がリリースされて驚かされ無いように、私はスイート名の”stable” や”testing” でなくコード名の”trixie” や”forky” を使います。

ティップ
もし上記の例で”trixie” ではなく”sid” が使われる場合には、”deb: http://security.debian.org/ …” 行やその deb822 相当の内容はソースリストには不要です。これは”sid” (不安定版 unstable) には、セキュリティー更新のアーカイブが存在しないからです。

次は、trixie リリース後の設定ファイル中に用いられる Debian アーカイブサイトの URL とスイーツ名もしくはコード名です。

アーカイブの **URL**	スイート名	コードネーム	レポジトリーの用途
http://deb.debian.org/debian/	`stable`	`trixie`	徹底的な確認後の擬似静的 `stable` (安定版) リリース
http://deb.debian.org/debian/	`stable-updates`	`trixie-updates`	スイーツ用のセキュリティーアップデート (重要)
http://security.debian.org/debian-security/	`stable-security`	`trixie-security`	スイーツ用のセキュリティーアップデート (重要)
http://deb.debian.org/debian/	`testing`	`forky`	適度の確認と短い待機後の動的 `testing` (テスト版) リリース
http://deb.debian.org/debian/	`unstable`	`sid`	最小限の確認と無待機後の動的 `unstable` (不安定版) リリース
http://deb.debian.org/debian/	`experimental`	N/A	開発者によるプリリリース実験 (任意、開発者専用)
http://deb.debian.org/debian/	`stable-proposed-updates`	`trixie-proposed-updates`	次期 `stable` (安定版) ポイントリリース用の更新 (任意)
http://deb.debian.org/debian/	`stable-backports`	`trixie-backports`	主に `testing` リリースから再コンパイルされたパッケージのランダムな集合
http://security.debian.org/debian-security/	`testing-security`	`forky-security`	セキュリティーチームによるサポートは無く利用されてません

Table 2.2: Debian アーカイブサイトのリスト



注意
セキュリティー更新された純粋な安定版 **`stable`** リリースのみが最善の安定性を提供します。一部 **`testing`** や **`unstable`** 由来のパッケージを混用してほとんど **`stable`** リリースを実行することは、純粋な **`unstable`** リリースを実行するよりリスクがあります。**`stable`** リリースの下で最新バージョンのいくつかのプログラムが本当に必要なら、stable-updates や backports (項 2.7.4 を参照下さい) サービスからのパッケージを使って下さい。これらのサービスは細心の注意を持って使う必要があります。



注意
基本的に、`stable` か `testing` か `unstable` のスイーツの内の 1 つだけを”deb” 行に書くべきです。もし、`stable` と `testing` と `unstable` のスイーツの何らかの組み合わせを”deb” 行に書けば、APT プログラムは、最新のアーカイブのみが有効であるにもかかわらず、実行速度が低下します。”`/etc/apt/preferences`” ファイルがはっきりとした目的を持って使われている場合 (項 2.7.7) のみ複数のリストに意味があります。

ティップ
`stable` スイーツの Debian システムは、上記の例のようにソースリスト中にセキュリティー更新を有効とする”`http://security.debian.org/`” を含む内容を織り込むことが望ましいです。

注意
`stable` アーカイブのセキュリティーバグは Debian のセキュリティーチームにより修正されます。本活動は非常に厳格で信頼できるものです。`testing` アーカイブのセキュリティーバグは Debian の testing セキュリティーチームにより修正されます。諸所の事情で、本活動は `stable` ほどは厳格ではなく、修正された `unstable` パッケージの `testing` アーカイブへの移行を待つ必要があるかもしれません。`unstable` アーカイブのセキュリティーバグは個別のメンテナにより修正されます。活発にメンテされている `unstable` パッケージはアップストリームのセキュリティー修正を使うことで通常比較的良い状態です。Debian がセキュリティーバグへ如何に対応するかに関しては Debian security FAQ を参照下さい。

エリア	パッケージ数	パッケージ構成要素のクライテリア
`main`	77218	DFSG に完全準拠し、`non-free` のパッケージに非依存 (main = 主要)
`non-free-firmware`	85	DFSG 非コンプライアント、合理的なシステムインストール経験のために必要なファームウエアー
`contrib`	365	DFSG に完全準拠だが、`non-free` のパッケージに依存有り (contrib = 寄与)
`non-free`	1041	DFSG に非準拠で `non-free-firmware` に含まれない

Table 2.3: Debian アーカイブエリアのリスト

ここで、上記にあるパッケージ数は amd64 アーキテクチャーに関する数字です。`main` エリアのアーカイブのみが Debian システムです (項 2.1.6 を参照下さい)。

Debian アーカイブの構成は、各アーカイブの URL の後ろに `dists` か `pool` をつけた URL にブラウザーを向ければ学習できます。

ディストリビューションは、スイーツとコード名の 2 つの方法で言及されます。この他にディストリビューションと言う言葉は多くの文書でスイーツの同義語としても使われています。スイーツとコード名の関係は以下のようにまとめられます。

タイミング	スイーツ = 安定版 **`stable`**	スイーツ = テスト版 **`testing`**	スイーツ = 不安定版 **`unstable`**
`trixie` リリース後	コード名 = `trixie`	コード名 = `forky`	コード名 = `sid`
`forky` リリース後	コード名 = `forky`	コード名 = `duke`	コード名 = `sid`

Table 2.4: スイーツとコード名の関係

コード名の歴史は、Debian FAQ: 6.2.1 Which other codenames have been used in the past? に記載されています。

比較的厳格な Debian アーカイブの用語法では、” セクション” という言葉はアプリケーションの分野によるパッケージ分類に特化して使われます。(しかし、”main セクション” という言葉は main エリアを提供する Debian アーカイブ部分を表現するのにしばしば使われています。)

Debian デベロッパー (DD) が不安定版 `unstable` アーカイブに新たなアップロードを (incoming での処理を経由して) する度毎に、アップロードするパッケージが最新の不安定版 `unstable` アーカイブの最新のパッケージ集合と互換とする義務が DD にはあります。

重要なライブラリーのアップグレード他の理由で DD がこのコンパチビリティーを壊す際には、debian-devel のメーリングリスト他に通常アナウンスがされます。

Debian のアーカイブ管理スクリプトによって非安定版 `unstable` アーカイブからテスト版 `testing` アーカイブへパッケージ集合が移動される前に、アーカイブ管理スクリプトはパッケージの成熟度 (約 2-10 日経過) と RC バグレポート状況を確認するばかりでなく、テスト版 `testing` アーカイブの最新パッケージ集合との互換となるよう努めます。このプロセスがあるので、テスト版 `testing` アーカイブは非常に新しくかつ使いやすいのです。

リリースチームによる徐々のアーカイブ凍結過程を通じて、少々の手動の介入を伴いつつテスト版 `testing` アーカイブは完全に整合性をもったバグの無い状態へと徐々に熟成されます。そして、古いテスト版 `testing` アーカイブのコード名を新たな安定版 `stable` アーカイブへと割り当て、新たなコード名を新たなテスト版 `testing` アーカイブへと割り当てることで、新たな安定版 `stable` がリリースされます。新たなテスト版 `testing` アーカイブの当初の内容は、新たにリリースされた安定版 `stable` アーカイブとまったく同じです。

不安定版 `unstable` もテスト版 `testing` アーカイブもともにいくつかの要因で一時的に細かな問題発生があるかもしれません。

- ブロークンなパッケージのアーカイブへのアップロード (主に `unstable` にて)
- 新規パッケージをアーカイブに受け入れる際の遅延 (主に `unstable` にて)
- アーカイブの同期のタイミング問題 (`testing` と `unstable` の両方にて)。

- パッケージの除去などのアーカイブへの手動の介入 (どちらかといえば testing にて)、等。

もしこれらのアーカイブを使おうと考えるなら、この種の細かな問題の修復や回避は必須技能です。



注意
非安定版 unstable やテスト版 testing アーカイブを通常使っていようとも、ほとんどのデスクトップユーザーは新たな安定版 stable リリースの後約数ヶ月はセキュリティー更新された安定版 stable アーカイブを使うべきです。この移行期は、非安定版 unstable もテスト版 testing アーカイブの何れももほとんどの人に良いものではありません。非安定版 unstable アーカイブを使おうとすると、核となるパッケージが大アップグレードの嵐に見舞われるので、あなたのシステムをうまく使える状態に保つのは困難です。テスト版 testing アーカイブを使おうとしても、安定版 stable アーカイブとほとんど同じ内容でセキュリティーサポートはありません (Debian testing-security-announce 2008-12)。1 ヶ月ほど経てば、注意を払えば非安定版 unstable やテスト版 testing アーカイブも使い物になるかもしれません。

ティップ
テスト版 testing アーカイブを追跡している際には、除去されたパッケージによって引き起こされる問題は該当するバグ修正のためにアップロードされたパッケージを非安定版 unstable アーカイブからインストールすれば通常回避できます。

アーカイブの定義は、Debian ポリシーマニュアルを参照下さい。

- ” セクション”
- ” 優先度 (priorities)”
- ” ベースシステム”
- ” 必須パッケージ”

2.1.6 Debian は 100% フリーソフトウェアーです

Debian は以下の理由で 100% フリーソフトウェアーです:

- Debian はユーザーの自由を尊重すべくデフォルトではフリーソフトウェアのみをインストールします。
- Debian は main 中にはフリーソフトウェアーのみを提供します。
- Debian は main からのフリーソフトウェアーのみを実行することを推奨します。
- main 中のいかなるパッケージも non-free や non-free-firmware や contrib 中のいずれのパッケージに依存しないし、これらを推薦することもありません。

一部の人は以下の 2 つの事実が矛盾するのでは無いかとの疑問を持ちます。

- 「Debian は 100% フリーソフトウェアーであり続けます」。Debian 社会契約の第一項)
- Debian サーバーは non-free-firmware や non-free や contrib パッケージをホストします。

これらは以下の理由で矛盾しません。

- Debian システムは 100% フリーソフトウェアーでそのパッケージは Debian サーバーの main エリア中にホストされます。

- Debian システム外のパッケージは Debian サーバーの non-free と non-free-firmware と contrib エリア中にホストされます。

これらは Debian 社会契約の第 4 項と第 5 項中に正確に説明されています:

- 私たちはユーザーとフリーソフトウェアーを大切にします
  - 私たちはユーザーとフリーソフトウェアーコミュニティーからの要求に従います。彼らの関心を最優先に考えます。私たちはさまざまな状況におけるコンピューター利用環境の運用に関して、ユーザーの必要を満たすように行動します。私たちは Debian システム上での利用を目的としたフリーではない著作物に敵対することはありません。またそのような著作物を作成または利用する人々に対して、料金を徴収することはありません。私たちは、Debian システムとその他の著作物の両方を含むディストリビューションを、第三者が作成することも認めています。その際、私たちは料金を徴収しません。私たちはこれらの目標を増進させるために、これらのシステムの使用を妨げるような法的な制約のない、高品質な素材を統合したシステムを提供します。
- 私たちのフリーソフトウェアー基準に合致しない著作物について
  - 私たちは、Debian フリーソフトウェアーガイドラインに適合していない著作物を使わなければならないユーザーがいることを認めています。このような著作物のために、私たちはアーカイブに「non-free」と「non-free-firmware」と「contrib」という領域を作りました。これらの領域にあるパッケージは、Debian 上で使用できるよう設定されていますが、Debian システムの一部ではありません。私たちは、CD 製造業者がこれらの領域にあるパッケージを彼らの CD に収録して配布できるかどうか判断する際に、それぞれのパッケージのライセンスを読んで決めるよう奨めています。このように、フリーではない著作物は Debian の一部ではありませんが、その使用をサポートし、フリーではないパッケージのための (バグ追跡システムやメーリングリストのような) インフラストラクチャーを用意しています。

注意
Debian 社会契約 1.2 の第 5 項の実際の文言は上記と少々違います。この編集上導入したズレは、社会契約の本質的内容を変えること無く本ユーザー文書の自己整合性を確保するために意識的に作られたズレです。

ユーザーは non-free や non-free-firmware や contrib エリア中のパッケージを使用するリスクを認識すべきです。

- そのようなソフトウェアーパッケージに関する自由の欠如
- そのようなソフトウェアーパッケージに関する Debian からのサポートの欠如 (Debian はソフトウェアーのソースコードに適切なアクセスなしにはソフトウェアーをサポートできません。)
- あなたの 100% フリーソフトウェアーの Debain システムへの汚染

Debian フリーソフトウェアーガイドラインは Debian のフリーソフトウェアー基準です。Debian は「ソフトウェアー」に関して、パッケージ中の文書、ファームウエアー、ロゴ、アート作品を含む最も広義の解釈をします。このことにより Debian のフリーソフトウェアー基準は非常に厳格なものとなります。

典型的な non-free や non-free-firmware や contrib パッケージは以下のタイプの自由に頒布できるパッケージを含んでいます。

- GCC や Make 等の変更不可部分付きの GNU フリー文書利用許諾契約書に基づく文書パッケージ。(主に non-free/doc セクション中にある)
- 項 9.10.5 に列記された中で non-free-firmware とあるソースコード無しのバイナリーデーターを含むファームウエアーパッケージ。(主に non-free-firmware/kernel セクション中にある)
- 商用使用やコンテント変更に関する制約のあるゲームやフォントのパッケージ。

non-free と non-free-firmware と contrib パッケージの数は main パッケージの数の 2% 以下ということを承知下さい。non-free や non-free-firmware や contrib エリアへのアクセスを有効にしてもパッケージソースは不明瞭になりません。aptitude(8) をインタラクティブでフルスクリーンに使用すると、どのエリアからどのパッケージをインストールするのかを完全に可視化しコントロールできるので、あなたのシステムをあなたの意向通りの自由の程度に合わせて維持できます。

2.1.7 パッケージ依存関係

Debian システムはコントロールファイル中のバージョン情報付きのバイナリー依存関係宣言を通して整合性のあるバイナリーパッケージの集合を提供します。ここにその少々簡素化し過ぎの定義を示します。

- ”Depends”
  - これは絶対依存を宣言し、このフィールドにリストされた全てのパッケージは同時または事前にインストールされていなければいけません。
- ”Pre-Depends”
  - これは、リストされたパッケージが事前にインストールを完了している必要がある以外は、Depends と同様です。
- ”Recommends”
  - これは強いが絶対でない依存を宣言します。多くのユーザーはこのフィールドにリストされたパッケージ全てがインストールされていなければ、当該パッケージを望まないでしょう。
- ”Suggests”
  - これは弱い依存を宣言します。このパッケージの多くのユーザーはこのフィールドにリストされたパッケージをインストールすればメリットを享受できるとは言え、それら抜きでも十分な機能が得られます。
- ”Enhances”
  - これは Suggests 同様の弱い依存を宣言しますが、依存作用の方向が逆です。
- ”Breaks”
  - これは通常バージョン制約付きでパッケージのインコンパチビリティーを宣言します。一般的にこのフィールドにリストされた全てのパッケージをアップグレードすることで解決します。
- ”Conflicts”
  - これは絶対的排他関係を宣言します。このフィールドにリストされた全てのパッケージを除去しない限り当該パッケージをインストールできません。
- ”Replaces”
  - 当該パッケージによりインストールされるファイルがこのフィールドにリストされたパッケージのファイルを置き換える際にこれを宣言します。
- ”Provides”
  - 当該パッケージがこのフィールドにリストされたパッケージのファイルと機能の全てを提供する際にこれを宣言します。

注意
合理的な設定として”Provides” と”Conflicts” と”Replaces” とを単一バーチャルパッケージに対し同時宣言することが合理的な設定であることを承知下さい。こうするといかなる時にも当該バーチャルパッケージを提供する実パッケージのうち確実に一つだけがインストールされます。

ソースの依存関係をも含む正式の定義は the Policy Manual: Chapter 7 - Declaring relationships between packages にあります。

2.1.8 パッケージ管理のイベントの流れ

パッケージ管理の簡略化されたイベントの流れをまとめると以下のようになります。

- 更新 (”`apt update`” か”`aptitude update`” か”`apt-get update`”):
  1. アーカイブメタデーターをリモートアーカイブから取得
  2. APT が使えるようローカルメタデーターの再構築と更新
- 更新 (”`apt upgrade`” と”`apt full-upgrade`” か、”`aptitude safe-upgrade`” と”`aptitude full-upgrade`” か、”`apt-get upgrade`” と”`apt-get dist-upgrade`”):
  1. 全てのインストール済みパッケージに関して、通常最新の利用可能なバージョンを選択します (例外については項 2.7.7 を参照下さい)
  2. パッケージ依存関係解決の実行
  3. もし候補バージョンがインストール済みバージョンと異なる際には、選ばれたバイナリーパッケージをリモートアーカイブから取得
  4. 取得バイナリーパッケージの開梱
  5. **preinst** スクリプトの実行
  6. バイナリーファイルのインストール
  7. **postinst** スクリプトの実行
- インストール (”`apt install …`” か”`aptitude install …`” か”`apt-get install …`”):
  1. コマンドラインにリストされたパッケージの選択
  2. パッケージ依存関係解決の実行
  3. 選ばれたバイナリーパッケージをリモートアーカイブから取得
  4. 取得バイナリーパッケージの開梱
  5. **preinst** スクリプトの実行
  6. バイナリーファイルのインストール
  7. **postinst** スクリプトの実行
- 削除 (”`apt remove …`” か”`aptitude remove …`” か”`apt-get remove ...`”):
  1. コマンドラインにリストされたパッケージの選択
  2. パッケージ依存関係解決の実行
  3. **prerm** スクリプトの実行
  4. 設定ファイル以外のインストール済みファイルの削除
  5. **postrm** スクリプトの実行
- 完全削除 (”`apt purge ...`” か”`aptitude purge …`” か”`apt-get purge ...`”):
  1. コマンドラインにリストされたパッケージの選択
  2. パッケージ依存関係解決の実行
  3. **prerm** スクリプトの実行
  4. 設定ファイルを含めたインストール済みファイルの削除
  5. **postrm** スクリプトの実行

上記では全体像の理解のためにわざと技術詳細を端折っています。

2.1.9 パッケージ管理のトラブルへの応急対処法

内容が正確な正式文書を読むように心がけるべきです。まず Debian に特定のことが記載された”/usr/share/doc/*packa*
を最初に読むべきです。また”/usr/share/doc/*package_name*/” の中にある他の文書も参照すべきです。項 1.4.2
に書かれたようなシェル設定がされていれば、以下のようにタイプして下さい。

```
$ cd package_name
$ pager README.Debian
$ mc
```

さらに詳しい情報を得るには”-doc” というサフィクスを持った対応する文書パッケージをインストールする必要があるかもしれません。

特定パッケージに関する問題に出会った際には、Debian バグトラッキングシステム (BTS) サイトを必ず確認します。

ウェッブサイト	コマンド
Debian バグトラッキングシステム (BTS) のホームページ	`sensible-browser "https://bugs.debian.org/"`
既知のパッケージに関するバグレポート	`sensible-browser "https://bugs.debian.org/`*`package_name`*`"`
既知のバグ番号に関するバグレポート	`sensible-browser "https://bugs.debian.org/`*`bug_number`*`"`

Table 2.5: 特定パッケージの問題解決のためのキーとなるウェッブサイトのリスト

”`site:debian.org`” や”`site:wiki.debian.org`” や”`site:lists.debian.org`” 等を含む検索語で Google を検索します。

バグ報告をする際には、reportbug(1) コマンドを使います。

2.1.10 Debian パッケージの選択方法

2 つ以上の似たパッケージに出会い” 試行錯誤” の努力無しにどのパッケージをインストールするか迷った際には、常識を使って下さい。次に示す点は好ましいパッケージの良い指標と考えます。

- 必須 (essential): yes > no
- エリア (area): メイン (main) > contrib > non-free
- 優先度 (priority): 必須 (required) > 重要 (important) > 標準 (standard) > 任意 (optional) > 特別 (extra)
- タスク (tasks): ” デスクトップ環境” のようなタスクにリストされたパッケージ
- 依存パッケージにより選ばれたパッケージ (例えば、`gcc` による `gcc-16`)
- ポプコン: 投票やインストールの数が多い
- changelog: メンテナによる定期的更新
- BTS: RC bug が無いこと (critical も grave も serious もいずれのバグも無い)
- BTS: バグレポートに反応の良いメンテナ
- BTS: 最近修正されたバグの数が多い
- BTS: wishlist 以外のバグが少ない

Debian は分散型の開発モデルのボランティアプロジェクトですので、そのアーカイブには目指すところや品質の異なる多くのパッケージがあります。これらをどうするかは自己判断をして下さい。

2.1.11 矛盾した要求への対処法

どのスイートの Debian システムを使うと決めようと、そのスイートで利用可能となっていないプログラムのバージョンをなんとか実行したいかもしれません。そのようなプログラムのバイナリーパッケージが他の Debian スイートとか他の非 Debian リソースで見つかるかもしれませんが、あなたの現行の Debian システムはそれらの要求条件と相容れないかもしれません。

そのような不同期のバイナリーパッケージをインストールする項 2.7.7 に記載されたような **apt-pinning** テクニック等を用いてパッケージ管理システムを微調整することができるとはいえ、そのような微調整はそれらのプログラムやあなたのシステムを壊すかもしれないので、そのような微調整のアプローチには限定的なユースケースしかありません。

そのような非同期のパッケージを強引にインストールする前に、あなたの現行の Debian システムとコンパチブルな、全ての利用可能な安全な技術的解決策を探索すべきです。

- 対応するサンドボックス化したアップストリームのバイナリーパッケージをインストール (項 7.7 を参照下さい)。
  - LibreOffice や GNOME アプリケーション等の多くの主に GUI プログラムは、Flatpak や Snap や AppImage パッケージが利用可能です。
- chroot か類似の環境を作り、その中でそのようなプログラムを実行 (項 9.11 を参照下さい)。
  - CLI コマンドは、それとコンパチブルな chroot 下で簡単に実行できます (項 9.11.4 を参照下さい)。
  - 複数のフルのデスクトップ環境をリブートすること無く簡単に試せます (項 9.11.5 を参照下さい)。
- あなたの現行の Debian システムとコンパチブルな望ましいバージョンのバイナリーパッケージをビルド。
  - これは、簡便でない操作です (項 2.7.13 を参照下さい)。

2.2 基本的パッケージ管理操作

Debian システム上でのレポジトリーを使ったパッケージ管理操作は Debian システム上にある多くの APT を使うパッケージ管理ツールを使いできます。ここでは、`apt` /`apt-get` / `apt-cache` や `aptitude` といった 3 つの基本的なパッケージ管理ツールを説明します。

パッケージをインストールしたりパッケージのメタデーターを更新するようなパッケージ管理操作には root 権限が必要です。

2.2.1 `apt` と `apt-get`/`apt-cache` と `aptitude` の比較

`aptitude` は筆者が主に使う非常に良いインタラクティブツールではありますが、注意すべき事実があることを知っておくべきです。

- `stable`(安定版) Debian システムにおいて、新リリースがあった後の新リリースシステムへのアップグレードに `aptitude` コマンドを使用することは推薦されません。
  - それには、”`apt full-upgrade`” か”`apt-get dist-upgrade`” を使うことが推薦されます。Bug #411280 参照下さい。
- `aptitude` コマンドは時折 `testing`(試験版) や `unstable` (不安定版) Debian システム上でシステムアップグレードをしようとする際に、大量のパッケージ削除を提案することが時々あります。
  - この状況は多くのシステム管理者を驚かせて来ました。パニックしないで下さい。
  - このようなことは `gnome-core` の様なメタパッケージにより依存や推薦されるパッケージ間のバージョンのずれにより発生するようです。

– この状況は aptitude コマンドのメニューから” 未実行アクションの取り消し” を選択し、aptitude を終了し、”apt full-upgrade” を使うことで解決できます。

apt-get や apt-cache コマンドは APT を使う最も基本的なパッケージ管理ツールです。

- apt-get/apt-cache はコマンドラインのユーザーインターフェースのみを提供します。
- apt-get はリリース間のような大掛かりなシステムアップグレードに最適です。
- apt-get は頑強で安定なパッケージリゾルバーを提供します。
- apt-get はハードウエアリソースへの要求が楽である。メモリーの消費は少なく、実行速度が早い。
- apt-cache はパッケージ名や説明に関して標準の regex を使った検索機能を提供します。
- apt-get と apt-cache は /etc/apt/preferences を使って複数のバージョンのパッケージを管理できますが、それはとても面倒です。

apt コマンドはパッケージ管理のための上位コマンドラインインターフェースです。基本的に apt-get や apt-cache 等のコマンドのラッパーで、インタラクティブな用途に良いオプションをデフォルトで有効にしてエンドユーザーインターフェース向けとなっています。

- apt は、apt install としてパッケージをインストールするとフレンドリーなプログレスバーを提供します。
- apt は、ダウンロードされたパッケージが上手くインストールされた後、デフォルトでキャッシュされた .deb パッケージを削除します。

ティップ
ユーザーはインタラクティブ用途には apt(8) コマンドを使うことが推奨されますし、シェルスクリプト中では apt-get(8) や apt-cache(8) コマンドを使うことが推奨されます。

---

aptitude コマンドは最も多芸な APT を使うパッケージ管理ツールです。

- aptitude はフルスクリーンのインタラクティブなテキストユーザーインターフェースを提供します。
- aptitude はコマンドラインのユーザーインターフェースも提供します。
- aptitude はインストール済みパッケージを検査したり利用可能なパッケージを探索したりするような日常のインタラクティブなパッケージ管理に最適です。
- aptitude はハードウエアリソースへの要求が厳しい。メモリーの消費は多く、実行速度も遅い。
- aptitude はパッケージメタデータ全てに関する拡張された regex を使った探索を提供します。
- aptitude は /etc/apt/preferences を使わずに複数のバージョンのパッケージを管理できますし、それは非常に直感的です。

2.2.2 コマンドラインによる基本的なパッケージ管理操作

apt(8) や aptitude(8) や apt-get(8) /apt-cache(8) を使うコマンドラインによるパッケージ管理操作を次に記します。

apt / apt-get と aptitude は大きな問題なく混用が可能です。

”aptitude why *regex*” は”aptitude -v why *regex*” とすることで、さらに詳しい情報を表示します。同様の情報は”apt rdepends *package*” や”apt-cache rdepends *package*” とすることでも得られます。

aptitude コマンドが最初コマンドラインモードで実行されパッケージ間のコンフリクトのような問題に直面した場合は、プロンプトがでた際に”e” を押すことでフルスクリーンのインタラクティブモードに切り替えられます。

apt シンタックス	**aptitude** シンタックス	**apt-get/apt-cache** シンタックス	説明
`apt update`	`aptitude update`	`apt-get update`	パッケージアーカイブメタデーター更新
`apt install foo`	`aptitude install foo`	`apt-get install foo`	"foo" パッケージの候補バージョンをその依存関係とともにインストール
`apt upgrade`	`aptitude safe-upgrade`	`apt-get upgrade`	他のパッケージを削除すること無くインストール済みパッケージの候補バージョンをインストール
`apt full-upgrade`	`aptitude full-upgrade`	`apt-get dist-upgrade`	必要なら他のパッケージを削除しながらインストール済みパッケージの候補バージョンをインストール
`apt remove foo`	`aptitude remove foo`	`apt-get remove foo`	設定ファイルを残したまま"foo" パッケージを削除
`apt autoremove`	N/A	`apt-get autoremove`	既に必要なくなっている自動済みパッケージを削除
`apt purge foo`	`aptitude purge foo`	`apt-get purge foo`	設定ファイルを含めて"foo" パッケージを完全削除
`apt clean`	`aptitude clean`	`apt-get clean`	収集されローカルに貯蔵されたパッケージファイルを完全消去
`apt autoclean`	`aptitude autoclean`	`apt-get autoclean`	収集されローカルに貯蔵されたパッケージファイルのうち古くなったパッケージを消去
`apt show foo`	`aptitude show foo`	`apt-cache show foo`	"foo" パッケージに関する詳細情報を表示
`apt search` *regex*	`aptitude search` *regex*	`apt-cache search` *regex*	*regex* とマッチするパッケージを検索
N/A	`aptitude why` *regex*	N/A	なぜ *regex* とマッチするパッケージがインストールされるのかを説明
N/A	`aptitude why-not` *regex*	N/A	なぜ *regex* とマッチするパッケージがインストールされないのかを説明
`apt list --manual-installed`	`aptitude search '~i!~M'`	`apt-mark showmanual`	手動インストール済みパッケージをリスト

Table 2.6: apt(8) やaptitude(8) やapt-get(8) /apt-cache(8) を使うコマンドラインによる基本パッケージ管理操作

注意
aptitude コマンドはその拡張されたパッケージリゾルバーのような豊富な機能が同梱されていますが、この複雑さは Bug #411123 や Bug #514930 や Bug #570377 のようないくつかのリグレッションを引き起こしました (また起こしているかもしれません)。疑義のある場合には、aptitude コマンドに代えて apt や apt-get や apt-cache コマンドを使って下さい。

”aptitude” のすぐ後ろにコマンドオプションをつけられます。

コマンドオプション	説明
-s	コマンド結果のシミュレート
-d	インストール / アップグレード無しにダウンロードのみする
-D	自動的なインストールや削除の前に簡単な説明を表示

Table 2.7: aptitude(8) に関する特記すべきコマンドオプション

詳細は aptitude(8) や”/usr/share/doc/aptitude/README” にある”aptitude user’s manual” を参照下さい。

2.2.3 aptitude のインタラクティブな使用

インタラクティブなパッケージ管理のためには aptitude をインタラクティブモードでコンソールのシェルプロンプトから以下のように立ち上げます。

```
$ sudo aptitude -u
Password:
```

これによりアーカイブ情報のローカルコピーは更新され、フルスクリーンのパッケージリストがメニュー付きで表示されます。Aptitude の設定ファイルは”~/.aptitude/config” にあります。

ティップ
user の設定ファイルでなく root の設定ファイルを使いたい際には、上記の例で”sudo aptitude …” の代わりに”sudo -H aptitude …” を使います。

ティップ
Aptitude はインタラクティブに起動されると次にするアクションを自動的に設定します。その設定が好ましくない場合はメニュー: ”Action” → ”Cancel pending actions” からリセットすることができます。

2.2.4 aptitude のキーバインディング

パッケージの状態を閲覧し、” 予定のアクション” の設定をこのフルスクリーンモードで各パッケージするための重要なキーを次に記します。

コマンドラインのファイル名の規定や、”l” や”//” を押した後のメニュープロンプトは次に記す aptitude の regex (正規表現) が使われます。aptitude の regex は”~n” で始めそれにパッケージ名を続けた文字列を使うことで明示的にパッケージ名とマッチさせられます。

ティップ
ビジュアルインターフェースで全てのインストール済みパッケージを候補バージョンにアップグレードさせるには”U” を押さなければいけません。これをしないと選ばれたパッケージとそれにバージョン付きの依存関係のある特定のパッケージのみが候補バージョンにアップグレードされます。

キー	キーバインディング
F10 もしくくは Ctrl-t	メニュー
?	(より詳細な) キーの意味のヘルプの表示
F10 → ヘルプ → ユーザーマニュアル	ユーザーマニュアルの表示
u	パッケージアーカイブ情報の更新
+	パッケージをアップグレードまたはインストールするとマーク
-	パッケージを削除するとマーク (設定ファイルは温存)
_	パッケージを完全削除するとマーク (設定ファイルも削除)
=	パッケージをホールド
U	全てのアップグレード可能なパッケージをマーク (**full-upgrade** として機能)
g	選ばれたパッケージのダウンロードとインストールをスタート
q	現在のスクリーンを終了し変更を保存
x	現在のスクリーンを終了し変更を廃棄
Enter	パッケージに関する情報閲覧
C	パッケージの変更履歴を閲覧
l	表示されるパッケージの制限を変更
/	最初のマッチを検索
\	最終検索の反復

Table 2.8: aptitude のキーバインディングのリスト

2.2.5 **aptitude** の下でのパッケージの表示

インタラクティブなフルスクリーンモードの aptitude(8) はパッケージリスト中のパッケージは以下の例のように表示されます。

```
idA   libsmbclient                              -2220kB 3.0.25a-1  3.0.25a-2
```

上記の行は左から次に記すような意味です。

- ” 現状” フラグ (1 番目の文字)
- ” 予定のアクション” フラグ (2 番目の文字)
- ” 自動” フラグ (3 番目の文字)
- パッケージ名
- ” 予定のアクション” に帰属されるディスク空間の使用の変化
- パッケージの現バージョン
- パッケージの候補バージョン

> ティップ
> ”?” を押して表示されるヘルプスクリーンの一番下に全フラグのリストがあります。

現在のローカルの環境設定によって候補バージョンは選ばれます (apt_preferences(5) と項 2.7.7 を参照下さい)。

” 表示” メニューの下にある数種のパッケージ表示が利用できます。

標準” パッケージ画面” はパッケージを dselect にいくつかの機能を加えた感じで分類します。

> ティップ
> Tasks ビューはあなたのタスクに使うパッケージをいいとこ取りするのに使えます。

表示	ビューの説明
`Package View`	表 2.10 を参照 (デフォールト)
`Audit Recommendations`	何らかのインストール済みパッケージによって推薦されているがインストールされていないパッケージをリスト
`Flat Package List`	パッケージを分類せずにリスト (regex とともに使用)
`Source Package View`	ソースパッケージごとに分けてパッケージのリストを表示

Table 2.9: aptitude の表示のリスト

分類	ビューの説明
`Upgradable Packages`	`section` → `area` → `package` と整理してパッケージをリスト
`New Packages`	,,
`Installed Packages`	,,
`Not Installed Packages`	,,
`Obsolete and Locally Created Packages`	,,
`Virtual Packages`	同一機能のパッケージをリスト
`Tasks`	タスクに一般的に必要な機能を持つパッケージのリスト

Table 2.10: 標準パッケージ画面の分類

2.2.6 aptitude を使った探索方法

Aptitude はその regex 式機能を通してパッケージを探索する方法をいくつか提供します。

- シェルコマンドライン:
  - マッチするパッケージのインストール状態やパッケージ名や短い説明をリストをすには、"`aptitude search '`*`aptitude_regex`*`'`"
  - パッケージの詳細説明のリストをするには、"`aptitude show '`*`package_name`*`'`"
- 対話型フルスクリーンモード:
  - マッチするパッケージにパッケージビューを絞る、"`l`"
  - マッチするパッケージを探す、"`/`"
  - マッチするパッケージを逆方向に探す、"`\`"
  - 次を探す、"`n`"
  - 次を逆方向に探す、"`N`"

ティップ
package_name という文字列は、"~" で始めて regex 式と明示されていない限り、パッケージ名との完全な一致検索として扱います。

2.2.7 aptitude の regex 式

aptitude の regex 式は mutt 的な拡張 **ERE** (項 1.6.2 を参照下さい) で `aptitude` に特定なマッチ規則の拡張は次に示すとおりです。

- regex 部分は、"^" や"`.*`" や"$" などを使う egrep(1) や awk(1) や perl(1) といった典型的な Unix 的テキストツールで使われる **ERE** と同様です。

拡張マッチ規則の説明	**regex** 式
パッケージ名とのマッチ	~n 名前の *regex*
記述とのマッチ	~d 記述の *regex*
タスク名とのマッチ	~t タスクの *regex*
debtag とのマッチ	~G*debtag* の *regex*
メンテナとのマッチ	~m*maintainer* の *regex*
パッケージセクションとのマッチ	~s セクションの *regex*
パッケージバージョンとのマッチ	~V バージョンの *regex*
アーカイブ (archive) とのマッチ	~A{trixie,forky,sid}
オリジン (origin) とのマッチ	~O{debian,…}
優先度 (priority) とのマッチ	~p{extra,important,optional,required,standard}
必須 (essential) パッケージとのマッチ	~E
仮想パッケージとのマッチ	~v
新規パッケージとのマッチ	~N
以下のアクションとのマッチ	~a{install,upgrade,downgrade,remove,purge,hold,keep}
インストール済みパッケージとのマッチ	~i
A-マークのついたインストール済みパッケージとマッチ (自動インストール済みパッケージ)	~M
A-マークのついていないインストール済みパッケージとマッチ (管理者が選択したパッケージ)	~i!~M
インストール済みかつアップグレード可能なパッケージとマッチ	~U
削除済みだが完全削除されていないパッケージとマッチ	~c
削除済みか完全削除済みか削除可能なパッケージとマッチ	~g
壊れた依存関係宣言をしたパッケージとマッチ	~b
type の壊れた依存関係を宣言しているパッケージとマッチ	~B*type*
type の壊れた依存関係を宣言している *pattern* パッケージとマッチ	~D[*type*:]*pattern*
type の壊れた依存関係を宣言している *pattern* パッケージとマッチ	~DB[*type*:]*pattern*
pattern マッチするパッケージが *type* の依存関係を宣言しているパッケージとマッチ	~R[*type*:]*pattern*
pattern マッチするパッケージが *type* の壊れた依存関係を宣言しているパッケージとマッチ	~RB[*type*:]*pattern*
他のインストール済みパッケージが依存するパッケージとマッチ	~R~i
他のインストール済みパッケージが一切依存しないパッケージとマッチ	!~R~i
他のインストール済みパッケージが依存もしくは推薦するパッケージとマッチ	~R~i|~Rrecommends:~i
フィルターされたバージョンの *pattern* とマッチ	~S filter *pattern*
常に全てのパッケージにマッチ (真)	~T
どのパッケージにもマッチしない (偽)	~F

Table 2.11: aptitude の regex 式のリスト

- 依存関係を表す *type* はパッケージの相互関係を指定する (depends, predepends, recommends, suggests, conflicts, replaces, provides) の内の 1 つです。
- デフォールトの依存タイプは”depends” です。

ティップ
regex_pattern がヌル文字列の場合は”~T” をコマンドの直後に使って下さい。

次がショートカットです。

- ”~P*term*” == ”~Dprovides:*term*”
- ”~C*term*” == ”~Dconflicts:*term*”
- ”…~W term” == ”(…|term)”

mutt が表現のお手本なので、mutt に慣れているユーザーはすぐ慣れるでしょう。”User’s Manual” (”/usr/share/doc/apt 中の”SEARCHING, LIMITING, AND EXPRESSIONS” を参照下さい。

注意
lenny バージョンの aptitude(8) では、新規の”?broken” のような長形式の regex マッチ形式が、古い”~b” のような短形式のマッチ形式に代えて使えます。そのためチルダ文字”~” に加えてスペース文字” ” も regex の終端文字として扱われます。新規の長形式のマッチ形式については”User’s Manual” を参照下さい。

2.2.8 aptitude による依存関係の解決

aptitude によるパッケージの選択は、”F10 → Options → Preferences → Dependency handling” のメニュー設定に従って、”Depends:” リストに規定されたパッケージばかりでは無く”Recommends:” リストに規定されたパッケージも引き込みます。このような自動インストール済みパッケージは不要になると aptitude が自動的に削除します。

aptitude コマンドの” 自動インストール” 挙動を制御するフラグは apt パッケージ中の apt-mark(8) コマンドを用いても操作できます。

2.2.9 パッケージ活動ログ

パッケージ活動履歴はログファイルで確認できます。

ファイル	内容
/var/log/dpkg.log	全パッケージ活動の dpkg レベルの活動ログ
/var/log/apt/term.log	汎用 APT 活動ログ
/var/log/aptitude	aptitude コマンド活動ログ

Table 2.12: パッケージ活動のログファイル

これらのログから意味のある理解を迅速に得るのは実際には難しいです。より簡単な方法については項 9.3.9 を参照下さい。

2.3 aptitude 操作例

aptitude(8) 操作例を次に示します。

2.3.1 興味あるパッケージの探索

パッケージの説明や”Tasks” の下のリストを使ってあなたが必要なパッケージを aptitude で見つけることができます。

2.3.2 regex にマッチするパッケージ名のパッケージをリスト

以下のコマンドはパッケージの名前が regex にマッチするパッケージをリストします。

```
$ aptitude search '~n(pam|nss).*ldap'
v   libnss-ldap                                                -
p   libnss-ldapd                                               - NSS module for using LDAP  ↩
    as a naming service
v   libpam-ldap                                                -
p   libpam-ldapd                                               - PAM module for using LDAP  ↩
    as an authentication service
```

これはパッケージの正確な名前を探すときに非常に便利です。

2.3.3 regex マッチをしての閲覧

” 平坦なパッケージリスト” のビューで”l” のプロンプトに regex ”~dipv6” を入れるとその意味にマッチするパッケージにビューが制限され、その情報をインタラクティブに閲覧できます。

2.3.4 パッケージの完全削除

削除したパッケージが残した全ての設定ファイルを以下のようにして完全削除できます。

以下のコマンドの結果をチェックします。

```
# aptitude search '~c'
```

もしリストされたパッケージが完全削除されても問題ないなら、以下のコマンドを実行します。

```
# aptitude purge '~c'
```

同様のことをインタラクティブにすればよりきめの細かい結果が得られます。

” 新規パッケージ画面” のビューで”l” のプロンプトに regex ”~c” を入れると regex にマッチする” 削除されたが完全駆除されていない” パッケージにビューが制限されます。トップレベルの見出しの上で”[” を押すと regex にマッチする全てのパッケージが表示されます。

次に” インストールされていないパッケージ” 等のトップレベルの見出しの上で”_” を押します。その見出しの下の regex にマッチするパッケージだけが完全削除と設定されます。インタラクティブに個々のパッケージの上で”=” を押せばそれらのパッケージを完全削除対象から外せます。

このテクニックは非常に便利で、他の多くのコマンドキーでも使えます。

2.3.5 自動 / 手動インストール状態の整理

(非 aptitude のパッケージインストーラー等を使った後で) パッケージの自動 / 手動インストールの状態を整理する私の方法を次に記します。

1. aptitude を root としてインタラクティブに起動します。

2. "u" と"U" と"f" と"g" とタイプしてパッケージリストを更新しパッケージをアップグレードします。
3. パッケージ表示制限を"~i(~R~i|~Rrecommends:~i)" と入力するために"l" とタイプし、自動インストール済みとするように"Installed Packages" (インストール済みパッケージ) の上で"M" とでタイプします。
4. パッケージ表示制限を"~required|~pimportant|~pstandard|~E" と入力するために"l" とタイプし、手動インストールとなるよう"m" と"Installed Packages" (インストール済みパッケージ) の上でタイプします。
5. パッケージ表示制限を"~i!~M" と入力するために"l" とタイプし、"Installed Packages" (インストール済みパッケージ) の上で"[" とタイプしてパッケージを見えるようにした後で個々のパッケージの上で"-" とタイプして使っていないパッケージを削除します。
6. パッケージ表示制限を"~i" と入力するように"l" とタイプし、そして" タスク" の上で手動インストールとなるよう"m" とタイプします。
7. aptitude を終了します。
8. "apt-get -s autoremove|less" と root から起動して何が使われていないのか確認します。
9. aptitude とインタラクティブモードで再起動して必要なパッケージを"m" でマークします。
10. "apt-get -s autoremove|less" と root から再起動して削除対象が期待にかなっていることを再確認します。
11. "apt-get autoremove|less" と root から起動して使用していないパッケージを自動削除します。

"Tasks" の上で"m" を押すのも一案で、大量ファイル除去となる事態が回避できます。

2.3.6 システム全体のアップグレード

注意
新規リリース等への移行は、Debian では下記のようにアップグレードできるのですが、新たなシステムをクリーンインストールすることを考えるべきです。こうすると溜めてきたゴミの除去ができる上に最新のパッケージの最良の組み合わせも分かります。もちろん安全な場所に完全なシステムのバックアップ (項 10.2 を参照下さい) を事前にしなくてはいけません。異なったパーティションを使ったデュアルブート設定をすることをスムーズな移行をするためにお薦めします。

ソースリストの内容を新規リリースへ向けるように変更し、"apt update; apt dist-upgrade" コマンドを実行することでシステム全体のアップグレードができます。

安定版 stable リリースが trixie の期間中の、安定版 stable からテスト版 testing や不安定版 unstable にアップグレードするには、項 2.1.5 にあるソースリスト例の"trixie" を"forky" か"sid" に置き換えます。

一部のパッケージで移行に関して支障をきたすことが実際には起こるかもしれません。これは大体パッケージ依存関係に起因します。アップグレードする差が大きければ大きいほど比較的大きな問題似合う可能性がより大きくなります。以前の安定版 stable からリリース後の新規安定版 stable への移行では新規リリースノートを読んでそこに記載された手続き通りに完全にすれば問題発生を防げます。

安定版 stable からテスト版 testing へ移行すると決めた時には頼りにするリリースノートはありません。前回の安定版 stable のリリースの後で安定版 stable とテスト版 testing の差がかなり大きくなっているかもしれません。そうだとアップグレードをする状況は複雑になっています。

メーリングリストから最新情報を収集するとか常識を使うといった予防措置をしながらフルアップグレードをするべきです。

1. 前回の" リリースノート" を読みます。
2. 全システム (特にデーターや設定情報) をバックアップします。

3. ブートローダーが壊れたときのためにブートできるメディアを確保します。
4. システムを使っているユーザーに十分事前に通告します。
5. script(1) を使ってアップグレード活動を記録します。
6. 削除をされないように”`aptitude unmarkauto vim`” 等として、”unmarkauto” を重要なパッケージに適用します。
7. デスクトップタスクにあるパッケージ等を削除して、インストール済みパッケージを減らしてパッケージがコンフリクトする可能性を減らします。
8. ”`/etc/apt/preferences`” ファイルを削除します (**apt-pinning** を無効化)。
9. 段階的にアップグレードしましょう: 旧安定版 `oldstable` → 安定版 `stable` → テスト版 `testing` → 不安定版 `unstable`。
10. ソースリストを更新して新アーカイブ対象に”`aptitude update`” を実行します。
11. ”`aptitude install perl`” 等として、先に新規の中核的パッケージを必要に応じてインストールします。
12. ”`apt-get -s dist-upgrade`” コマンドを実行して影響を確認します。
13. 最後に”`apt-get dist-upgrade`” コマンドを実行します。



注意
`stable` リリース間でアップグレードする際に Debian のメジャーリリースを飛ばすのは賢明ではありません。



注意
過去の” リリースノート” ではシステム全体のアップグレードをするのに GCC や Linux カーネルや initrd-tools や Glibc や Perl や APT tool chain 等には特別な配慮が必要でした。



注意
「リリースノート」は可能性ある全ケースをカバーしていないかもしれません。低レベル設定を変更した場合、あなたの次のアップグレードは”... segfault after upgrade ...” のように上手く行かないかもしてません。

`unstable` での毎日のアップグレードは項 2.4.3 を参照下さい。

2.4 高度なパッケージ管理操作

2.4.1 コマンドラインによる高度なパッケージ管理操作

`aptitude` ではハイレベル過ぎるとか必要な機能を欠くという他のパッケージ管理操作のリストです。

注意
multi-arch 機能のあるパッケージに関して、一部のコマンドはアーキテクチャー名を必要があるかもしれません。例えば、`amd64` アーキテクチャーの `libglib2.0-0` パッケージの内容をリストするには”`dpkg -L libglib2.0-0:amd64`” を使います。

コマンド	アクション
`COLUMNS=120 dpkg -l` パッケージ名パターン	バグレポートのためにインストール済みパッケージの状態をリスト
`dpkg -L` パッケージ名	インストール済みパッケージの内容をリスト
`dpkg -L` パッケージ名 `| egrep '/usr/share/man/man.*/.+'`	インストール済みパッケージのマンページをリスト
`dpkg -S` ファイル名パターン	マッチするファイル名があるインストール済みパッケージをリスト
`apt-file search` ファイル名パターン	マッチするファイル名があるアーカイブ中のパッケージをリスト
`apt-file list` パッケージ名パターン	アーカイブ中のマッチするパッケージをリスト
`dpkg-reconfigure` パッケージ名	特定パッケージを再設定
`dpkg-reconfigure -plow` パッケージ名	もっとも詳細な質問で特定パッケージを再設定
`configure-debian`	フルスクリーンメニューからパッケージを再設定
`dpkg --audit`	部分インストール済みパッケージに関してシステムを監査
`dpkg --configure -a`	全ての部分インストール済みパッケージを設定
`apt-cache policy` バイナリーパッケージ名	バイナリーパッケージに関して利用可能なバージョンやプライオリティーやアーカイブ情報を表示
`apt-cache madison` パッケージ名	パッケージに関して利用可能なバージョンやアーカイブ情報を表示
`apt-cache showsrc` バイナリーパッケージ名	バイナリーパッケージに関してソースパッケージの情報を表示
`apt-get build-dep` パッケージ名	パッケージをビルドするのに必要なパッケージをインストール
`aptitude build-dep` *package_name*	パッケージをビルドするのに必要なパッケージをインストール
`apt-get source` パッケージ名	(標準アーカイブから) ソースをダウンロード
`dget` *dsc* ファイルの *URL*	(他のアーカイブから) ソースをダウンロード
`dpkg-source -x` パッケージ名`_`バージョン*-debian* のレビジョン`.dsc`	ソースパッケージの組 (”`*.tar.gz`” と”`*.debian.tar.gz`” / ”`*.diff.gz`”) からソースツリーをビルド
`debuild binary`	ローカルのソースツリーからパッケージをビルド
`make-kpkg kernel_image`	カーネルソースツリーからカーネルパッケージをビルド
`make-kpkg --initrd kernel_image`	カーネルソースツリーから initramfs を有効にしてカーネルパッケージをビルド
`dpkg -i` パッケージ名`_`バージョン*-debian* のレビジョン`_`アーキテクチャー名`.deb`	ローカルパッケージをシステムにインストール
`apt install /path/to/`*package_filename*`.deb`	自動的に依存関係を解決しながらローカルパッケージをシステムにインストールする
`debi` パッケージ名`_`バージョン*-debian* のレビジョン`_`アーキテクチャー名`.dsc`	ローカルパッケージ (複数) をシステムにインストール
`dpkg --get-selections '*' >selection.txt`	dpkg レベルのパッケージ選択状態情報を保存
`dpkg --set-selections <selection.txt`	dpkg レベルのパッケージ選択状態情報を設定
`echo` *package_name* `hold | dpkg --set-selections`	特定パッケージの dpkg レベルのパッケージ選択状態を **hold** にする (”`aptitude hold` *package_name*” と等価)

Table 2.13: 高度なパッケージ管理操作



注意
”dpkg -i …” や”debi …” といった低いレベルのパッケージツールはシステム管理者によって注意深く使われなければいけません。必要なパッケージ依存関係を自動的に面倒見てくれません。Dpkg の”--force-all” や類似のコマンドラインオプション (dpkg(1) 参照下さい) はエキスパートだけが使うようにできています。十分にその影響を理解せずに使うとシステム全体を壊してしまうかもしれません。

以下を承知下さい。

- 全てのシステム設定やインストールコマンドは root から実行なければいけません。
- regex (項 1.6.2 を参照下さい) を使う aptitude と異なり、他のパッケージ管理コマンドはシェルグロブ (項 1.5.6 を参照下さい) のようなパターンを使います。
- apt-file パッケージに入っている apt-file(1) は事前に”apt-file update” を実行する必要があります。
- configure-debian パッケージに入っている configure-debian(8) はそのバックエンドとして dpkg-reconfigure(8) を実行します。
- dpkg-reconfigure(8) はそのバックエンドとして debconf(1) を利用するパッケージスクリプトを実行します。
- ”apt-get build-dep” や”apt-get source” や”apt-cache showsrc” コマンドはソースリストの中に”deb-src” エントリーが必要です。
- dget(1) や debuild(1) や debi(1) は devscripts パッケージが必要です。
- ”apt-get source” を使った (再) パッケージ化の手続きは項 2.7.13 を参照下さい。
- make-kpkg コマンドは kernel-package パッケージが必要です (項 9.10 を参照下さい)。
- 一般的なパッケージ化に関しては項 12.9 を参照下さい。

2.4.2 インストール済みパッケージファイルの検証

debsums をインストールすると debsums(1) を使って”/var/lib/dpkg/info/*.md5sums” ファイル中の MD5sum 値との比較でインストール済みパッケージファイルを検証できます。MD5sum がどのような仕組かは項 10.3.5 を参照下さい。

注意
侵入者によって MD5sum のデーターベースが改竄されているかもしれないので debsums(1) はセキュリティーツールとしては限定的有用性しかありません。管理者によるローカルの変更や記憶メディアのエラーによる損傷を点検するぐらいには有用です。

2.4.3 パッケージ問題からの防御

多くのユーザーは新規機能やパッケージを求めて Debian システムのテスト版 **testing**(もしくは、非安定版 **unstable**) リリースを追いかけることを好みます。こういうことをするとクリティカルなパッケージのバグにシステムが遭遇しやすくなります。

apt-listbugs パッケージをインストールすれば、APT システムを使ってアップグレードする時に Debian の BTS を自動的にクリティカルなバグに関して点検することで、クリティカルなバグからあなたのシステムを防御できます。

apt-listchanges パッケージをインストールすれば、APT システムを使ってアップグレードする時に NEWS.Debian 中の重要ニュースを表示します。

2.4.4 パッケージメタデーターの検索

最近は Debian サイトの https://packages.debian.org/ を訪問するとパッケージメタデーターの検索を簡単に出きるようになっていますが、より伝統的な方法を見てみます。

grep-dctrl(1) や grep-status(1) や grep-available(1) コマンドは Debian のパッケージコントロールファイルの一般的フォーマットに従ういかなるファイルを検索するのにも使えます。

マッチする名前のファイルを含む dpkg でインストール済みパッケージ名を探索するのに”`dpkg -S ファイル名パターン`” が使えます。しかしメンテナスクリプトで生成されるファイルはこれでは見逃されます。

dpkg のメタデーターに関してより詳細な検索をする必要がある場合、”`/var/lib/dpkg/info/`” ディレクトリーで”`grep -e regex パターン *`” コマンドを実行しないといけません。こうすることでパッケージスクリプトやインストール時の質問テキスト中の言葉まで検索できます。

パッケージ依存関係を再帰的に検索したい際には、apt-rdepends(8) を使います。

2.5 Debian パッケージ管理の内部

Debian のパッケージ管理システムが内部的のどのように機能するのかを学びます。何らかのパッケージ問題が発生した際にあなた自身の解決を見出すのに役立つでしょう。

2.5.1 アーカイブのメタデーター

各ディストリビューションのメタデーターのファイルは例えば”`http://deb.debian.org/debian/`” のような各 Debian ミラーサイトの”`dist/`コード名” の下に保存されています。そのアーカイブ構造はウェッブブラウザーで閲覧できます。6 つのタイプの重要メタデーターがあります。

ファイル	場所	内容
`Release`	ディストリビューションのトップ	アーカイブの説明との整合性情報
`Release.gpg`	ディストリビューションのトップ	アーカイブキーで署名された”`Release`” ファイルに関する署名ファイル
`Contents-`アーキテクチャー	ディストリビューションのトップ	該当アーカイブ中全てのパッケージに関する全ファイルリスト
`Release`	各ディストリビューション / エリア / アーキテクチャーの組み合わせのトップ	apt_preferences(5) のルールに利用されるアーカイブの記述。
`Packages`	各ディストリビューション / エリア / バイナリーアーキテクチャーの組み合わせのトップ	バイナリーパッケージに関して `debian/control` を連結
`Sources`	各ディストリビューション / エリア / ソースの組み合わせのトップ	ソースパッケージに関して `debian/control` を連結

Table 2.14: Debian アーカイブのメタデーターの内容

最近のアーカイブではネットワークトラフィックを減らすべく圧縮された差分ファイルとしてこれらのメタデーターは保存されています。

2.5.2 トップレベルの”Release” ファイルと信憑性

ティップ
セキュアー **APT** システムではトップレベルの”Release” ファイルがアーカイブを署名するのに使われています。

Debian アーカイブの各スイーツには例えば次に示すような”http://deb.debian.org/debian/dists/unstable/Re
のようなトップレベルの”Release” ファイルがあります。

```
Origin: Debian
Label: Debian
Suite: unstable
Codename: sid
Date: Sat, 14 May 2011 08:20:50 UTC
Valid-Until: Sat, 21 May 2011 08:20:50 UTC
Architectures: alpha amd64 armel hppa hurd-i386 i386 ia64 kfreebsd-amd64 kfreebsd-i386 mips ↩
     mipsel powerpc s390 sparc
Components: main contrib non-free
Description: Debian x.y Unstable - Not Released
MD5Sum:
 bdc8fa4b3f5e4a715dd0d56d176fc789 18876880 Contents-alpha.gz
 9469a03c94b85e010d116aeeab9614c0 19441880 Contents-amd64.gz
 3d68e206d7faa3aded660dc0996054fe 19203165 Contents-armel.gz
...
```

注意
項 2.1.5 の中で” スイーツ (suite)” や” コード名 (codename)” を使う理由はこれを見れば分かるでしょう。” ディストリビューション” は” スイーツ” と” コード名” との両方を指したい際に用いられます。アーカイブが提供する全アーカイブ” エリア (area)” 名が”Components” の下にリストされます。

トップレベルの”Release” ファイルの整合性は apt-secure(8) で説明されるようにセキュアー apt という暗号学手法インフラストラクチャーによって検証されます。

- 暗号手法による署名ファイル”Release.gpg” は真正のトップレベルの”Release” ファイルと秘密の Debian アーカイブキーから作成されます。
- 公開の Debian アーカイブキーは最新の debian-archive-keyring パッケージによってローカルに導入されます。
- セキュアー **APT** システムはこの”Release.gpg” ファイルとローカルに導入された公開アーカイブキーを用いてダウンロードされたトップレベルの”Release” ファイルの整合性を暗号学手法を用いて自動的に検証します。
- ” 全ての Packages” と”Sources ファイルの整合性はそのトップレベルの”Release” ファイル中の MD5sum 値を用いて検証します。” パッケージファイルの整合性は”Packages” や”Sources” ファイル中の MD5sum 値を用いて検証します。debsums(1) と項 2.4.2 を参照下さい。
- 暗号学手法を用いた署名の検証は MD5sum 値の計算よりも非常に CPU を使うプロセスなので、トップレベルの”Release” ファイルには暗号学手法を用いた署名を使いつつ各パッケージには MD5sum 値を用いることでパーフォーマンスを保ったまま良好なセキュリティーが確保できます (項 10.3 を参照下さい)。

もしソースリストの記載内容が”signed-by” オプションを指定した場合には、指定された公開キーでダウンロードしたトップレベルの”Release” ファイルを指定した公開キーで検証されます。これは、ソースリストが非 Debian アーカイブを含むとき非常に有用です。

これとは別に、gpg を用いて”Release.gpg” ファイルと ftp-master.debian.org に掲示された公開アーカイブキーで”Release” ファイルの整合性を手動で検証できます。

2.5.3 アーカイブレベルの”Release” ファイル

ティップ
アーカイブレベルの”Release” ファイルが apt_preferences(5) のルールに使われます。

”http://deb.debian.org/debian/dists/unstable/main/binary-amd64/Release” や”http://deb.debian.
等のソースリストが指定する全てのアーカイブロケーションにはアーカイブレベルの次に示すような”Release” フ
ァイルがあります。

```
Archive: unstable
Origin: Debian
Label: Debian
Component: main
Architecture: amd64
```



注意
”Archive:” スタンザには、Debian アーカイブではスイート名 (”stable” や”testing” や”unstable” 等) が使われますが、Ubuntu アーカイブではコード名 (”trusty” や”xenial” や”artful”、…) が使われます。

experimental や trixie-backports のような自動的にインストールされるべきでないパッケージを含むような一部アーカイブでは次に示す”http://deb.debian.org/debian/dists/experimental/main/binary-amd64/
のような追加の行があります。

```
Archive: experimental
Origin: Debian
Label: Debian
NotAutomatic: yes
Component: main
Architecture: amd64
```

”NotAutomatic: yes” となっていない普通のアーカイブではデフォールトの Pin-Priority 値は 500 ですが、”NotAutomatic: yes” となっている特別なアーカイブではデフォールトの Pin-Priority 値は 1 であることを承知下さい (apt_preferences(5) と項 2.7.7 を参照下さい)。

2.5.4 パッケージメタデーターの取得

aptitude や apt-get や synaptic や apt-file や auto-apt 等の APT ツールが使われる際には Debian アーカイブ情報を含むメタデーターのローカルコピーを更新する必要があります。この様なローカルのコピーはソースリスト中のディストリビューション (distribution) とエリア (area) とアーキテクチャー (architecture) の名前に対応する以下のファイル名です (項 2.1.5 を参照下さい)。

- ”/var/lib/apt/lists/deb.debian.org_debian_dists_ ディストリビューション _Release”
- ”/var/lib/apt/lists/deb.debian.org_debian_dists_ ディストリビューション _Release.gpg”
- ”/var/lib/apt/lists/deb.debian.org_debian_dists_ ディストリビューション _ エリア _binary-アーキテクチャー _Packages”
- ”/var/lib/apt/lists/deb.debian.org_debian_dists_ ディストリビューション _ エリア _source_Sources”
- ”/var/cache/apt/apt-file/deb.debian.org_debian_dists_ ディストリビューション _Contents-アーキテクチャー.gz” (apt-file 用)

最初の 4 つのタイプのファイルは全ての適切な APT コマンド間で共有されておりコマンドラインから”apt-get update” や”aptitude update” によって更新されます。もしソースリスト中に”deb” が指定されていれば”Packages” メタデーターが更新されます。もしソースリスト中に”deb-src” が指定されていれば”Sources” メタデーターが更新されます。

”Packages” や”Sources” メタデーターはバイナリーやソースパッケージのファイルの場所を指している”Filename:” スタンザを含んでいます。現在、それらのパッケージはリリース間の移行を滞り無くするために”pool/” ディレクトリーツリーの下に置かれています。

”Packages” メタデーターのローカルコピーは aptitude を使ってインタラクティブに検索できます。grep-dctrl(1) という専用の検索コマンドを使うと”Packages” と”Sources” メタデーターのローカルコピーを検索できます。

”Contents-アーキテクチャー” メタデーターのローカルコピーは”apt-file update” で更新でき、他の 4 つと異なるところにあります。apt-file(1) を参照下さい。(auto-apt では”Contents-アーキテクチャー.gz” のローカルコピーがデフォールトでは異なるところにあります。)

2.5.5 APT に関するパッケージ状態

lenny 以降の APT ツールではリモートから取得したメタデーターに追加でローカルで生成されるインストール状態情報を”/var/lib/apt/extended_states” に保存して、自動インストールされた全パッケージを全ての APT ツールで追跡するのに用います。

2.5.6 aptitude に関するパッケージ状態

aptitude コマンドではリモートから取得したメタデーターに追加でローカルで生成されるインストール状態情報を”/var/lib/aptitude/pkgstates” に保存して用いています。

2.5.7 取得したパッケージのローカルコピー

APT メカニズムでリモートから取得されたパッケージは消去されるまでは”/var/cache/apt/archives” に貯蔵されます。

aptitude では、このキャッシュファイルのクリーニングポリシーは”Options” → ”Preferences” の下で設定でき、”Actions” の下の”Clean package cache” か”Clean obsolete files” メニューによって強制実行できる。

2.5.8 Debian パッケージファイル名

Debian のパッケージファイルには特定の名前の構造があります。

ティップ
ここでは基本的なパッケージフォーマットのみが記述されています。詳細は dpkg-source(1) を参照下さい。

注意
パッケージバージョンの順位は dpkg(1) を使って、例えば”dpkg --compare-versions 7.0 gt 7.~pre1 ; echo $?” とすると確認できます。

注意
Debian インストーラー (d-i) のバイナリーパッケージには、普通の deb ではなく udeb をファイル拡張子として使われます。udeb パッケージはポリシー条件を緩和しドキュメントのように必須でない内容を削除した減量 deb パッケージです。deb と udeb パッケージは同一のパッケージ構造を共有しています。”u” はマイクロと言う意味で使っています。

パッケージタイプ	名前の構造
バイナリーパッケージ (所謂 deb)	パッケージ名 _ アップストリームのバージョン-*debian* のレビジョン _ アーキテクチャー.deb
Debian インストーラー用のバイナリーパッケージ (所謂 udeb)	パッケージ名 _ アップストリームのバージョン-*debian* のレビジョン _ アーキテクチャー.udeb
ソースパッケージ (アップストリームのソース)	パッケージ名 _ アップストリームのバージョン-*debian* のレビジョン.orig.tar.gz
1.0 ソースパッケージ (Debian の変更部分)	パッケージ名 _ アップストリームのバージョン-*debian* のレビジョン.diff.gz
3.0 (quilt) ソースパッケージ (Debian の変更部分)	パッケージ名 _ アップストリームのバージョン-*debian* のレビジョン.debian.tar.gz
ソースパッケージ (内容記述)	パッケージ名 _ アップストリームのバージョン-*debian* のレビジョン.dsc

Table 2.15: Debian パッケージの名前の構造

名前の部分	使用可能文字 **(ERE regex)**	存在
パッケージ名	[a-z0-9][-a-z0-9.+]+	必須
エポック:	[0-9]+:	任意
アップストリームのバージョン	[-a-zA-Z0-9.+:]+	必須
debian のレビジョン	[a-zA-Z0-9.+~]+	任意

Table 2.16: Debian パッケージ名の各部分に使用可能な文字

2.5.9 dpkg コマンド

dpkg(1) は Debian パッケージ管理の最も低レベルのツールです。非常に強力ですから気をつけて使う必要があります。

" パッケージ名" というパッケージをインストールする際に、dpkg は次に記す順番でパッケージを処理します。

1. deb ファイルを解凍 ("ar -x" と等価)
2. debconf(1) を使い"*package_name*.preinst" を実行
3. システムにパッケージ内容をインストール ("tar -x" と等価)
4. debconf(1) を使い"*package_name*.postinst" を実行

debconf システムによって I18N と L10N (第 8 章) のサポートのある標準化されたユーザーとの対話が実現できます。

"status" ファイルは dpkg(1) や"dselect update" や"apt-get -u dselect-upgrade" のようなツールによって使われます。

grep-dctrl(1) という専用の検索コマンドを使うと"status" と"available" メタデーターのローカルコピーを検索できます。

ティップ
デビアンインストーラー環境下では、udpkg コマンドが udeb パッケージを開けるのに用いられます。udpkg コマンドはストリップダウンされたバージョンの dpkg コマンドです。

ファイル	内容の説明
/var/lib/dpkg/info/パッケージ名.conffiles	設定ファイルのリスト。(ユーザー変更可能)
/var/lib/dpkg/info/パッケージ名.list	パッケージによりインストールされるファイルやディレクトリーのリスト
/var/lib/dpkg/info/パッケージ名.md5sums	パッケージによりインストールされるファイルの MD5 ハッシュ値のリスト
/var/lib/dpkg/info/パッケージ名.preinst	パッケージインストールの前に実行するパッケージスクリプト
/var/lib/dpkg/info/パッケージ名.postinst	パッケージインストールの後に実行するパッケージスクリプト
/var/lib/dpkg/info/パッケージ名.prerm	パッケージ削除の前に実行するパッケージスクリプト
/var/lib/dpkg/info/パッケージ名.prerm	パッケージ削除の前に実行するパッケージスクリプト
/var/lib/dpkg/info/パッケージ名.conffiles	debconf システムのためのパッケージスクリプト
/var/lib/dpkg/alternatives/パッケージ名	update-alternatives コマンドが用いる代替情報
/var/lib/dpkg/available	すべてのパッケージの入手可能性情報
/var/lib/dpkg/diversions	dpkg(1) が利用し、dpkg-divert(1) が設定するする迂回情報
/var/lib/dpkg/statoverride	dpkg(1) が利用し、dpkg-statoverride(1) が設定する状態オーバーライド情報
/var/lib/dpkg/status	全パッケージに関する状態情報
/var/lib/dpkg/status-old	"/var/lib/dpkg/status" ファイルの第一世代のバックアップ
/var/backups/dpkg.status*	"/var/lib/dpkg/status" ファイルの第二世代以前のバックアップ

Table 2.17: dpkg が作成する特記すべきファイル

2.5.10 update-alternative コマンド

Debian システムには update-alternatives(8) を用いて何らかの重畳するプログラムを平和裏にインストールするメカニズムがあります。例えば vim と nvi の両方のパッケージがインストールされた状況下で vi コマンドが vim を選択して実行するようにできます。

```
$ ls -l $(type -p vi)
lrwxrwxrwx 1 root root 20 2007-03-24 19:05 /usr/bin/vi -> /etc/alternatives/vi
$ sudo update-alternatives --display vi
...
$ sudo update-alternatives --config vi
  Selection    Command
 -----------------------------------------------
      1        /usr/bin/vim
*+    2        /usr/bin/nvi

Enter to keep the default[*], or type selection number: 1
```

Debian の代替 (alternatives) システムは、その選択を"/etc/alternatives/" の中のシムリンクとして保持します。選択プロセスには"/var/lib/dpkg/alternatives/" の中の対応するファイルが使われます。

2.5.11 dpkg-statoverride コマンド

dpkg-statoverride(1) コマンドで提供される状態の上書きは、パッケージをインストールする際にファイルに関して異なる所有者やモードを使うよう dpkg(1) に指示する方法です。もし"--update" が指定されファイルが存在すれば、即座に新たな所有者やモードに設定されます。



注意
パッケージが所有するファイルの所有者やモードをシステム管理者が chmod や chown コマンドを用いて直接変更しても次回パッケージアップグレードによりリセットされます。

注意
ここでファイルと言いましたが、実際には dpkg が扱うディレクトリーやデバイス等のいかなるファイルシステムオブジェクトであってもいいです。

2.5.12 dpkg-divert コマンド

dpkg-divert(1) コマンドによって提供されるファイル迂回は、ファイルをデフォールトの場所ではなく迂回した場所にインストールするように dpkg(1) にさせます。dpkg-divert は本来パッケージメインテナンススクリプトのためのものです。システム管理者がこれを軽々に使うのは非推奨です。

2.6 壊れたシステムからの復元

テスト版 testing や非安定版 unstable システムを実行する時には、管理者には壊れたパッケージ管理状況から復元できることが望まれます。



注意
ここで説明するいくつかの方法は非常にリスクが高いアクションです。警告しましたよ!

2.6.1 古いユーザーの設定との非互換性

もしデスクトップ GUI プログラムが上流の大きなバージョンアップグレードの後に不安定性を経験した際には、そのプログラムが作った古いローカル設定ファイルとの干渉を疑うべきです。もし新規作成したユーザーアカウントでそのプログラムが安定なら、この仮説が裏付けられます。(これはパッケージングのバグで、通常パッケージャーによって回避されます。)

安定性を復元するには、対応するローカル設定ファイルを移動し GUI プログラムを再スタートします。後日設定情報を復元するために古い設定ファイルの内容を読む必要があるかもしれません。(あまり慌てて消去しないようにしましょう。)

2.6.2 パッケージデーターのキャッシングエラー

APT を使うと、"GPG error: ... invalid: BADSIG ..." 等の首を傾げたくなるエラーがパッケージデーターのキャッシュエラーで引き起こされます。

全てのキャッシュデーターを"`sudo rm -rf /var/lib/apt/*` "で削除しもう一度トライしましょう。(`apt-cacher-n`が使われた場合には"`sudo rm -rf /var/cache/apt-cacher-ng/*` "も実行しましょう。)

2.6.3 dpkg コマンドを使っての救済

`dpkg` は非常に低レベルフのパッケージツールなので、ネットワーク接続なしでも機能します。

`foo` パッケージが壊れていて修復の必要があると仮定しましょう。

バグの無い古いバージョンの `foo` パッケージが"`/var/cache/apt/archives/`" にあるパッケージキャッシュの中に見つかるかもしれません。(ここにみつからなければ、https://snapshot.debian.org/ アーカイブからダウンロードしたり、機能している機器のパッケージキャッシュからコピーできます。)

もしブート不可能な場合には、以下のコマンドを使ってインストールすることもできます。

```
# dpkg -i /path/to/foo_old_version_arch.deb
```

もしこの方法でパッケージをインストールしようとして何らかの依存関係違反のためにうまくいかなくてどうしようもなくなった場合には、`dpkg` の"`--ignore-depends`" や"`--force-depends`" や他のオプションを使って依存関係をオーバーライドすることができます。こうした場合には、後で適正な依存関係を修復するように真剣に取り組む必要があります。詳細は dpkg(8) を参照下さい。

注意
システムがひどく壊れた場合には、システムを安全な場所に完全バックアップし (項 10.2 を参照下さい)、クリーンインストールを実行するべきです。こうすることは時間の節約でもあり最終的に良い結果に結びつきます。

ティップ
システムがそれほど壊れていないなら、項 2.7.11 に書かれているようにして、より高レベルの APT システムを通じてシステム全体をダウングレードする手もあります。

2.6.4 依存関係の欠落により失敗したインストール

依存パッケージをインストールせず"`sudo dpkg -i ...`" でパッケージを強制インストールした場合、パッケージのインストールは部分インストールとなり失敗します。

"`sudo dpkg -i ...`" を繰り返し使って全ての依存パッケージをインストールします。

```
# apt --fix-broken install
```

そして、全ての部分インストール済みパッケージを以下のコマンドで設定します。

```
# dpkg --configure -a
```

2.6.5 重複するファイルを持つ相異なるパッケージ

aptitude(8) や apt-get(8) 等の、アーカイブレベルのパッケージ管理システムはパッケージの依存関係を使って重複するファイルを持つファイルのインストールしようとさえしません (項 2.1.7 を参照下さい)。

パッケージメインテナによるエラーや、システム管理者による不整合な混合ソースのアーカイブの採用 (項 2.7.6 を参照下さい) があった場合には、パッケージ依存関係が誤って定義される事態が発生するかもしれません。そういう状況下で重複するファイルを持つパッケージを aptitude(8) や apt-get(8) を使ってインストールしようとすると、パッケージを展開する dpkg(1) は既存ファイルを上書きすることなく呼ばれたプログラムにエラーを確実に返します。



注意
第三者が作成したパッケージを使うと、root 権限で実行されるシステムに関して何でもできるメンテナスクリプトが実行されるので、システムが重大なリスクにさらされます。dpkg(1) はパッケージを展開するするさいに上書きする事を防止するだけです。

そのような壊れたインストール状況は、まず古い問題原因となっているパッケージ *`old-package`* を削除すれば回避できます。

```
$ sudo dpkg -P old-package
```

2.6.6 壊れたパッケージスクリプトの修正

パッケージスクリプト内のコマンドが何らかの理由でエラーを返しスクリプトがエラーで終了した場合には、パッケージ管理システムは動作を途中終了するので部分的にインストールされたパッケージのある状況が生まれます。パッケージがその削除スクリプト内にバグを持つ場合には、パッケージが削除不能になりうるので大変厄介です。

”パッケージ名” のパッケージスクリプトの問題に関しては、以下のパッケージスクリプトの内容を確認するべきです。

- ”`/var/lib/dpkg/info/`パッケージ名`.preinst`”
- ”`/var/lib/dpkg/info/`パッケージ名`.postinst`”
- `/var/lib/dpkg/info/`パッケージ名`.prerm`
- ”`/var/lib/dpkg/info/`パッケージ名`.prerm`”

スクリプトの問題原因部分を以下のようなテクニックを使い root から編集します。

- 行頭に”#” を挿入し問題行を無効にします
- 行末に”`|| true`” を挿入し強制的に成功をリターンします

そして、全ての部分インストール済みパッケージを以下のコマンドで設定します。

```
# dpkg --configure -a
```

2.6.7 パッケージセレクションの復元

もし何らかの理由で”/var/lib/dpkg/status” の内容が腐った場合には、Debian システムはパッケージ選択データーが失われ大きな打撃を被ります。古い”/var/lib/dpkg/status” ファイルは、”/var/lib/dpkg/status-old” や”/var/backups/dpkg.status.*” としてあるので探します。

”/var/backups/” は多くの重要な情報を保持しているので、これを別のパーティション上に置くのも良い考えです。

ひどく壊れた場合には、システムのバックアップをした後フレッシュに再インストールすることをお薦めします。たとえ”/var/” ディレクトリーの中が完全に消去されても、”/usr/share/doc/” ディレクトリー中から新規インストールのガイドとなる情報を復元できます。

最低限の (デスクトップ) システムを再インストールします。

```
# mkdir -p /path/to/old/system
```

”/path/to/old/system/” に古いシステムをマウントします。

```
# cd /path/to/old/system/usr/share/doc
# ls -1 >~/ls1.txt
# cd /usr/share/doc
# ls -1 >>~/ls1.txt
# cd
# sort ls1.txt | uniq | less
```

こうすると、インストールすべきパッケージ名が表示されます。(”texmf” のようなパッケージ名以外が一部あるかもしれません。)

2.7 パッケージ管理のヒント

簡単のため、本セクション中のソースリスト例は trixie リリース後の 1 行スタイルの”/etc/apt/sources.list” として示されています。

2.7.1 誰がパッケージをアップロードしたのか?

”/var/lib/dpkg/available” や”/usr/share/doc/package_name/changelog” の中にリストされたメンテナの名前は” 誰がパッケージ化活動の背後にいるのか” に関していくばくかの情報を提供しますが、パッケージを実際にアップロードをした人がはっきりしません。devscripts パッケージ中の who-uploads(1) は Debian のソースパッケージを実際にアップロードした人を確定します。

2.7.2 APT のよるダウンロードバンド幅の制限

APT によるダウンロードのバンド幅を例えば 800Kib/sec (=100kiB/sec) に制限したい場合には、APT のパラメーターを以下のように設定します。

```
APT::Acquire::http::Dl-Limit "800";
```

2.7.3 パッケージの自動ダウンロードとアップグレード

apt パッケージには、パッケージの自動ダウンロードのサポートする専用の cron スクリプト"/etc/cron.daily/apt"が同梱されています。このスクリプトは unattended-upgrades パッケージをインストールすることで自動アップグレード実行の機能拡張をします。これらは、"/usr/share/doc/unattended-upgrades/README" に記述されているように、"/etc/apt/apt.conf.d/02backup" と"/etc/apt/apt.conf.d/50unattended-upgrades"の中のパラメーターでカスタム化できます。

unattended-upgrades パッケージは基本的に stable システムのセキュリティーアップグレードのためです。既存の stable システムが、自動アップグレードで壊される危険性が、セキュリティーアップグレードがすでに閉じたセキュリティーホールからの侵入者によりシステムが壊わされる危険性より小さいなら、パラメーターを以下のように設定して自動アップグレードをするのも一計です。

```
APT::Periodic::Update-Package-Lists "1";
APT::Periodic::Download-Upgradeable-Packages "1";
APT::Periodic::Unattended-Upgrade "1";
```

テスト版 testing や非安定版 unstable システムを実行する場合には、自動アップグレードするとシステムはいつの日か確実に壊われるので、それはしたくないでしょう。そんなテスト版 testing や非安定版 unstable の場合でも、次に記すような事前にパッケージをダウンロードするパラメーターを設定でインタラクティブなアップグレードをするための時間を節約したいでしょう。

```
APT::Periodic::Update-Package-Lists "1";
APT::Periodic::Download-Upgradeable-Packages "1";
APT::Periodic::Unattended-Upgrade "0";
```

2.7.4 Updates と Backports

stable のためのアップグレードパッケージを提供する、stable-updates (安定版 stable リリースが trixie の期間中の"trixie-updates") や backports.debian.org アーカイブがあります。

これらのアーカイブを使うには、以下に示すように"/etc/apt/preferences" ファイル中に全ての必要なアーカイブをリストします。

```
deb http://deb.debian.org/debian/ trixie main non-free-firmware contrib non-free
deb http://security.debian.org/debian-security trixie-security main non-free-firmware ↩
    contrib non-free
deb http://deb.debian.org/debian/ trixie-updates main non-free-firmware contrib non-free
deb http://deb.debian.org/debian/ trixie-backports main non-free-firmware contrib non-free
```

"/etc/apt/preferences" ファイル中に Pin-Priority 値を明示的に設定する必要はありません。より新しいパッケージが利用可能となった場合はいつも、デフォルトの設定によりもっとも合理的なアップグレードがなされます (項 2.5.3 参照下さい)。

- 全てのインストールされている古いパッケージが trixie-updates からのより新しいパッケージにアップグレードされます。
- trixie-backports からインストールしされた古いパッケージのみが trixie-backports からのより新しいパッケージにアップグレードされます。

"*package-name*" という名前のパッケージをその依存関係ともども trixie-backports アーカイブからインストールしたい時には、"-t" オプションでターゲットリリースを切り替えながら以下のコマンドを使います。

```
$ sudo apt-get install -t trixie-backports package-name
```



警告
backports.debian.org アーカイブから多すぎるパッケージをインストールしてはいけない。そんなことをするとパッケージ依存関係合併症を引き起こすかもしれません。代替解決策は、項 2.1.11 を参照下さい。

2.7.5 外部のパッケージアーカイブ



警告
外部のパッケージはあなたのシステムのルート権限を獲得することを意識するべきです。信頼できる外部のパッケージアーカイブのみを使うべきです。代替策は項 2.1.11 を参照下さい。

ソースリストに Debian コンパチブルな外部パッケージアーカイブを加え"`/etc/apt/trusted.gpg.d/`" ディレクトリーそのアーカイブキーファイルを置くことで、セキュアー APT でそのアーカイブが使えます。sources.list(5) と apt-secure(8) を参照下さい。

2.7.6 **apt-pinning** を使わない混合のアーカイブソースからのパッケージ



注意
安定版 (`stable`) と security updates と stable-updates のような公式にサポートされた特定の組み合わせ以外は、混合したアーカイブソースからのパッケージをインストールすることを、公式には Debian ディストリビューションとしてサポートしていません。

`testing` を追跡しながら、`unstable` にある特定の新規アップストリームバージョンのパッケージを 1 回だけ取り入れる操作例を次に示します。

1. "`/etc/apt/sources.list`" ファイルを変更し、単一の"`unstable`" エントリーのみにします。
2. "`aptitude update`" を実行します。
3. "`aptitude install パッケージ名`" の実行します。
4. `testing` のためのオリジナルの"`/etc/apt/sources.list`" ファイルを復元します。
5. "`aptitude update`" を実行します。

この様な手動のアプローチをすると"`/etc/apt/preferences`" ファイルを作ることもないし、また **apt-pinning** について悩むこともありません。でもこれではとても面倒です。



注意
混合したアーカイブソースを使うことを Debian が保証していないので、その場合にはパッケージ間の互換性は自分自身で確保しなければいけません。もしパッケージに互換性がないと、システムを壊すことになるかもしれません。この様な技術的要件を判断できる必要があります。ランダムな混合したアーカイブソースを使うことは全く任意の操作ですが、私としてはこの操作はお薦めできません。

異なるアーカイブからパッケージをインストールするための一般ルールは以下です。

- 非バイナリーパッケージ ("`Architecture: all`") のインストールは比較的安全です。
  - 文書パッケージ: 特段の要件無し
  - インタープリタプログラムパッケージ: 互換性あるインタープリタ環境が利用可能
- バイナリーパッケージ (非"`Architecture: all`") のインストールは、通常多くの障害があり、安全ではありません。
  - ライブラリー ("`libc`" 等) のバージョン互換性
  - 関連ユーティリティープログラムのバージョン互換性

- カーネル ABI 互換性
- C++ の ABI 互換性
- …

注意
パッケージを比較的安全にインストールできるようにするために、一部の商用 non-free バイナリープログラムパッケージは完全に静的にリンクされたライブラリーとともに提供される事があります。そんなパッケージに関しても ABI 互換性等の問題は確認するべきです。

注意
短期的に壊れたパッケージを回避するため以外は、非 Debian アーカイブからバイナリーパッケージをインストールするのは大体良くないアイデアです。あなたの現行の Debian システムとコンパチブルな、全ての利用可能な安全な技術的解決策を探索すべきです (項 2.1.11 を参照下さい)。

2.7.7 apt-pinning で候補バージョンを調整



警告
初心者ユーザーによる **apt-pinning** テクニックの利用は大トラブル発生を間違いなく起こします。絶対必要な時以外はこのテクニックの利用を避けなければいけません。代替解決策は、項 2.1.11 を参照下さい。

”`/etc/apt/preferences`” ファイル無しだと、APT システムはバージョン文字列を用いて、最新利用可能バージョンを候補バージョンとします。これが普通状態で APT システムの最も推薦される使い方です。全ての公式にサポートされたアーカイブの組み合わせは、自動的にアップグレードするソースとすべきでないアーカイブは **NotAutomatic** とマークされ適正な扱いを受けるので、”`/etc/apt/preferences`” ファイルを必要としません。

ティップ
バージョン文字列比較ルールは、例えば”`dpkg --compare-versions ver1.1 gt ver1.1~1; echo $?`” とすれば確認できます (dpkg(1) 参照下さい)。

パッケージを混合したアーカイブからのソース (項 2.7.6 を参照下さい) から定常的にインストールする場合には、apt_preferences(5) に書かれたように適正な項目のある”`/etc/apt/preferences`” ファイルを作り候補バージョンに関するパッケージ選択ルールを操作することによってこういった複雑な操作を自動化できます。これを **apt-pinning** と呼びます。

apt-pinning を利用する際には、Debian はパッケージの互換性を保証しないので、ユーザー自身がパッケージの互換性を確保しなければいけません。**apt-pinning** は全くの随意操作で、使用を勧めているわけではありません。

アーカイブレベルの Release ファイル (項 2.5.3 を参照下さい) が apt_preferences(5) のルールに使われます。だから、**apt-pinning** は normal Debian archives や security Debian archives ではスイート (”suite”) 名を使って機能します。(これは Ubuntu アーカイブとは異なります)。例えば”`/etc/apt/preferences`” ファイル中で、”`Pin: release a=unstable`” とはできますが、”`Pin: release a=sid`” とはできません。

非 Debian アーカイブを **apt-pinning** の一部に使う場合には、それが提供されている対象の確認とその信頼性の確認をします。例えば、Ubuntu と Debian は混合して使うようにはなっていません。

注意
”`/etc/apt/preferences`” ファイルを作成することなしでも、かなり複雑なシステム操作 (項 2.6.3 と項 2.7.6 を参照下さい) が **apt-pinning** を使わずにできます。

単純化した **apt-pinning** テクニックの説明を次にします。

APT システムは”`/etc/apt/sources.list`” ファイル中に規定された利用可能なパッケージソースから最高の Pin-Priority でアップグレードするパッケージを候補バージョンパッケージとして選択します。パッケージの Pin-Priority が 1000 より大きい場合には、このアップグレードするというバージョン制約が外れるのでダウングレードできるようになります (項 2.7.11 を参照下さい)。

各パッケージの Pin-Priority 値は”`/etc/apt/preferences`” ファイル中の”Pin-Priority” 項目にて規定されるか、そのデフォールト値が使われます。

Pin-Priority	パッケージに関する **apt-pinning** 効果
1001	パッケージのダウングレードになる場合でもパッケージをインストールする
990	ターゲットのリリースアーカイブのデフォルトとして使用
500	ノーマルアーカイブのデフォルトとして使用
100	**NotAutomatic** かつ **ButAutomaticUpgrades** アーカイブのデフォルトとして使用
100	インストール済みパッケージに使用
1	**NotAutomatic** アーカイブのデフォルトとして使用
-1	たとえ推奨 (Recommend) されても、パッケージを絶対にインストールしない

Table 2.18: **apt-pinning** テクニックに関する特記すべき Pin-Priority 値をリストします。

ターゲットのリリースアーカイブは”`apt-get install -t testing some-package`” のようにコマンドラインオプションによって設定できます。

アーカイブ中のアーカイブレベルの Release ファイル (項 2.5.3 を参照下さい) に”`NotAutomatic: yes`” と”`ButAutomat yes`” が含まれると **NotAutomatic** かつ **ButAutomaticUpgrades** アーカイブが設定されます。アーカイブ中のアーカイブレベルの Release ファイルに”`NotAutomatic: yes`” だけが含まれると **NotAutomatic** アーカイブが設定されます。

複数アーカイブソースの *package* に関する **apt-pinning** 状況は”`apt-cache policy` *package*” の出力で表示されます。

- ”`Package pin:`” で始まる行は、*package* のみとの関連付けが”`Package pin: 0.190`” 等と定義されている場合に、**pin** のパッケージバージョンを示します。
- *package* とのみの関連付けが定義されていない場合には、”`Package pin:`” という行はありません。
- *package* とのみの関連付けが定義されている場合の Pin-Priority 値は、全バージョン文字列の右側に”`0.181 700`” 等としてリストされます。
- *package* とのみの関連付けが定義されていない場合には、全バージョン文字列の右側に”`0`” が”`0.181 0`” 等としてリストされます。
- アーカイブの Pin-Priority 値 (”`/etc/apt/preferences`” ファイル中に”`Package: *`” として定義) はアーカイブへのパスの左側に、”`100 http://deb.debian.org/debian/ trixie-backports/main Packages`” 等としてリストされます。

2.7.8 ” 推奨 (Recommends)” によりパッケージがインストールされるのを阻止



警告
初心者ユーザーによる **apt-pinning** テクニックの利用は大トラブル発生を間違いなく起こします。絶対必要な時以外はこのテクニックの利用を避けなければいけません。代替解決策は、項 2.1.11 を参照下さい。

たとえ” 推奨 (Recommends)” されていても自動的に特定のパッケージが引きこまれ無くしたいときには、”`/etc/apt/preferences`” ファイルを作成しその中に全てのパッケージを以下のように明示的にリストしなければいけません。

```
Package: package-1
Pin: version *
Pin-Priority: -1

Package: package-2
Pin: version *
Pin-Priority: -1
```

2.7.9 `unstable` からのパッケージと共に、`testing` を追いかける



警告
初心者ユーザーによる **apt-pinning** テクニックの利用は大トラブル発生を間違いなく起こします。絶対必要な時以外はこのテクニックの利用を避けなければいけません。代替解決策は、項 2.1.11 を参照下さい。

`testing` を追跡しながら、`unstable` にある特定の新規アップストリームバージョンのパッケージが定常的にアップグレードされる、**apt-pinning** テクニックの例を次に示します。全ての必要なアーカイブを”`/etc/apt/sources.list`” ファイル中に以下のようにリストします。

```
deb http://deb.debian.org/debian/ testing main contrib non-free
deb http://deb.debian.org/debian/ unstable main contrib non-free
deb http://security.debian.org/debian-security testing-security main contrib
```

”`/etc/apt/preferences`” を以下のように設定します。

```
Package: *
Pin: release a=unstable
Pin-Priority: 100
```

”*package-name*” という名前のパッケージとその依存ファイルを `unstable` アーカイブからこの設定の下でインストールしたい場合、”`-t`” オプションを使ってターゲットのリリースを切り替える (`unstable` の Pin-Priority が 990 になる) 以下のコマンドを実行します。

```
$ sudo apt-get install -t unstable package-name
```

この設定では、通常の”`apt-get upgrade`” や”`apt-get dist-upgrade`” (”`aptitude safe-upgrade`” や”`aptitude full-upgrade`”) の実行は、`testing` アーカイブからインストールされたパッケージは最新の `testing` アーカイブを使ってアップグレードし、`unstable` アーカイブからインストールされたパッケージは最新の `unstable` アーカイブを使ってアップグレードします。



注意
”`/etc/apt/sources.list`” ファイルから”`testing`” の項目を削除しないように注意します。”`testing`” 項目がその中にないと、APT システムは最新の `unstable` アーカイブを使ってアップグレードします。

ティップ
著者は上記操作のすぐ後に”`/etc/apt/sources.list`” ファイルを通常編集して”`unstable`” アーカイブ項目をコメントアウトします。こうすることで、最新の `unstable` アーカイブによって `unstable` からインストールされたパッケージをアップグレードしなくなりますが、”`/etc/apt/sources.list`” ファイル中に項目が多すぎて更新のプロセスが遅くなることを避けられます。

ティップ
もし”/etc/apt/preferences” ファイル中で”Pin-Priority: 100” の代わりに”Pin-Priority: 1” が用いられた場合は、”/etc/apt/sources.list” ファイルの中の”testing” 項目が削除されようと、Pin-Priority 値は 100 のインストール済みパッケージは unstable アーカイブによってアップグレードされる事はありません。

最初の”-t unstable” によるインストール無しに、unstable の特定パッケージを自動的に追跡したい場合、”/etc/apt/preferences” ファイルを作りそのトップにこれらパッケージを明示的に以下のようにリストします。

```
Package: package-1
Pin: release a=unstable
Pin-Priority: 700

Package: package-2
Pin: release a=unstable
Pin-Priority: 700
```

以上で、各特定パッケージに関して Pin-Priority 値が設定されます。例えば最新の unstable バージョンのこの”Debian リファレンス” を英語版で追跡するためには、”/etc/apt/preferences” ファイルに以下の項目を設定します。

```
Package: debian-reference-en
Pin: release a=unstable
Pin-Priority: 700

Package: debian-reference-common
Pin: release a=unstable
Pin-Priority: 700
```

ティップ
この **apt-pinning** テクニックは stable アーカイブを追跡している際にも有効です。著者の経験では、文書パッケージは unstable アーカイブからインストールしても今までいつも安全でした。

2.7.10 `experimental` からのパッケージと共に、`unstable` を追いかける



警告
初心者ユーザーによる **apt-pinning** テクニックの利用は大トラブル発生を間違いなく起こします。絶対必要な時以外はこのテクニックの利用を避けなければいけません。代替解決策は、項 2.1.11 を参照下さい。

次に unstable を追跡しながら experimental にある特定の新規アップストリームバージョンのパッケージを取り込む **apt-pinning** テクニックの例を示します。すべての必要なアーカイブを”/etc/apt/sources.list” ファイルに以下のようにリストします。

```
deb http://deb.debian.org/debian/ unstable main contrib non-free
deb http://deb.debian.org/debian/ experimental main contrib non-free
deb http://security.debian.org/ testing-security main contrib
```

experimental アーカイブのデフォールトの Pin-Priority 値は、**NotAutomatic** アーカイブ (項 2.5.3 を参照下さい) なので、常に 1 (<<100) です。次回アップグレード時に experimental アーカイブにある特定パッケージを自動的に追跡しようとしない限り、”/etc/apt/preferences” ファイル中で Pin-Priority 値をそれを明示的に設定する必要はありません。

2.7.11 緊急ダウングレード



警告
初心者ユーザーによる **apt-pinning** テクニックの利用は大トラブル発生を間違いなく起こします。絶対必要な時以外はこのテクニックの利用を避けなければいけません。代替解決策は、項 2.1.11 を参照下さい。



注意
Debian では設計としてはダウングレードを正式にサポートしません。緊急の復元処置の一部としてのみ実行されるべきです。こういう状況であるにもかかわらず、多くの場合にうまく機能することが知られています。重要なシステムでは復元操作の後に全ての重要データーをバックアップし、最初から新規システムを再インストールします。

壊れたシステムアップグレードからの復元するために、候補バージョンを操作して新しいアーカイブから古いアーカイブにダウングレードすることがうまくいくかもしれません (項 2.7.7 を参照下さい)。これは、何度も"`dpkg -i broken-package_old-version.deb`" コマンドを実行する退屈な作業をしないでよくする方法です (項 2.6.3 を参照下さい)。

次に記すような"`unstable`" を追跡する"`/etc/apt/sources.list`" ファイル中の行を探します。

```
deb http://deb.debian.org/debian/ sid main contrib non-free
```

それを `testing` を追いかけるように次と交換します。

```
deb http://deb.debian.org/debian/ forky main contrib non-free
```

"`/etc/apt/preferences`" を以下のように設定します。

```
Package: *
Pin: release a=testing
Pin-Priority: 1010
```

"`apt-get dist-upgrade`" を実行して、システム全体にわたってパッケージのダウングレードを強制します。

この緊急ダウングレードの後でこの特別の"`/etc/apt/preferences`" ファイルを削除します。

ティップ
依存関係の問題を最小限とすべく、できるだけ多くのパッケージを削除 (remove で、完全削除 purge ではありません!) します。システムのダウングレードのためには手動でいくつかのパッケージを削除とインストールしなければいけないかも知れません。Linux カーネルやブートローダーや udev や PAM や APT やネットワーク関係のパッケージやそれらの設定ファイルには特に注意が必要です。

2.7.12 **equivs** パッケージ

ソースからプログラムをコンパイルして Debian パッケージを置換えたい際には、それを実際にローカルで Debian 化してパッケージ (`*.deb`) して、私的アーカイブを使うのが好ましいです。

しかし、プログラムをソースからコンパイルして"`/usr/local`" にインストールすることを選んだ際には、パッケージ依存関係を満足させるための最後の手段として `equivs` を使う必要があるかもしれません。

```
Package: equivs
Priority: optional
Section: admin
Description: Circumventing Debian package dependencies
```

```
This package provides a tool to create trivial Debian packages.
Typically these packages contain only dependency information, but they
can also include normal installed files like other packages do.
.
One use for this is to create a metapackage: a package whose sole
purpose is to declare dependencies and conflicts on other packages so
that these will be automatically installed, upgraded, or removed.
.
Another use is to circumvent dependency checking: by letting dpkg
think a particular package name and version is installed when it
isn't, you can work around bugs in other packages' dependencies.
(Please do still file such bugs, though.)
```

2.7.13 安定版システムへのパッケージ移植



注意
ここに書かれた手順がシステムの違いのために追加の手動の努力無しにうまく行く保証はありません。

stable システムの部分アップグレードのためには、その環境内でソースパッケージを使ってパッケージをリビルドするのが好ましいです。こうすることでパッケージ依存関係による大掛かりなアップグレードをしないで済みます。

stable システムのための”/etc/apt/sources.list” ファイルに以下のエントリーを追加します。

```
deb-src http://deb.debian.org/debian unstable  main contrib non-free
```

コンパイルするのに必要なパッケージをインストールしソースパッケージをダウンロードをします。

```
# apt-get update
# apt-get dist-upgrade
# apt-get install fakeroot devscripts build-essential
# apt-get build-dep foo
$ apt-get source foo
$ cd foo*
```

バックポートに必要な際には、dpkg や debhelper 等のツールチェインパッケージをバックポートパッケージを用いて更新します。

次を実行します。

```
$ dch -i
```

”+bp1” を後ろに付けるなどして、”debian/changelog” 中でパッケージバージョンを先に進める

以下のようにしてパッケージをビルドしシステムにインストールします。

```
$ debuild
$ cd ..
# debi foo*.changes
```

2.7.14 APT のためのプロキシサーバー

Debian アーカイブの特定サブセクション全てをミラーするとディスク空間とネットワークのバンド幅の大いなる無駄遣いですので、LAN 上に多くのシステムを管理している際には APT のためのローカルのプロキシサーバーを設

置することを考えるのは良いことです。APT は、apt.conf(5) とか”/usr/share/doc/apt/examples/configure-index に説明されたようにして、汎用の squid のようなウェッブ (http) プロキシサーバー (項 6.5 を参照下さい) を使うように設定できます。”$http_proxy” 環境変数による設定は、”/etc/apt/apt.conf” ファイル中の設定より優先します。

Debian アーカイブ専用のプロキシツールがあります。実際に使う前に BTS をチェック下さい。

パッケージ	ポプコン	サイズ	説明
apt-cacher	V:0.27, I:0.32	265	Debian パッケージとソースファイルのキャッシュプロキシ (Perl プログラム)
apt-cacher-ng	V:3.8, I:3.9	2063	ソフトウェアーパッケージの頒布ためのキャッシュプロキシ (コンパイルされた C++ プログラム)

Table 2.19: Debian アーカイブ専用のプロキシツールのリスト



注意
Debian がそのアーカイブ構造を再編した際に、このような専用のプロキシツールはパッケージメンテナによるコードの修正が必要で、一定期間使えなくなることがあります。一方、汎用のウェッブ (http) プロキシは比較的堅牢ですしそのような変化に合わすのも簡単です。

2.7.15 パッケージ管理の追加参考文書

パッケージ管理に関しては以下の文書からさらに学習できます。

- パッケージ管理の一義的文書:
  - aptitude(8) と dpkg(1) と tasksel(8) と apt(8) と apt-get(8) と apt-config(8) と apt-secure(8) と sources.list(5) と apt.conf(5) と apt_preferences(5);
  - apt-doc パッケージの”/usr/share/doc/apt-doc/guide.html/index.html” と”/usr/share/doc/apt-doc
  - aptitude-doc-en パッケージに入っている、”/usr/share/doc/aptitude/html/en/index.html”。
- 正規で詳細な Debian アーカイブに関する文書:
  - Debian アーカイブの正式のポリシーは Debian ポリシーマニュアル、第 2 章 - Debian アーカイブに規定されています。
  - ”Debian 開発者リファレンス、第 4 章 Debian 開発者が利用可能なリソース 4.6 Debian アーカイブ” と、
  - ”The Debian GNU/Linux FAQ, Chapter 6 - The Debian FTP archives”.
- Debian ユーザー向けの Debian パッケージ作成の入門書:
  - ”Debian Maintainer 向け案内書”.

Chapter 3

システムの初期化

Debian システムが如何に起動され設定されるかを知っていることはシステム管理者として賢明です。正確で詳細な情報がインストール済みパッケージのソースや文書中にあるとは言え、我々の大部分にとってはちょっと大変過ぎます。

Debian システム初期化の大まかな概要をここに記します。Debian システムは動くターゲットなので、最新のドキュメンテーションを参照する必要があります。

- Debian Linux Kernel Handbook は Debian カーネルに関する一次情報源です。
- `systemd` に準拠するシステムのブートアッププロセスは bootup(7) に詳述されている。(最新の Debian)
- UNIX System V Release 4 に準拠するシステムのブートアッププロセスは boot(7) に詳述されている。(過去の Debian)

3.1 ブートストラッププロセスの概要

コンピューターシステムは、電源投入イベントからユーザーに機能の完備したオペレーティングシステム (OS) を提供するまでブートストラッププロセスを数段通過します。

単純化のため、デフォールトのインストールをした典型的な PC プラットフォームに限定し議論します。

典型的なブートストラッププロセスは 4 段ロケットのようです。各段のロケットは次段のロケットにシステムのコントロールを引き継ぎます。

- 項 3.1.1
- 項 3.1.2
- 項 3.1.3
- 項 3.1.4

もちろん、これらに関して異なる設定をすることはできます。例えば、自分自身で専用カーネルをコンパイルした場合、ミニ Debian システムのステップをスキップできます。自分自身で確認するまでは、あなたのシステムがこの様になっていると決めつけないで下さい。

3.1.1 1 段目: UEFI

Unified Extensible Firmware Interface (UEFI) は、UEFI 仕様書の一部としてブートマネージャーを定義します。ブートマナージャーは、コンピューター電源投入時にブート設定をチェックしその設定に基づき指定された OS ブートローダーかオペレーティングシステムのカーネル (通常ブートローダー) を実行するブートプロセスの第一段階です。ブート設定は、OS ローダーや OS カーネルまでのファイルシステムパスを示す変数を含めて、NVRAM に保存された変数に定義されています。

EFI システムパーティション (ESP) は UEFI 仕様書に準拠するコンピューターで使われるデーターストレージデバイスのパーティションです。それはコンピューター電源投入時に UEFI ファームウエアーによってアクセスされ、UEFI アプリケーションやそれらのアプリケーションが実行する必要のある OS ブートローダーを含めたファイルが保存されています。(旧式の PC システム上では MBR 中に保存された BIOS がこれに代え使われているかもしれません)

3.1.2 2 段目: ブートローダー

ブートローダーは UEFI によって起動されるブートプロセスの 2 段目です。それはシステムのカーネルイメージと initrd イメージをメモリーにロードし、それらにコントロールを引き継ぎます。この initrd イメージはルートファイルシステムイメージで、そのサポートは使われるブートローダーによります。

Debian システムは普通 Linux カーネルをデフォルトのシステムカーネルとして使います。現行の 5.x Linux カーネルのための initrd イメージは技術的には initramfs (initial RAM filesystem) イメージです。

多くのブートローダと設定オプションが利用可能です。

パッケージ	ポプコン	サイズ	**initrd**	ブートローダー	説明
grub-efi-amd64	I:472	142	サポート	GRUB UEFI	ディスクパーティションや vfat や ext4 等のファイルシステムを理解するぐらいスマートです。(UEFI)
grub-pc	V:17, I:504	479	サポート	GRUB 2	ディスクパーティションや vfat や ext4 等のファイルシステムを理解するぐらいスマートです。(BIOS)
grub-rescue-pc	V:0.04, I:0.55	7269	サポート	GRUB 2	GRUB 2 のブート可能なレスキューイメージ (CD とフロッピー) (PC/BIOS バージョン)
grml-rescueboot	V:0.1, I:1.1	39	N/A	GRUB 2 プラグイン	grml-rescueboot は、live Linux ISO image を grub2 のブートメニューに追加します。
syslinux	V:3, I:30	325	サポート	Isolinux	ISO9660 ファイルシステムを理解します。ブート CD に使われています。
syslinux	V:3, I:30	325	サポート	Syslinux	MSDOS ファイルシステム (FAT) 理解します。ブートフロッピーで使われます。
loadlin	V:0.02, I:0.36	87	サポート	Loadlin	新しいシステムが FreeDOS/MSDOS システムから起動されます。
mbr	V:0.2, I:2.6	47	非サポート	Neil Turton の MBR	MSDOS の MBR を代替するフリーソフトウェアーです。ディスクパーティションを理解するだけです。

Table 3.1: ブートローダーのリスト

UEFI システムの場合、GRUB2 は ESP パーティションを最初に読み、"`/boot/efi/EFI/debian/grub.cfg`" 中の `search.fs_uuid` に指定された UUID を使って GRUB2 メニュー設定ファイル"`/boot/grub/grub.cfg`" があるパーティションを決めます。

GRUB2 メニュー設定ファイルの重要部分は以下です:

```
menuentry 'Debian GNU/Linux' ... {
        load_video
        insmod gzio
        insmod part_gpt
        insmod ext2
        search --no-floppy --fs-uuid --set=root fe3e1db5-6454-46d6-a14c-071208ebe4b1
        echo    'Loading Linux 5.10.0-6-amd64 ...'
        linux   /boot/vmlinuz-5.10.0-6-amd64 root=UUID=fe3e1db5-6454-46d6-a14c-071208ebe4b1 ↩
            ro quiet
        echo    'Loading initial ramdisk ...'
        initrd  /boot/initrd.img-5.10.0-6-amd64
}
```

`/boot/grub/grub.cfg` のこの部分に関して、このメニューエントリーは以下の意味があります。

設定	変数値
導入済みの GRUB2 モジュール	`gzio`, `part_gpt`, `ext2`
使用されるルートファイルシステムのパーティション	`UUID=fe3e1db5-6454-46d6-a14c-071208ebe4b1` により識別されるパーティション
ルートファイルシステム中のカーネルイメージのパス	`/boot/vmlinuz-5.10.0-6-amd64`
使用されるカーネルブート変数	"`root=UUID=fe3e1db5-6454-46d6-a14c-071208ebe4b1 ro quiet`"
ルートファイルシステム中の initrd イメージパス	`/boot/initrd.img-5.10.0-6-amd64`

Table 3.2: `/boot/grub/grub.cfg` の上記部分のメニューエントリーの意味

Debian システム上では、"`/boot/grub/grub.cfg`" はインストールされた GRUB パッケージ (例えば、`grub-efi-amd64`) で管理されていて、ユーザーによる直接の改変は非推奨です。その代わり、"`/etc/grub.d/`" 中の設定ファイルや"`/etc/default/grub`" をカスタマイズしましょう。そして、GRUB 設定ファイルを設定し、NVRAM 変数を自動的に Debian をブートするように以下のようにします:

```
# dpkg-reconfigure grub-efi-amd64
```

ティップ
カーネルブートログを見えるようにするには、"`/etc/default/grub`" 中の"`GRUB_CMDLINE_LINUX_DEFAULT`" の値から `quiet` を除去して下さい。

ティップ
You can add GRUB splash background by placing its image file in 画像ファイルを"`/boot/grub/`" 中に置くことで、GRUB に画面背景を追加できます。

"`info grub`", grub-install(8), grub-mkconfig(8) を参照下さい。

3.1.3 3 段目: ミニ Debian システム

ミニ Debian システムはブートローダーによって起動されるブートプロセスの 3 段目です。メモリー上でルートファイルシステムとともにシステムカーネルを実行します。これはオプションのブートプロセスの準備段階です。

注意
” ミニ Debian システム” は著者がこの 3 段目のブートプロセスを本文書中で記述するために作った言葉です。このシステムは一般に initrd とか initramfs システムと呼ばれています。類似のメモリー上のシステムは Debian インストーラーでも使われています。

”/init” スクリプトはこのメモリー上のルートファイルシステムで最初に実行されるプログラムです。それはユーザー空間でカーネルを初期化しコントロールを次の段階に引き継ぐプログラムです。このミニ Debian システムは、メインのブートプロセスが始まる前にカーネルモジュールを追加したり、ルートファイルシステムを暗号化されたファイルシステムとしてマウントする等のブートプロセスの柔軟性を提供します。

- `initramfs-tools` で initramfs が作成された場合には”`/init`” プログラムはシェルプログラムです。
  - ”`break=init`” 等をカーネルブートパラメーターとして与えると、本部分のブートプロセスに割り込み root シェルを獲得できます。この他の割り込み条件は”`/init`” スクリプトを参照下さい。このシェル環境はあなたの機器のハードウエアーを詳細に検査できるだけ十分洗練されています。
  - このミニ Debian システムで利用可能なコマンドは機能を削ったコマンドで、主に busybox(1) という GNU ツールで提供されます。
- `dracut` で initramfs が作成された場合には”`/init`” プログラムはバイナリーの `systemd` プログラムです。
  - このミニ Debian システムで利用可能なコマンドは機能を削った systemd(1) 環境です。



注意
読出しのみのルートファイルシステム上では、`mount` コマンドには”`-n`” オプションを使う必要があります。

3.1.4 4 段目: 普通の Debian システム

通常の Debian システムはミニ Debian システムによって起動されるブートプロセスの 4 段目です。ミニ Debian システムのシステムカーネルはこの環境ででも実行され続けます。ルートファイルシステムはメモリー上から本当にハードディスク上にあるファイルシステムに切り替えられます。

多くのプログラムを起動する主ブートプロセスを行う init プログラムは、PID=1 で最初のプログラムとして実行されます。init プログラムのデフォールトのファイルパスは”`/usr/sbin/init`” ですが、”`init=/path/to/init_program`” のようなカーネルブートパラメーターにより変更できます。

Debian 8 jessie (2015 年リリース) 以降では”`/usr/sbin/init`” は”`/lib/systemd/systemd`” にシムリンクされています。

ティップ
あなたのシステム上の実際の init コマンドは”`ps --pid 1 -f`” コマンドで確認できます。

ティップ
ブートプロセスを高速化する最新のティップは Debian wiki: BootProcessSpeedup を参照下さい。

パッケージ	ポプコン	サイズ	説明
systemd	V:925, I:981	11401	並行処理のためのイベント依存の init(8) デーモン (sysvinit 代替)
cloud-init	V:4.6, I:6.4	3267	インフラストラクチャークラウドインスタンス用の初期化システム
systemd-sysv	V:921, I:984	77	systemd で sysvinit を置換するのに必要な、マニュアルページとリンク
init-system-helpers	V:754, I:989	132	sysvinit と systemd 間を切り替える補助ツール
initscripts	V:14, I:49	197	システムの始動と停止のためのスクリプト
sysvinit-core	V:3.1, I:3.5	368	System-V 的な init(8) ユーティリティー
sysv-rc	V:27, I:52	85	System-V 的なランレベル変更メカニズム
sysvinit-utils	V:734, I:1000	100	System-V 的なユーティリティー (startpar(1)、bootlogd(8)、…)
lsb-base	V:193, I:281	12	Linux Standard Base 3.2 の init スクリプト機能
insserv	V:32, I:52	132	LSB init.d スクリプト依存関係を使いブート順序を整理するツール
kexec-tools	V:1.6, I:5.3	320	kexec(8) リブートのための kexec ツール (ワームリブート)
systemd-bootchart	V:0.17, I:0.49	131	ブートプロセスのパーフォンマンスアナライザー
mingetty	V:0.1, I:2.3	36	コンソール専用 getty(8)
mgetty	V:0.13, I:0.44	318	インテリジェントモデム用の代替 getty(8)

Table 3.3: Debian システムのブートユーティリティーのリスト

3.2 レスキューシステム



警告
ブートストラッププロセス周辺のシステム管理作業をレスキューシステムを確保せず行ってはいけません。

レスキューシステムが使えることで、困難な作業に取り組めるようになります。

- 壊れたブートローダーのインストールからのブート
- 壊れたブートローダーのインストールの修復
- 壊れてブートできないシステムからのデーター抽出
- root ファイルシステムが絡む、ファイルシステムやディスクパーティションや LVM ボリュームの編集

一般的にはレスキューシステムは ISO イメージファイルで提供され、次のような脱着可能なストレージ媒体に書き込まれます:

- 項 9.7.2 のようにして準備される USB フラッシュドライブ
- 項 9.7.7 のようにして準備される CD / DVD

簡潔さのため、USB フラッシュドライブの場合が以下で例示し言及されているが、CD や DVD も同様に使えます。

ティップ
脱着可能なストレージ媒体上の不特定のブートローダーをブートするには、何らかの UEFI NVRAM 変数を変更しなければいけないかもしれません。

3.2.1 USB 上の GRUB UEFI レスキューシステム

「USB 上の GRUB UEFI レスキューシステム」を挿入しシステムの電源投入で、GRUB UEFI レスキューシステムが起動できます。

事前に Super Grub2 Disk ISO イメージを USB フラッシュドライブに書き込むことで、「USB 上の GRUB UEFI レスキューシステム」を準備できます。

他のオペーレーティングシステムの導入などでブートローダー設定が破壊された場合等は、以下のようにして修復します。

- GRUB UEFI レスキューシステムを起動し、自動的にブート可能なインストール済みシステムを発見します。
- インストール済み Debian システムを GRUB メニューから起動します。
- Linux の root シェルプロンプトで:

```
# dpkg-reconfigure grub-efi-amd64
```

注意
”`info grub-mkrescue`” に従うことでも、ブート可能な GRUB レスキュー ISO イメージが作成できます。これは、CLI の GRUB シェルプロンプトを提供しますが、ブート可能なシステムの自動発見は提供しません。

3.2.2 USB 上の Linux live レスキューシステム

Linux live レスキューシステムは「USB 上の Linux live レスキューシステム」を挿入してシステム電源を投入することで起動できます。

「USB 上の Linux live レスキューシステム」は、USB フラッシュドライブに Debian 準拠の Linux live ISO イメージの 1 つを事前に書き込むことで準備できます。そのような Linux live イメージの例は以下です。

- Debian Live イメージ
- Kali Linux Live
- Grml Live Linux

この Linux live レスキューシステムの使用例です:

- 他のオペーレーティングシステムの導入などで破壊されたブートローダー設定は以下のようにして修復します。
  - Linux live レスキューシステムをスタートします。
  - ブートできないインストールされたシステムの root ファイルシステムを含むパーティションを”/mnt” にマウントします。
  - Linux の root シェルプロンプトで:

```
# chroot /mnt; dpkg-reconfigure grub-efi-amd64
```

- 壊れた dpkg パッケージの修復:
  - Linux live レスキューシステムをスタートします。
  - 壊れた dpkg パッケージを含むインストールされた Debian システムの root ファイルシステムを含むパーティションを”/mnt” にマウントします。
  - Linux の root シェルプロンプトで:

```
# dpkg --root /mnt -i /mnt/var/cache/apt/archives/dpkg_old_version_amd64.deb
```

- インストール済みシステムへのファイルシステム操作 (項 9.6 参照) のような通常禁止されている変更の実行。



警告
無活動状態が続くと、Linux live システムの GUI スクリーンがロックされてしまうかもしれません。

ティップ

- Linux live システムをスタートしてすぐあなたのパスワードを設定しましょう。
- Linux 仮想コンソール (項 1.1.6 参照) 上のユーザーのシェルプロンプトから、例えば”sudo passwd *user*” としてあなたのパスワードを設定してもいいです。
- 一部の Linux live システムはデフォルトの”*user*/*password*” を設定します: ”Debian Live”: ”user/live”, ”Kali Linux Live”: ”kali/kali”

3.2.3 GRUB からの Linux live レスキューシステム

Linux live レスキューシステムは、GRUB メニュー項目から起動できます。この GRUB 設定は以下のようにすると準備できます:

- `grml-rescueboot` パッケージをインストールします。
- Linux live ISO イメージ中に"`/boot/grub/loopback.cfg`" があるイメージを探します。
  - 項 3.2.2 中で触れた ISO イメージは"`/boot/grub/loopback.cfg`" そのイメージ中に保持しています。
- "`/boot/grml/`" ディレクトリー中にこれらの Linux live ISO イメージの一部をコピーします。
- 以下のように GRUB メニューを更新します。

```
# dpkg-reconfigure grub-efi-amd64
```

電源を投入すると、GRUB メニューは Linux live レスキューシステムの候補を含むメニューを示します。

3.3 Systemd

3.3.1 Systemd init

Debian システムが起動する時、`/usr/lib/systemd` にシムリンクされた `/usr/sbin/init` が、root (`UID=0`) が所有する init システムプロセス (`PID=1`) で起動されます。systemd(1) を参照下さい。

`systemd` の init プロセスは、SysV 的な手続き定義スタイルではなく宣言定義スタイルで書かれた unit 設定ファイルに従い並列で複数プロセスを起動します。(systemd-system.conf(5) 参照):

生成されたプロセスは、それらが所属するプライベート systemd ヒエラルキー中の所属する unit にちなんで名付けられた個別の Linux コントロールグループ中に置かれます (cgroups と項 4.7.5 を参照下さい。)

systemd.unit(5) 中に記載された"System Unit Search Path (システムユニット探索パス)" からシステムモード用の unit がロードされます。主要なパスは優先順位順に以下です:

- "`/etc/systemd/system/*`": 管理者が作成したシステム unit
- "`/run/systemd/system/*`": 実行時の unit
- "`/lib/systemd/system/*`": ディストリビューションパッケージマネージャーがインストールしたシステム unit

相互依存関係は"`Wants=`"、"`Requires=`"、"`Before=`"、"`After=`"、…("MAPPING OF UNIT PROPERTIES TO THEIR INVERSES" in systemd.unit(5) 参照) 等の指示定義によって規定される。リソースのコントロールは (systemd.resource-control(5) 参照) によっても定義されます。

unit 設定ファイルのサッフィックスにそのタイプを以下のように折込みます:

- ***.service** は `systemd` がコントロールしたりスーパーバイズするプロセスを記述します。systemd.service(5) 参照下さい。
- ***.device** は sysfs(5) 中に udev(7) デバイスツリーとして暴露されるデバイスを記述します。See systemd.device(5) を参照下さい。
- ***.mount** は `systemd` がコントロールしたりスーパーバイズするファイルシステムのマウントポイントを記述します。systemd.mount(5) を参照下さい。
- ***.automount** は `systemd` がコントロールしたりスーパーバイズするファイルシステムの自動マウントポイントを記述します。systemd.automount(5) を参照下さい。

- ***.swap** は systemd がコントロールやスーパーバイズするスワップデバイスやファイルを記述します。systemd.swap(5) を参照下さい。
- ***.path** は systemd がパス基準でアクティベーションするために監視するパスを記述します。systemd.path(5) を参照下さい。
- ***.socket** は systemd がソケット基準でアクティベーションするためにコントロールしたりスーパーバイズするソケットを記述します。systemd.socket(5) を参照下さい。
- ***.timer** は systemd がタイマー基準でアクティベーションするためにコントロールしたりスーパーバイズするタイマーを記述します。systemd.timer(5) を参照下さい。
- ***.slice** は cgroups(7) でリソースを管理します。systemd.slice(5) を参照下さい。
- ***.scope** はシステムプロセスの集合を systemd のバスインターフェースを用いて管理するためにプログラムで作られます。systemd.scope(5) を参照下さい。
- ***.target** は他の unit 設定ファイルを組み合わせて始動時同期点を作ります。systemd.target(5) を参照下さい。

システムの始動 (init) されると systemd プロセスは (普通"`graphical.target`" にシムリンクされている) "`/lib/systemd/system/default.target` を起動しようとします。. 最初に、"`local-fs.target`" や"`swap.target`" や"`cryptsetup.target`" 等のいくつかの特殊ターゲット unit (systemd.special(7) 参照) が引き込まれファイルシステムをマウントします。そして、他のターゲット unit が、ターゲット unit の依存関係で引き込まれます。詳細に関しては bootup(7) を読んで下さい。

systemd はバックワードコンパティビリティー機能を提供します。"`/etc/init.d/rc[0123456S].d/[KS]`*name*" 中の、SysV-スタイルのブートスクリプトは依然として読み込まれ処理されますし、telinit(8) は systemd の unit のアクティベーション要求に変換されます。



注意
擬似実装された runlevel の 2 から 4 は、すべて同じ"`multi-user.target`" にシムリンクされます。

3.3.2 Systemd login

gdm3(8)、sshd(8)、等々経由で Debian システムにユーザーがログインする際に、当該ユーザーが所有するユーザーサービスマネージャープロセスとして `/lib/systemd/system --user` が起動されます。systemd(1) を参照下さい。

systemd のユーザーサービスマネージャーは、宣言定義スタイルで書かれた unit 設定ファイルに従い並列で複数プロセスを起動します。(systemd.unit(5) や `user@.service`(5) 参照):

systemd.unit(5) 中に記載された"User Unit Search Path (ユーザーユニット探索パス)" からユーザーモード用の unit がロードされます。主要なパスは優先順位順に以下です:

- "`~/.config/systemd/user/*`": ユーザー設定 unit
- "`/etc/systemd/user/*`": 管理者が作成したユーザー units
- "`/run/systemd/user/*`": 実行時 unit
- "`/lib/systemd/user/*`": ディストリビューションパッケージマネージャーがインストールしたユーザー unit

これらは、項 3.3.1 と同様の手法で管理されます。

エラーレベル値	エラーレベル名	意味
0	KERN_EMERG	システムは不安定
1	KERN_ALERT	直ぐアクションが必要
2	KERN_CRIT	クリチカルなコンディション
3	KERN_ERR	エラーコンディション
4	KERN_WARNING	警告コンディション
5	KERN_NOTICE	ノーマルだが重要なコンディション
6	KERN_INFO	情報
7	KERN_DEBUG	デバグレベルのメッセージ

Table 3.4: カーネルエラーレベルのリスト

3.4 カーネルメッセージ

コンソールに表示されるカーネルのエラーメッセージは、その閾値て設定できる。

```
# dmesg -n3
```

3.5 システムメッセージ

systemd の下では､カーネルとシステムの両方のメッセージがジャーナルサービス systemd-journald.service (所謂 journald) で、"/var/log/journal" の下の恒久的なバイナリーデーターか"/run/log/journal/" 下の非恒久的なバイナリーデーターとして記録されます。このようなバイナリー記録データーには journalctl(1) コマンドを用いてアクセスできます。例えば以下のようにして最後のブートからのログを表示できます:

```
$ journalctl -b
```

操作	コマンド断片
最後のブートからのシステムサービスとカーネルのログを閲覧	"journalctl -b --system"
最後のブートからの現行ユーザーのサービスのログを閲覧	"journalctl -b --user"
最後のブートからの"$unit" のジョブのログを閲覧	"journalctl -b -u $unit"
最後のブートからの"$unit" のジョブのログを見る ("tail -f" スタイル)	"journalctl -b -u $unit -f"

Table 3.5: 典型的な journalctl コマンド断片の例

systemd のもとでは、システムログユーティリティー rsyslogd(8) はアンインストールしても大丈夫です。もしそれがインストールされている場合には、それは (systemd 以前のデフォルトの"/dev/log" に代え) 揮発性のバイナリーログデーターを読むように挙動を変え、伝統的で恒久的な ASCII システムログデーターを作成します。これは、"/etc/default/rsyslog" や"/etc/rsyslog.conf" を用いてログファイルと画面表示の両方に関してカスタマイズ可能です。rsyslogd(8) や rsyslog.conf(5) を参照下さい。また、項 9.3.2 も参照下さい。

3.6 システム管理

systemd は init システムを提供するのみならず、systemctl(1) コマンドで汎用システム管理機能を提供します。

操作	コマンド断片
全使用可能なユニットタイプをリスト	"systemctl list-units --type=help"
メモリー中の全ターゲット unit のリスト	"systemctl list-units --type=target"
メモリー中の全サービス unit のリスト	"systemctl list-units --type=service"
メモリー中の全デバイス unit のリスト	"systemctl list-units --type=device"
メモリー中の全マウント unit のリスト	"systemctl list-units --type=mount"
メモリー中の全ソケット unit のリスト	"systemctl list-sockets"
メモリー中の全タイマー unit のリスト	"systemctl list-timers"
"$unit" 始動	"systemctl start $unit"
"$unit" 停止	"systemctl stop $unit"
サービス特定の設定の再ロード	"systemctl reload $unit"
"$unit" 停止と始動	"systemctl restart $unit"
"$unit" 始動と、他全ての停止	"systemctl isolate $unit"
"graphical" に切り替え (GUI システム)	"systemctl isolate graphical"
"multi-user" に切り替え (CLI システム)	"systemctl isolate multi-user"
"rescue" に切り替え (シングルユーザー CLI システム)	"systemctl isolate rescue"
"$unit" に kill 信号を送る	"systemctl kill $unit"
"$unit" サービスがアクティブかを確認	"systemctl is-active $unit"
"$unit" サービスが失敗かを確認	"systemctl is-failed $unit"
"$unit\|$PID|device" の状態を確認	"systemctl status $unit\|$PID|$device"
"$unit\|$job" の属性を表示	"systemctl show $unit\|$job"
失敗した"$unit" をリセット	"systemctl reset-failed $unit"
全ての unit サービスの依存関係をリスト	"systemctl list-dependencies --all"
システムにインストールされた unit ファイルをリスト	"systemctl list-unit-files"
"$unit" を有効にする (symlink 追加)	"systemctl enable $unit"
"$unit" を無効にする (symlink 削除)	"systemctl disable $unit"
"$unit" のマスクを外す ("/dev/null" への symlink を削除)	"systemctl unmask $unit"
"$unit" にマスクをかける ("/dev/null" への symlink を追加)	"systemctl mask $unit"
デフォールトのターゲット設定を取得	"systemctl get-default"
"graphical" にデフォールトのターゲットを設定 (GUI システム)	"systemctl set-default graphical"
"multi-user" にデフォールトのターゲットを設定 (CLI システム)	"systemctl set-default multi-user"
ジョブ環境の表示	"systemctl show-environment"
ジョブ環境"variable"(変数) を"value(値) に設定する"	"systemctl set-environment variable=value"
ジョブ環境"variable" (変数) の設定を解除する	"systemctl unset-environment variable"
全 unit ファイルとデーモンをリロード	"systemctl daemon-reload"
システムをシャットダウンする	"systemctl poweroff"
システムのシャットダウンとリブート	"systemctl reboot"
システムのサスペンド	"systemctl suspend"
システムのハイバーネート	"systemctl hibernate"

Table 3.6: 典型的な systemctl コマンド断片の例

ここで、上記の例の中の”$unit” は単一の unit 名 (.service や.target といったサフィックスは任意) とか、多くの場合、現在メモリー中の全 unit の主名称に対して fnmatch(3) を用いて”*” や”?” や”[]” 等のシェルスタイルのグロブによる複数 unit 指定であっても良い。

上記例中のシステムの状態を変えるコマンドは必要な管理特権を獲得させるべく”sudo” を典型的には前置されています。

”systemctl status $unit|$PID|$device” の出力は色付きドット (”●”) を使い unit の状態が一目瞭然としています。

- 白い”●” は” 活動停止” や” 活動停止中” の状態を示します。
- 赤い”●” は” 失敗発生” や” エラー発生” の状態を示します。
- 緑の”●” は” 活動中” や” リローディング” や” 活動化中” の状態を示します。

3.7 他のシステムモニター

以下は、systemd の下での他のモニタリングコマンドです。cgroups(7) を含めた該当のマンページを読んで下さい。

操作	コマンド断片
それぞれの初期化ステップにかかった時間を表示します	”systemd-analyze time”
初期化にかかった時間を全ての unit に関してリストします	”systemd-analyze blame”
読み込み”$unit” ファイル中のエラーを検出します。	”systemd-analyze verify $unit”
呼んだセッションからのユーザーの簡潔な実行時状態の表示します	”loginctl user-status”
呼んだセッションからの簡潔な実行時状態の表示します	”loginctl session-status”
cgroups を用いてブートプロセスを追跡します	”systemd-cgls”
cgroups を用いてブートプロセスを追跡します	”ps xawf -eo pid,user,cgroup,args”
cgroups を用いてブートプロセスを追跡します	”/sys/fs/cgroup/” の下の sysfs を読みます

Table 3.7: systemd の下での他のモニタリングコマンドのリスト

3.8 システム設定

3.8.1 ホスト名

カーネルがシステムのホスト名を維持管理します。systemd-hostnamed.service により起動されたシステム unit が”/etc/hostname” に保存された名前を使ってブート時にホスト名を設定します。このファイルには、完全修飾ドメイン名ではなく、システムのホスト名のみが含まれているべきです。

現在のホスト名を確認するには、hostname(1) を引数無しで実行します。

3.8.2 ファイルシステム

普通のディスクやネットワークのファイルシテムのマウントオプションは”/etc/fstab” で設定されます。fstab(5) と項 9.6.7 を参照下さい。

暗号化されたファイルシステムの設定は”/etc/crypttab” で設定されます。crypttab(5) を参照下さい。

mdadm(8) を用いるソフトウェアー RAID は”/etc/mdadm/mdadm.conf” で設定されます。mdadm.conf(5) を参照下さい。



警告
各ブートアップごとに、全てのファイルシステムをマウントした後で、”/tmp” と”/var/lock” と”/var/run” 中の一時ファイルはクリーンされます。

3.8.3 ネットワークインターフェースの初期化

systemd 下の現代的な Debian デスクトップ環境では、ネットワークインターフェースは、lo が”networking.service” で、他のインターフェースが”NetworkManager.service” で典型的には初期化されます。

どのように設定するのかは第 5 章を参照下さい。.

3.8.4 クラウドシステムの初期化

クラウドシステムインスタンス”Debian オフィッシャルクラウドイメージ” とか類似のイメージのクローンとしてローンチされるかもしれません。そのようなシステムインスタンスの場合、ホスト名やファイルシステムやネットワーキングやロケールや SSH キーやユーザーとかグループとかは cloud-init や netplan.io パッケージが提供する機能を使って、オリジナルのシステムイメージ中に置かれたファイルとかそのローンチ時に供給される外部データー等の複数のデーターソースを用いて設定されます。これらのパッケージは YAML データーを使った宣言的なシステム設定を可能にします。

詳しくは”Debian とその系統でのクラウドコンピューティング” や”Cloud- init ドキュメンテーション” や項 5.4 を参照下さい。

3.8.5 sshd サービスを調整するカスタム化例

デフォルトのインストールでは、多くのネットワークサービス (第 6 章を参照) はブート時に systemd によってブート時に network.target の後に起動されます。”sshd” も例外ではありません。カスタム化の例としてオンデマンド起動に”sshd” を変えましょう。

最初に、システムがインストールしたサービスの unit を無効化しましょう。

```
$ sudo systemctl stop sshd.service
$ sudo systemctl mask sshd.service
```

古典的 Unix サービスでは indetd (または xinetd) スーパーサーバーによりオンデマンドでソケットをアクティベーションしました。systemd では、***.socket** や ***.service** unit 設定ファイルを追加することでこれと同等のことができます。

聞くソケットを指定するには sshd.socket

```
[Unit]
Description=SSH Socket for Per-Connection Servers

[Socket]
ListenStream=22
```

```
Accept=yes

[Install]
WantedBy=sockets.target
```

`sshd.socket` に対応するサービスファイルの `sshd@.service`

```
[Unit]
Description=SSH Per-Connection Server

[Service]
ExecStart=-/usr/sbin/sshd -i
StandardInput=socket
```

そして、再ロードします。

```
 $ sudo systemctl daemon-reload
```

3.9 udev システム

udev システムはハードウエアーの自動検出と初期化のメカニズムを提供します (udev(7) 参照下さい)。カーネルが各デバイスを発見すると、udev システムは sysfs ファイルシステム (項 1.2.12 を参照下さい) からの情報を使いユーザープロセスを起動し、modprobe(8) プログラム (項 3.10 を参照下さい) を使ってそれをサポートする必要なカーネルモジュールをロードし、対応するデバイスノードを作成します。

ティップ
もし"`/lib/modules/`*`kernel-version`*`/modules.dep`" が何らかの理由で depmod(8) によって適正に生成されていなかった場合には、モジュールは udev システムによる期待にそってロードされないかもしれません。これを修正するには、"`depmod -a`" を実行します。
"`/etc/fstab`" 中のマウントルールでは、デバイス名が静的なデバイス名である必要がありません。"`/dev/sda`" 等のデバイス名ではなく UUID を使ってデバイスをマウントできます。項 9.6.3 を参照下さい。

---

udev システムは少々動くターゲットなので、詳細は他のドキュメントに譲り、ここでは最小限の記述に止めます。



警告
udev(7) で語られているように udev ルール中の `RUN` でバックアップスクリプトのような長時間実行されるプログラムを実行しようとしないで下さい。そうする代わりには、適切な systemd.service(5) ファイルを作成しそれを有効化しましょう。項 10.2.3.2 を参照下さい。

3.10 カーネルモジュール初期化

modprobe(8) プログラムは、ユーザープロセスからカーネルモジュールを追加や削除することで実行中の Linux カーネルの設定を可能にします。udev システム (項 3.9 を参照下さい) は、その起動を自動化しカーネルモジュールの初期化を補助します。

"`/etc/modules-load.d/`" ディレクトリ中の `.conf` エクステンション付きのファイル中に、1行づつリストされるプリロードする必要のある以下のような非ハードウエアーや特殊ハードウエアーのドライバーモジュールがあります。(modules-load.d(5) 参照下さい)

- ポイント間ネットワークデバイス (TUN) と仮想 Ethernet ネットワークデバイス (TAP) を提供する、TUN/TAP モジュール

- netfilter ファイアーウォール機能 (iptables(8) と項 5.7) を提供する netfilter モジュール
- ウォッチドッグタイマードライバーのモジュール

modprobe(8) プログラムのための設定ファイルは、modprobe.conf(5) で説明されているように”/etc/modprobes.d/” ディレクトリーの下にあります。(あるカーネルモジュールが自動ロードされるのを避けるには、”/etc/modprobes.d/bla ファイル中にブラックリストします。)

depmod(8) プログラムによって生成される”/lib/modules/*version*/modules.dep” ファイルは、modprobe(8) プログラムによって使われるモジュール依存関係を記述します。

注意
ブート時に modprobe(8) を使ってのモジュールロードの問題に出会った場合には、”depmod -a” として”modules.dep” を再構築をするとこの様な問題が解消できるかもしれません。

---

modinfo(8) プログラムは Linux カーネルモジュールに関する情報を表示します。

lsmod(8) プログラムは”/proc/modules” の内容を読みやすい形式にして、どのカーネルモジュールが現在ロードされているかを表示します。

ティップ
あなたのシステム上の正確なハードウエアーを特定します。項 9.5.3 を参照下さい。
ブート時に期待されるハードウエアー機能をアクティベートするように設定もできます。項 9.5.4 を参照下さい。
あなたのデバイスのサポートは、カーネルを再コンパイルすれば追加できます。項 9.10 を参照下さい。

Chapter 4

認証とアクセスの制御

人 (またはプログラム) がシステムへのアクセスの要求をした際に、認証はその正体が信頼できるものだと確認します。



警告
PAM の設定のエラーはあなたをあなた自身のシステムから締め出すかも知れません。レスキュー CD を手元に置くか代替ブートパーティション設定を必ずします。復元するには、それらを使ってシステムをブートしそこから修正します。

4.1 普通の Unix 認証

普通の Unix 認証は PAM (プラグ可能な認証モジュール) のもとで pam_unix.so(8) モジュールによって提供される。":" で分離されたエントリーを持つその 3 つの重要な設定ファイルは次です。

ファイル	パーミッション (許可)	ユーザー	グループ	説明
/etc/passwd	-rw-r--r--	root	root	(浄化された) ユーザーアカウント情報
/etc/shadow	-rw-r-----	root	shadow	保護されたユーザーアカウント情報
/etc/group	-rw-r--r--	root	root	グループ情報

Table 4.1: 3 つのpam_unix(8) に関する重要な設定ファイル

"/etc/passwd" ファイルは以下の内容です。

```
 ...
user1:x:1000:1000:User1 Name,,,:/home/user1:/bin/bash
user2:x:1001:1001:User2 Name,,,:/home/user2:/bin/bash
 ...
```

passwd(5) に説明されているように、このファイルの":" で分離されたエントリーそれぞれは以下の意味です。

- ログイン名
- パスワード規定エントリー
- 数値のユーザー ID
- 数値のグループ ID

- ユーザー名またはコメント領域
- ユーザーのホームディレクトリー
- ユーザーのコマンドインタープリター (無いこともある)

”/etc/passwd” の 2 番目のエントリーは暗号化したパスワードのエントリーとして使われていました。”/etc/shadow” が導入された後は、このエントリーはパスワード規定エントリーとして使われています。

内容	意味
(空白)	パスワード無しアカウント
x	暗号化したパスワードは”/etc/shadow” ファイルの中にあります。

Table 4.2: ”/etc/passwd” の 2 番目のエントリーの内容

”/etc/shadow” の内容は次です。

```
 ...
user1:$1$Xop0FYH9$IfxyQwBe9b8tiyIkt2P4F/:13262:0:99999:7:::
user2:$1$vXGZLVbS$ElyErNf/agUDsm1DehJMS/:13261:0:99999:7:::
 ...
```

shadow(5) で説明されているように、このファイルの”:” で分離されたエントリーそれぞれは以下の意味です。

- ログイン名
- 暗号化されたパスワード (最初が”1” で始まっているのは MD5 暗号化が使われていることを示します。”*” はログイン不可を示します。)
- 1970 年 1 月 1 日から、最後にパスワードが変更された日までの日数
- パスワードが変更可能となるまでの日数
- パスワードを変更しなくてはならなくなる日までの日数
- パスワード有効期限が来る前に、ユーザが警告を受ける日数
- パスワード有効期限が過ぎてからアカウントが使用不能になるまでの日数
- 1970 年 1 月 1 日からアカウントが使用不能になる日までの日数
- …

”/etc/group” のファイル内容は次です。

```
group1:x:20:user1,user2
```

group(5) に説明されているように、このファイルの”:” で分離されたエントリーそれぞれは以下の意味です。

- グループ名
- 暗号化されたパスワード (実際は使われていない)
- 数値のグループ ID
- ”,” で分離されたユーザー名のリスト

注意
”/etc/gshadow” ファイルは”/etc/shadow” ファイルが”/etc/group” ファイルに対する機能と同様の機能がありますが、実際には使われていません。

注意
もし”/etc/pam.d/common-auth” に”auth optional pam_group.so” 行が書き加えれ、”/etc/security/group.conf” に対応する設定がされていれば、実際のユーザーのグループメンバーシップは動的に割り当てられます。pam_group(8) を参照下さい。

注意
base-passwd パッケージはユーザーとグループに関する権威のあるリストが含まれます: ”/usr/share/doc/base-passwd/users-and-groups.html”。

4.2 アカウントとパスワードの情報管理

アカウント情報管理のための重要コマンドを記します。

コマンド	機能
`getent passwd` *user_name*	”*user_name*” のアカウント情報の閲覧
`getent shadow` *user_name*	”*user_name*” のシャドーされたアカウント情報の閲覧
`getent group` *group_name*	”*group_name*” のグループ情報の閲覧
`passwd`	アカウントのパスワード管理
`passwd -e`	アカウントのアクティベーションのために一回だけ使えるパスワードの設定
`chage`	パスワードのエージング情報管理

Table 4.3: アカウント情報を管理するコマンドのリスト

一部機能が機能するには root 権限が必要な場合があります。パスワードとデーターの暗号化は crypt(3) を参照下さい。

注意
Debian が提供する salsa 機器と同様な PAM と NSS の設定をされたシステム上では、ローカルの”/etc/passwd” や”/etc/group” や”/etc/shadow” の内容がシステムにアクティブに利用されていないことがあります。そういった環境下でも上記コマンドは有効です。

4.3 良好なパスワード

passwd(1) によるとシステムインストール時や passwd(1) コマンドによってアカウント作成する際には、次に記すようなセットからなる少なくとも 6 から 8 文字の良好なパスワードを選択するべきです。

- 小文字のアルファベット
- 数字の 0 から 9
- 句読点



警告
容易に推測できるパスワードを選んではいけません。アカウント名、社会保険番号、電話番号、住所、誕生日、家族員やペットの名前、辞書にある単語、”12345” や”qwerty” のような単純な文字列…、これらすべてパスワードに選んではいけません。

4.4 暗号化されたパスワード作成

ソルトを使って暗号化されたパスワードを生成する独立のツールがあります。

パッケージ	ポプコン	サイズ	コマンド	機能
whois	V:25, I:199	384	mkpasswd	crypt(3) ライブラリーの充実しすぎたフロントエンド
openssl	V:853, I:996	2564	openssl passwd	パスワードハッシュの計算 (OpenSSL)。passwd(1ssl)

Table 4.4: パスワード生成ツールのリスト

4.5 PAM と NSS

Debian システムのような現代的な Unix 的システムは PAM (プラグ可能な認証モジュール: Pluggable Authentication Modules) と NSS (ネームサービススイッチ: Name Service Switch) メカニズムをローカルのシステム管理者がそのシステム管理用に提供します。それらの役割をまとめると以下のようになります。

- PAM は、アプリケーションソフトウェアーが使う柔軟な認証メカニズムを提供し、パスワードデーターの交換に関与します。
- NSS は、ls(1)andid(1) 等のプログラムがユーザーやグループの名前を得ために C 標準ライブラリー経由で頻用する柔軟なネームサービスメカニズムを提供します。

これらの PAM と NSS システムは一貫した設定が必要です。

PAM と NSS システムに関する注目のパッケージは次です。

パッケージ	ポプコン	サイズ	説明
libpam-modules	V:945, I:1000	892	差し替え可能な認証モジュール (基本サービス)
libpam-ldapd	V:8, I:13	82	LDAP インターフェースを可能にする差し替え可能な認証モジュール
libpam-systemd	V:788, I:972	849	logind のために登録ユーザーセッションを登録するプラガブルオーセンティケーション (PAM)
libpam-doc	I:7.7	1492	差し替え可能な認証モジュール (html と text の文書)
libc6	V:924, I:1000	5483	GNU C ライブラリー: ” ネームサービススイッチ” も提供する共有ライブラリー
glibc-doc	I:4.9	3927	GNU C ライブラリー: マンページ
glibc-doc-reference	I:3.0	14764	GNU C ライブラリー: info と pdf と html フォーマットでのリファレンスマニュアル (non-free)
libnss-mdns	V:314, I:505	142	マルチキャスト DNS を使った名前解決のための NSS モジュール
libnss-ldapd	V:9, I:16	138	LDAP をネームサービスとして使う NSS モジュール

Table 4.5: 特記すべき PAM と NSS システムのリスト

- libpam-doc 中の”The Linux-PAM System Administrators’ Guide” は PAM 設定を学ぶ上で必須です。

- `glibc-doc-reference` の中の”System Databases and Name Service Switch” セクションは NSS 設定を学ぶ上で必須です。

注意
より大規模かつ最新のリストは”`aptitude search 'libpam-|libnss-'`” コマンドを実行すると得られます。NSS という頭字語は” ネームサービススイッチ: Name Service Switch” と異なる” ネットワークセキュリティーサービス: Network Security Service” を指すこともあります。

注意
PAM は個別プログラムに関する環境変数をシステム全体のデフォールト値に初期化する最も基本的な手段です。

---

systemd の下では、logind のために `systemd` のコントロールグループ階層中にユーザーセッションを登録することでユーザーのログインを管理すべく `libpam-systemd` パッケージがインストールされている。systemd-logind(8) や logind.conf(5) や pam_systemd(8) を参照下さい。

4.5.1 PAM と NSS によってアクセスされる設定ファイル

PAM と NSS がアクセスする注目すべき設定ファイルを次に記します。

設定ファイル	機能
`/etc/pam.d/`プログラム名	”*`program_name`*” に関する PAM 設定の設定; pam(7) と pam.d(5) 参照下さい
`/etc/nsswitch.conf`	各サービスに関するエントリーによる NSS 設定の設定; nsswitch.conf(5) 参照下さい
`/etc/nologin`	ユーザーのログイン制限のために pam_nologin(8) モジュールがアクセス
`/etc/securetty`	pam_securetty(8) モジュールにより root アクセスに使う tty を制限
`/etc/security/access.conf`	pam_access(8) モジュールによりアクセス制限を設定
`/etc/security/group.conf`	pam_group(8) モジュールによりグループに基づく制約を設定
`/etc/security/pam_env.conf`	pam_env(8) モジュールにより環境変数を設定
`/etc/environment`	”`readenv=1`” 引数を付きの pam_env(8) モジュールによって追加での環境変数を設定
`/etc/default/locale`	”`readenv=1envfile=/etc/default/locale`” 引数を付きの pam_env(8) モジュールによって追加でロケールを設定します (Debian)
`/etc/security/limits.conf`	pam_linits(8) モジュールによってリソース制限 (ulimit, core, ⋯) を設定
`/etc/security/time.conf`	pam_time(8) モジュールによって時間制限を設定
`/etc/systemd/logind.conf`	`systemd` ログイン管理設定の設定 (logind.conf(5) と systemd-logind.service(8) を参照)

Table 4.6: PAM NSS によりアクセスされる設定ファイルのリスト

パスワード選択の制限は pam_unix(8) と pam_cracklib(8) モジュールで実装されています。それらは引数を使って設定します。

ティップ
PAM モジュールはファイル名のサフィクスとして”`.so`” を使います。

4.5.2 集中システム管理

集中化された軽量ディレクトリーアクセスプロトコル (LDAP) を採用することで多くのネットワーク上の Unix 的や非 Unix 的システムを運営する、現代的な集中システム管理が実現できます。軽量ディレクトリーアクセスプロトコルのオープンソース実装は OpenLDAP ソフトウェアーです。

LDAP サーバーは、`libpam-ldapd` と `libnss-ldapd` パッケージで提供される PAM と NSS を使うことで Debian システムにアカウント情報を提供します。この実現ためにはいくつかの設定が必要です (著者は本設定を使っていないため、以下の情報は完全に二次情報です。ご理解の上お読み下さい。)。

- スタンドアローンの LDAP デーモンである slapd(8) 等のプログラムを実行することで集中化された LDAP サーバーを設置します。
- デフォールトの”`pam_unix.so`” に代えて”`pam_ldap.so`” を使うには”`/etc/pam.d/`” ディレクトリー中の PAM 設定ファイルを変更します。
- デフォールト (”`compat`” または”`file`”) に代えて”`ldap`” を使うには”`/etc/nsswitch.conf`” ファイル中の NSS 設定を変更します。
- パスワードのセキュリティー確保のために `libpam-ldapd` が SSL (もしくは TLS) 接続を使うよう設定しなければいけません。
- LDAP のネットワークオーバーヘッドのコストは掛かりますが、データの整合性確保のために `libnss-ldapd` が SSL (もしくは TLS) 接続を使うように設定できます。
- LDAP のネットワークトラフィックを減少させるために LDAP サーチ結果を一時保持するための nscd(8) をローカルで走らせるべきです。

nsswitch.conf(5) や pam.conf(5) や ldap.conf(5) や `libpam-doc` パッケージで提供される `/usr/share/doc/libpam-do` や”`info libc 'NameServiceSwitch'`” といった文書を参照下さい。

同様に、これに代わる集中化されたシステムは他の方法を使っても設定できます。

- Windows システムとのユーザーとグループの統合
  - `winbind` と `libpam_winbind` パッケージを使って Windows ドメインサービスにアクセスします。
  - winbind(8) と SAMBA による MS Windows Networks への統合を参照下さい。
- 旧来の Unix 的なシステムとのユーザーとグループの統合
  - `nis` パッケージにより NIS (当初 YP と呼ばれた) または NIS+ にアクセス
  - The Linux NIS(YP)/NYS/NIS+ HOWTO 参照下さい。

4.5.3 「どうして **GNU** の **su** は **wheel** グループをサポートしないのか」

これは Richard M. Stallman が書いた昔の”`info su`” の最後に書かれていた有名な文言です。ご心配は無用です。現在 Debian にある `su` は PAM を使っているので”`/etc/pam.d/su`” の中の”`pam_wheel.so`” の行をエネーブルすることで `su` を使えるのを `root` グループに限定できます。

4.5.4 パスワード規則強化

`libpam-pwquality` パッケージをインストールすると、より厳格なパスワード規則を強制できます。

自動的に `libpam-gnome-keyring` をインストールする典型的な GNOME システム上では、”`/etc/pam.d/common-pass` は以下です:

```
# here are the per-package modules (the "Primary" block)
password  requisite     pam_pwquality.so retry=3 minlen=8 difok=3
password  [success=1 default=ignore]  pam_unix.so obscure use_authtok try_first_pass  ↩
    yescrypt
# here's the fallback if no module succeeds
password  requisite     pam_deny.so
# prime the stack with a positive return value if there isn't one already;
# this avoids us returning an error just because nothing sets a success code
# since the modules above will each just jump around
password  required      pam_permit.so
# and here are more per-package modules (the "Additional" block)
password  optional  pam_gnome_keyring.so
# end of pam-auth-update config
```

4.6 認証のセキュリティー

注意
ここに書かれている情報はあなたのセキュリティーのニーズに充分ではないかもしれませんが、良いスタートです。

4.6.1 インターネット上でセキュアーなパスワード

多くのトランスポーテーションレイヤーサービスはパスワード認証も含めて暗号化せずにメッセージをプレーンテキストで通信します。途中で傍受されかねないインターネットの荒野を経由して暗号化せずパスワードを送ることは非常によくない考えです。これらに関しては、” トランスポーテーションレイヤーセキュリティー”(TLS) もしくはその前身の” セキュアーソケットレイヤー” (SSL) で暗号化することでパスワードを含むすべての通信をセキュアーにしてサービスができます。

インセキュアーなサービス名	ポート	セキュアーなサービス名	ポート
www (http)	80	https	443
smtp (mail)	25	ssmtp (smtps)	465
ftp-data	20	ftps-data	989
ftp	21	ftps	990
telnet	23	telnets	992
imap2	143	imaps	993
pop3	110	pop3s	995
ldap	389	ldaps	636

Table 4.7: インセキュアーとセキュアーのサービスとポートのリスト

暗号化には CPU タイムがかかります。CPU に友好的な代替方法として、POP には” パスワードを認証されたポストオフィスプロトコル”(APOP) や SMTP や IMAP には” チャレンジレスポンス認証メカニズム MD5” (CRAM-MD5) といったセキュアーな認証プロトコルでパスワードのみを保護しつつ通信はプレーンテキストですることもできます。(最近メールクライアントからメールサーバーにインターネット経由でメールメッセージを送る際には、CRAM-MD5 で認証をしたのちネットワークプロバイダーによるポート 25 ブロッキングを避けて従来の SMTP ポート 25 の代わりにメッセージサブミッションポート 587 を使うことがよく行われます。)

これでも、伝統的なパスワードを使う認証には複数ウエッブサイトをまたぐパスワード再使用のような内在するセキュリティー問題があります。他のセキュアーな解決のための認証法を参照: タイムベーストワンタイムパスワード (TOTP), ユニバーサル２要素 (U2F), Web 認証 (WebAuthn, 一般にパスキー”passkey” と呼ばれる)

4.6.2 セキュアーシェル

セキュアーシェル (SSH) プログラムはセキュアーな認証とともにインセキュアーなネットワークを通過したお互いに信頼し合っていないホスト間のセキュアーで暗号化された通信を可能にします。OpenSSH クライアント ssh(1) と OpenSSH デーモン sshd(8) から成り立っています。SSH はポートフォーワーディング機能を使い POP や X のようなインセキュアープロトコルの通信をインターネット経由でトンネルするのに使えます。

クライアントは、ホストベースド認証、公開キー認証、チャレンジレスポンス認証、パスワード認証を使って認証をとろうとします。公開キー認証を利用すると、リモートからのパスワード無しログインができるようになります。項 6.3 を参照下さい。

4.6.3 インターネットのためのセキュリティー強化策

たとえ、セキュアーシェル (SSH) やポイントツーポイントトンネリングプロトコル (PPTP) サーバーのようなセキュアーサービスを走らせる場合でも、ブルートフォースのパスワード推測等による侵入の可能性は残っています。以下のようなセキュリティーのためのツールとともに、ファイアーウォールポリシー (項 5.7 を参照下さい) を使うのはセキュリティー状況を向上させることが期待できます。

パッケージ	ポプコン	サイズ	説明
knockd	V:0.7, I:1.5	110	小さなポートノックのデーモン knockd(1) とクライアント konck(1)
fail2ban	V:101, I:112	2275	複数回の認証エラーを発生させる IP を使用禁止にします

Table 4.8: 追加セキュリティー策を提供するツールのリスト

4.6.4 root パスワードのセキュリティー確保

あなたの機器に第三者が root 権限を持ってアクセスするのを阻止するには、以下のアクションが必要です。

- ストレージデバイス (HDD / SSD / ...) への物理アクセスを阻止
- UEFI/BIOS をロックして、リムーバブルメディアからのブートを阻止
- GRUB のインタラクティブセッションのパスワードを設定
- GRUB のメニュー項目編集に施錠

ハードディスクへの物理的アクセスがあれば、以下の手順を使い比較的簡単にパスワードをリセットできます。

1. ハードディスクを USB からブート可能な UEFI/BIOS のついた PC に移動します。
2. レスキュー媒体 (see 項 3.2.2 参照) でシステムをブートします。
3. ルートパーティションを読出し / 書込みアクセスでマウントします。
4. ルートパーティションの”/etc/passwd” を編集し、root アカウントの 2 番目の項目を空にします。

grub-rescue-pc のブート時に GRUB のメニュー項目を編集可能 (項 3.1.2 を参照下さい) なら、以下の手順を使えてさらに簡単です。

1. カーネルパラメーターを”root=/dev/sda6 rw init=/bin/sh” のような感じに変更してシステムをブートします。
2. ”/etc/passwd” を編集し、root アカウントの 2 番目の項目を空にします。

3. システムをリブートします。

これで、システムの root シェルにパスワード無しに入れるようになりました。

注意
root シェルにアクセスさえできれば、システム上の全てにアクセスできシステム上のどのパスワードでもリセットできます。さらに。`john` とか `crack` パッケージ (項 9.5.10 を参照下さい) のようなブルートフォースのパスワードクラッキングツールを使ってすべてのユーザーアカウントのパスワードが破られるかもしれません。こうして破られたパスワードは他のシステムへの侵入を引き起こしかねません。

この様な懸念を回避できる唯一の合理的なソフトウェアー的解決法は、dm-crypt と initramfs (項 9.9 参照下さい) をつかう、ソフトウェアー暗号化されたルートパーティション (もしくは"`/etc`" パーティション) を使うことです。でも、パスワードがシステムのブート毎に必要になってしまいます。

4.7 他のアクセスコントロール

パスワードによる認証システムやファイルパーミッション以外のシステムへのアクセスコントロールがあります。

注意
カーネルのセキュアーアテンションキー (SAK) 機能の制限は項 9.4.16 を参照下さい。

4.7.1 アクセス制御リスト (ACLs)

ACL は、項 1.2.3 で説明された通常のパーミッションの上位集合です。

現代的なデスクトップ環境では ACL が作動していることに出くわします。フォーマットされた USB ストレージデバイスが例えば"`/media/penguin/USBSTICK`" として自動マウントされた場合、通常ユーザーの `penguin` は以下を実行できます:

```
 $ cd /media/penguin
 $ ls -la
total 16
drwxr-x---+ 1 root    root    16 Jan 17 22:55 .
drwxr-xr-x  1 root    root    28 Sep 17 19:03 ..
drwxr-xr-x  1 penguin penguin 18 Jan  6 07:05 USBSTICK
```

11 列目にある"+" は ACL 作動を表示しています。ACL 無しだと、通常ユーザーの `penguin` は、`penguin` が `root` グループに属さないためこのようにリストできません。ACL は以下のようにして確認できます:

```
 $ getfacl .
# file: .
# owner: root
# group: root
user::rwx
user:penguin:r-x
group::---
mask::r-x
other::---
```

ここで:

- "`user::rwx`" や"`group::---`" や"`other::---`" は、通常の所有者や、グループや、第三者のパーミッションに対応します。

- ”user:penguin:r-x” という ACL は通常ユーザーの penguin が”r-x” パーミッションを保有することを可能にします。これにより、”ls -la” でディレクトリー内容をリストできるようになります。
- ”mask::r-x” という ACL はパーミッションの上限を設定します。

詳細は”POSIX Access Control Lists on Linux” や acl(5) や getfacl(1) や setfacl(1) を参照下さい。

4.7.2 sudo

sudo はシステム管理者がユーザーに制限付きの root 権限を与え、その root 活動を記録するように設計されたプログラムです。sudo は通常のユーザーのパスワードだけが必要です。sudo パッケージをインストールし、”/etc/sudoers” の中のオプションを設定することによりアクティベートして下さい。”/usr/share/doc/sudo/examp
や項 1.1.12 の設定例を参照下さい。

単一ユーザーシステムにおける私の sudo の使い方 (項 1.1.12 を参照下さい) は自分自身の失敗からの防衛を目指しています。sudo を使うことは、常に root アカウントからシステムを使うよりは良い方法だと個人的には考えます。例えば、次は”*some_file*” の所有者を”*my_name*” に変更します。

```
$ sudo chown my_name some_file
```

root のパスワード (自分でシステムインストールをした Debian ユーザーなら当然知っています) を知っていれば、どのユーザーアカウントからいかなるコマンドも”su -c” とすれば root もとで実行できます。

4.7.3 PolicyKit

PolicyKit は Unix 系オペレーティングシステムにおけるシステム全体の特権を制御するオペレーティングシステム構成要素です。

新しい GUI アプリケーションは、特権プロセスとして実行するように設計されていません。それらは、PolicyKit を経由し管理操作を実行する特権プロセスに話しかけます。

PolicyKit は、このような操作を Debian システム上の sudo グループ所属のユーザーアカウントに限定します。

polkit(8) を参照下さい。

4.7.4 サーバーのサービスへのアクセスの制限

システムのセキュリティーのためにできるだけ多くのサーバープログラムを無効とするのは良い考えです。このことはネットワークサーバーの場合は決定的です。直接デーモンとしてであれスーパーサーバープログラム経由であれアクティベートされている使っていないサーバーがあることはセキュリティーリスクと考えられます。

sshd(8) 等の多くのプログラムが PAM を使ったアクセスコントロールを使っています。サーバーサービスへのアクセスを制限するには多くの方法があります。

- 設定ファイル: ”/etc/default/プログラム名”
- デーモンに関する systemd サービス unit 設定
- PAM (プラグ可能な認証モジュール: Pluggable Authentication Modules)
- スーパーサーバーに関する”/etc/inetd.conf”
- TCP ラッパーに関する”/etc/hosts.deny” と”/etc/hosts.allow”、tcpd(8)
- Sun RPC に関する”/etc/rpc.conf”
- atd(8) に関する”/etc/at.allow” と”/etc/at.deny”

- atd(8) に関する”/etc/at.allow” と”/etc/at.deny”
- netfilter インフラのネットワークファイアーウォール

項 3.6 と項 4.5.1 と項 5.7 を参照下さい。

ティップ
NFS 他の RPC を使うプログラムためには Sun RPC サービスはアクティブにする必要があります。

ティップ
もし現代的な Debian システムでリモートアクセスで問題に会った場合には、”/etc/hosts.deny” 中に”ALL: PARANOID” 等の問題となっている設定があればコメントアウトします。(ただしこの種の行為に関するセキュリティーリスクに注意を払わなければいけません。)

4.7.5 Linux のセキュリティ機能

Linux カーネルは進化していて、伝統的な UNIX 実装には見当たらないセキュリティー機能をサポートします。

Linux は、伝統的な UNIX アトリビュートを拡張する拡張アトリビュートをサポートします (xattr(7) を参照下さい)。

Linux は、伝統的にスーパーユーザーに紐付けられた特権を capabilities(7) として知られた、独立に有効化や無効化できる、別個の単位に分割します。

Linux セキュリティーモジュール (LSM) フレームワークは、新規のカーネル拡張により接続される各種セキュリティーチェックのためのメカニズムを提供します。例えば:

- AppArmor
- Security-Enhanced Linux (SELinux)
- Smack (簡易強制アクセス制御カーネル)
- Tomoyo Linux

このような拡張は特権モデルを通常の Unix ライクのセキュリティーモデルポリシーより厳しく引き締められるので、ルートの力も制約されるかもしれません。kernel.org にある Linux セキュリティーモデル (LSM) フレームワーク文書を読むことをおすすめします。

Linux namespaces は、namespace 内のプロセスから見たらプロセスがグローバルリソースの中でそれ自身のインスタンスがあるようにグローバルステムリソースを抽象化し包み込んでいます。グローバルリソースの変更は namespace のメンバーである他のプロセスからは見えますが、それ以外のプロセスからは見えません。カーネルバージョン 5.6 以降、8 種の namespace があります (namespaces(7) と unshare(1) と nsenter(1) を参照下さい)。

Debian 11 Bullseye (2021 年) の時点では、Debian は統一 cgroup ヒエラルキー (cgroups-v2 と呼ばれています)。

プロセスを隔離しリソースコントロールを可能にする cgroups を使う namespaces の使用例は以下です:

- Systemd。項 3.3.1 を参照下さい。
- サンドボックス環境。項 7.7 を参照下さい。
- Docker や LXC 等の Linux コンテナ. 項 9.11 を参照下さい。

このような機能は項 4.1 では実現できません。このような高度のトピックスは当該入門書のほぼ対象外です。

Chapter 5

ネットワークの設定

ティップ
現代的な Debian に特化したネットワーク設定のガイドは The Debian Administrator's Handbook —Configuring the Network を参照下さい。

ティップ
systemd の下では、networkd がネットワーク管理に使えます。systemd-networkd(8)y> を参照下さい。

5.1 基本的ネットワークインフラ

現代的な Debian システムの基本的ネットワークインフラをレビューします。

5.1.1 ホスト名の解決

ホスト名の解決もまた、現在 NSS (ネームサービススイッチ、Name Service Switch) メカニズムによってサポートされています。この解決の流れは次です。

1. "`hosts: files dns`" のようなスタンザのある"`/etc/nsswitch.conf`" ファイルがホスト名の解消の順序を規定します。(これは、"`/etc/host.conf`" ファイル中の"`order`" スタンザの機能を置換します。)
2. `files` メソッドが最初に呼び出されます。ホスト名が"`/etc/hosts`" ファイルに見つかると、それに対応する全ての有効アドレスを返し終了します。("`/etc/host.conf`" ファイルは"`multi on`" を含みます。)
3. `dns` メソッドが発動されます。"`/etc/resolv.conf`" ファイルで識別されるインターネットドメイン名システム (DNS) への問い合わせでホスト名が見つかれば、それに関する全ての有効アドレスを返します。

典型的なワークステーションでは、ホスト名は例えば"*host_name*" と設定され、オプショナルなドメイン名は空文字列に設定されてインストールされているかもしてません。その場合、"`/etc/hosts`" は以下のようになります。

```
127.0.0.1 localhost
127.0.1.1 host_name

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

パッケージ	ポプコン	サイズ	タイプ	説明
network-manager	V:429, I:472	7366	設定::NM	NetworkManager (デーモン): ネットワークを自動管理
network-manager-gnome	V:38, I:136	19	設定::NM	NetworkManager (GNOME フロントエンド)
netplan.io	V:2.4, I:8.7	340	設定::NM+networkd	Netplan (生成器): NetworkManager と systemd-networkd バックエンド用の統一的で宣言的なインターフェース
ifupdown	V:686, I:973	204	設定::ifupdown	ネットワークを接続したり切断したりする標準化されたツール (Debian 特定)
pppoeconf	V:0.2, I:3.6	176	設定:: ヘルパー	PPPoE コネクションの設定ヘルパー
wpasupplicant	V:431, I:520	4337	設定:: ヘルパー	WPA と WPA2 (IEEE 802.11i) のためのクライアントサポート
wpagui	V:0.2, I:1.2	780	設定:: ヘルパー	wpa_supplicant の Qt GUI クライアント
wireless-tools	V:196, I:263	232	設定:: ヘルパー	Linux のワイヤレス拡張を操作するツール
iw	V:42, I:478	332	設定:: ヘルパー	Linux のワイヤレスデバイスを設定するツール
iproute2	V:802, I:988	4279	設定::iproute2	iproute2、IPv6 や他の高度なネットワーク設定: ip(8) や tc(8) 等
iptables	V:402, I:635	2408	設定::Netfilter	パケットフィルターと NAT のための管理ツール (Netfilter)
nftables	V:301, I:889	189	設定::Netfilter	パケットフィルターと NAT のための管理ツール (Netfilter) ({ip,ip6,arp,eb}tables の後継)
iputils-ping	V:222, I:998	188	テスト	ホスト名か IP アドレスによってリモートホストのネットワークからの到達性をテスト (iproute2)
iputils-arping	V:2, I:17	53	テスト	ARP アドレスによって特定されるリモートホストのネットワークからの到達性をテスト
iputils-tracepath	V:2, I:19	50	テスト	リモートホストへのネットワークパスを追跡
ethtool	V:103, I:252	1106	テスト	Ethernet デバイス設定の表示と変更
mtr-tiny	V:5, I:38	182	テスト:: 低レベル	リモートホストへのネットワークパスを追跡するツール (curses)
mtr	V:4, I:39	231	テスト:: 低レベル	リモートホストへのネットワークパスを追跡するツール (curses と GTK)
gnome-nettool	V:0.5, I:7.9	2480	テスト:: 低レベル	共通のネットワーク情報操作のためのツール (GNOME)
nmap	V:27, I:180	4946	テスト:: 低レベル	ネットワークマッパー / ポートスキャナー (Nmap、コンソール)
tcpdump	V:24, I:181	1418	テスト:: 低レベル	ネットワークトラフィックアナライザー (Tcpdump、コンソール)
wireshark	V:4, I:38	17442	テスト:: 低レベル	ネットワークトラフィックアナライザー (Wireshark、GTK)
tshark	V:3, I:23	615	テスト:: 低レベル	ネットワークトラフィックアナライザー (コンソール)
tcptrace	V:0.2, I:1.5	407	テスト:: 低レベル	tcpdump の出力から接続状況のまとめを作成
dnsutils	V:4, I:137	23	テスト:: 低レベル	BIND によって提供されるネットワーククライアント: nslookup(1) と nsupdate(1) と dig(1)
dlint	V:0.1, I:1.9	52	テスト:: 低レベル	ネームサーバーの閲覧で DNS のゾーン情報をチェック
dnstracer	V:0.1, I:1.0	63	テスト:: 低レベル	DNS サーバーをその源流まで追跡

Table 5.1: GUI のネットワーク設定ツール

各行は IP アドレスで始まり、関連するホスト名がそれに続きます。

本例の 2 行目の IP アドレス `127.0.1.1` は他の Unix 系システムでは見かけないかもしれません。bug #719621 に記録されているように、Debian インストーラーは恒久的 IP アドレスのないシステムのために一部ソフトウェアー (GNOME 等) のための回避策としてこの項目を作成します。

host_name は、"`/etc/hostname`" の中に定義されたホスト名と一致します (項 3.8.1 参照)。

恒久的 IP アドレスを持つシステムでは `127.0.1.1` の代えてその恒久的 IP アドレスがここにあるべきです。

恒久的 IP アドレスと Domain 名システム (DNS) が提供する完全修飾ドメイン名 (FQDN) を持つシステムでは、その標準的な *host_name(*ホスト名*).domain_name(*ドメイン名*)* が *host_name(*ホスト名*)* のみに代えて使われるべきです。

`resolvconf` パッケージがインストールされなかったら、"`/etc/resolv.conf`" は静的なファイルです。インストールされると、それはシンボリックリンクになります。いずれにせよ、解決機構を初期化する情報を含んでいます。もし DNS が IP="`192.168.11.1`" に見つかるなら、それは以下の内容です。

```
nameserver 192.168.11.1
```

`resolvconf` パッケージはこの"`/etc/resolv.conf`" をシンボリックリンクにし、フックスクリプトで自動的にその内容を管理します。

典型的 adhoc な LAN 環境にある PC ワークステーションの場合、基本的な `files` や `dns` 法に加えて Multicast DNS (mDNS) 経由でホスト名を解決する事ができます。

- Avahi は Debian で Multicast DNS サービスの探索の枠組みを提供します。
- Apple Bonjour / Apple Rendezvous と同等です。
- `libnss-mdns` プラグインパッケージが GNU C ライブラリー (glibc) の GNU Name Service Switch (NSS) 機能に mDNS 経由のホスト名解決を提供します。
- "`/etc/nsswitch.conf`" ファイルには"`hosts: files mdns4_minimal [NOTFOUND=return] dns`" のようなスタンザがあるべきです (他の設定は`/usr/share/doc/libnss-mdns/README.Debian` を参照下さい)。
- ".local" で終わる擬似-top-level domain が末尾についたホスト名は、IPv4 アドレス"`224.0.0.251`" か IPv6 アドレス"`FF02::FB`" から、マルチキャスト UDP パケット中の mDNS クエリーメッセージ送ることで解決されます。

注意
ドメイン名システムにおけるジェネリックトップレベルドメイン (gTLD) の拡張が進行中です。LAN 内のみで使うドメイン名を選ぶ際に名前衝突に注意が必要です。

注意
`systemd-resolved` と合わせての `libnss-resolve` や、`libnss-myhostname` や、`libnss-mymachine` のようなパッケージを"`/etc/nsswitch.conf`" ファイル中の"`hosts`" 行上に対応するリスト項目を挙げて使用すると、上記で述べた伝統的なネットワーク設定はオーバーライドされるかもしれません。詳しくは、nss-resolve(8) や、systemd-resolved(8) や、nss-myhostname(8) や、nss-mymachines(8) を参照下さい。

5.1.2 ネットワークインターフェース名

systemd は"`enp0s25`" のような" 予測可能なネットワークインターフェース名" を用います。

5.1.3 LAN のためのネットワークアドレス範囲

rfc1918 によってローカルエリアネットワーク (LAN) での使用に予約されている各クラス毎の IPv4 32 ビットアドレス範囲を確認します。これらのアドレスは本来のインターネット上のアドレスとかち合う事が無いことが保証されています。

注意
コロンのついた IP アドレスは IPv6 アドレスです。たとえば"`::1`" は `localhost` です。

クラス	ネットワークアドレス	ネットマスク	ネットマスク / ビット	サブネットの数
A	10.x.x.x	255.0.0.0	/8	1
B	172.16.x.x —172.31.x.x	255.255.0.0	/16	16
C	192.168.0.x —192.168.255.x	255.255.255.0	/24	256

Table 5.2: ネットワークアドレス範囲のリスト

注意
これらのアドレス内の 1 つがホストに付与されている場合、そのホストはインターネットに直接アクセスせず、各サービスのプロキシとなるかネットワークアドレス変換 (NAT) をするゲートウエーを通してアクセスしなければいけません。ブロードバンドルーターは消費者 LAN 環境のために通常 NAT を行います。

5.1.4 ネットワークデバイスサポート

Debian システムによってほとんどのハードウエアーデバイスはサポートされていますが、一部のネットワークデバイスはそのサポートのために DFSG non-free のファームウエアーが必要です。項 9.10.5 を参照下さい。

5.2 デスクトップのための現代的なネットワーク設定

`systemd` 下の現代的な Debian デスクトップ環境では、ネットワークインターフェースは、`lo` が"`networking.service`" で、他のインターフェースが"`NetworkManager.service`" で典型的には初期化されます。

Debian では、NetworkManager (NM) (network-manager と関連パッケージ) 等の管理デーモン経由でネットワーク接続の管理ができます。

- それらは洒落た GUI やコマンドラインのユーザーインターフェースが同梱されています。
- それらにはバックエンドシステムとして、自前のデーモンが同梱されています。
- それらによりあなたのシステムをインターネットへ容易に接続できます。
- それらによりインターネットへの有線や無線のネットワークの管理が容易にできます。
- それらにより旧来の"`ifupdown`" パッケージと独立にネットワークを設定できます。

注意
サーバーにはこの様な自動ネットワーク設定を使わないで下さい。これらはラップトップ上のモービルデスクトップを主対象としています。

これらの現代的なネットワーク設定ツールは旧来の"`ifupdown`" パッケージやその"`/etc/network/interfaces`" 設定ファイルとの競合を避けるように適正に設定する必要があります。

5.2.1 GUI のネットワーク設定ツール

Debian における NM の公式のドキュメンテーションは”/usr/share/doc/network-manager/README.Debian” にあります。

デスクトップのための現代的ネットワーク設定の要点は以下です。

1. 以下のようにして、例えば foo というデスクトップユーザーを”netdev” グループに属するようにします。(GNOME や KDE のような現代的デスクトップ環境の下では D-bus 経由でそれを自動的にするのも一つの方法です。)

```
$ sudo usermod -a -G netdev foo
```

2. ”/etc/network/interfaces” の設定を以下のようにできるだけ簡単にします。

```
auto lo
iface lo inet loopback
```

3. 以下のようにして NM を再起動します。

```
$ sudo systemctl restart NetworkManager
```

4. GUI 経由でネットワークを設定します。

注意
ifupdown との干渉を避けるために、NM は”/etc/network/interfaces” にリストされてないインターフェースのみを管理します。

ティップ
NM のネットワーク設定能力を拡張したい場合には、network-manager-openconnect、network-manager-openvpn-gnome、network-manager-pptp-gnome、mobile-broadband-provider-info 等の適当なプラグインモジュールや補足パッケージを探して下さい。

5.3 GUI 無しの現代的なネットワーク設定

上記とは異なり、systemd の下では、ネットワークは /etc/systemd/network/ を使って設定されているかもしれません。systemd-resolved(8) や resolved.conf(5) や systemd-networkd(8) を参照下さい。

これにより GUI 無しの現代的なネットワーク設定ができます。

DHCP クライアントの設定は”/etc/systemd/network/dhcp.network” を作成することで設定できます。例えば:

```
[Match]
Name=en*

[Network]
DHCP=yes
```

静的ネットワーク設定は”/etc/systemd/network/static.network” を作成することで設定できます。例えば:

```
[Match]
Name=en*

[Network]
Address=192.168.0.15/24
Gateway=192.168.0.1
```

5.4 クラウドのための現代的なネットワーク設定

クラウドのための現代的なネットワーク設定は `cloud-init` と `netplan.io` パッケージを使っているかもしれません (項 3.8.4 を参照下さい)。

`netplan.io` パッケージはネットワーク設定バックエンドとして `systemd-networkd` と `NetworkManager` をサポートし、宣言的な YAML データーを使ったネットワーク設定を可能にします。YAML を変更する時は:

- "`netplan generate`" コマンドを実行して、YAML からすべての必要なバックエンド設定を生成します。
- "`netplan apply`" コマンドを実行して、生成される設定をバックエンドに適用します。

"Netplan ドキュメンテーション" と netplan(5) と netplan-generate(8) と netplan-apply(8) を参照下さい。

更に、`cloud-init` がどのようにして `netplan.io` の設定を代替データーソースを使って一体化するのかは"Cloud-init ドキュメンテーション" (特に" 設定ソース" と"Netplan パススルー") を参照下さい。

5.4.1 クラウドのための現代的なネットワーク設定

DHCP クライアントの設定はデーターソースファイル"`/etc/systemd/network/dhcp.network`" を作成することで設定できます:

```
network:
  version: 2
  ethernets:
    all-en:
      match:
        name: "en*"
      dhcp4: true
      dhcp6: true
```

5.4.2 クラウドのための静的 IP を使う現代的なネットワーク設定

静的ネットワーク設定はデーターソースファイル"`/etc/systemd/network/static.network`" を作成することで設定できます:

```
network:
  version: 2
  ethernets:
    eth0:
      addresses:
        - 192.168.0.15/24
      routes:
        - to: default
          via: 192.168.0.1
```

5.4.3 クラウドのための Network Manager を使う現代的なネットワーク設定

Network Manager インフラを用いるネットワーククライアントの設定は、データーソースファイル"`/etc/netplan/00-ne`
を作成することで設定できます:

```
network:
  version: 2
  renderer: NetworkManager
```

5.5 低水準ネットワーク設定

Linux における低水準のネットワークを設定するには iproute2 プログラム (ip(8)、…) を用います。

5.5.1 Iproute2 コマンド

Iproute2 コマンドは低水準ネットワーク設定の完全な機能を提供します。旧式の net-tools コマンドと新しい iproute2 コマンド等との翻訳表を次に示します。

旧式の **net-tools**	新しい **iproute2** 等	操作
ifconfig(8)	`ip addr`	デバイスのプロトコル (IP または IPv6) アドレス
route(8)	`ip route`	ルーティングテーブル
arp(8)	`ip neigh`	ARP または NDISC キャッシュ項目
`ipmaddr`	`ip maddr`	マルチキャストアドレス
`iptunnel`	`ip tunnel`	IP 経由トンネル
nameif(8)	ifrename(8)	MAC アドレスに基づきネットワークインターフェースを命名
mii-tool(8)	ethtool(8)	イーサーネットデバイスの設定

Table 5.3: 旧式の `net-tools` コマンドと新しい `iproute2` コマンド等との翻訳表

ip(8) と Linux Advanced Routing & Traffic Control を参照下さい。

5.5.2 安全な低レベルネットワーク操作

以下の低レベルネットワークコマンドは、ネットワーク設定を変更しないので安全に使えます。

コマンド	説明
`ip addr show`	アクティブなインターフェースのリンクとアドレスの状態を表示
`route -n`	数字を使ったアドレスで全てのルーティングテーブルを表示
`ip route show`	数字を使ったアドレスで全てのルーティングテーブルを表示
`arp`	ARP キャッシュテーブルの現状の内容を表示
`ip neigh`	ARP キャッシュテーブルの現状の内容を表示
`plog`	ppp デーモンのログを表示
`ping yahoo.com`	"`yahoo.com`" までのインターネット接続の確認
`whois yahoo.com`	ドメインデーターベースに"`yahoo.com`" を誰が登録したかを確認
`traceroute yahoo.com`	"`yahoo.com`" までのインターネット接続の追跡
`tracepath yahoo.com`	"`yahoo.com`" までのインターネット接続の追跡
`mtr yahoo.com`	"`yahoo.com`" までのインターネット接続の追跡 (繰り返し)
`dig [@dns-server.com] example.com [{a|mx|any}]`	"`example.com`" の DNS レコードを"`dns-server.com`" で"a" か"mx" か"any" かのレコードに関して確認します。
`iptables -L -n`	パケットフィルターの確認
`ss -a`	オープンポートの発見
`ss -l`	聴取中のポートの発見
`ss -l -t -n`	聴取中の TCP ポートの発見 (数字)
`dlint example.com`	"`examle.com`" の DNS ゾーン情報を確認

Table 5.4: 低レベルネットワークコマンドのリスト

ティップ
これらの低レベルネットワーク設定ツールは”/usr/sbin/” 中にあります。”/usr/sbin/ifconfig” 等のような完全コマンドパスを使うか、”~/.bashrc” 中の”$PATH” リストに”/usr/sbin” を追加する必要があるかもしれません。

5.6 ネットワークの最適化

一般的なネットワークの最適化は本書の射程外です。ここでは消費者用の接続に関する課題にのみ触れます。

パッケージ	ポプコン	サイズ	説明
iftop	V:6, I:83	93	ネットワークインターフェースの帯域利用情報を表示
iperf	V:2, I:31	23	インターネットプロトコルのバンド幅測定ツール
ifstat	V:0.5, I:5.1	52	インターフェース統計モニター
bmon	V:2, I:19	154	可搬型バンド幅モニター兼速度推定機
ethstatus	V:0.2, I:2.3	41	ネットワークデバイスのスループットを迅速に測定するスクリプト
bwm-ng	V:1.2, I:8.8	95	簡単軽量のコンソール式のバンド幅モニター
ethstats	V:0.06, I:0.35	21	コンソール式のイーサーネット統計モニター

Table 5.5: ネットワーク最適化ツールのリスト

5.6.1 最適 MTU の発見

NM は普通最適の最大送信単位 (MTU: Maximum Transmission Unit) を自動的に設定します。

場合によっては、ping(8) を”-M do” オプションとともに用いて多くのデーターパケットサイズの ICMP パケットを送る実験後 MTU を手動で設定したいと考えるかもしれません。MTU は IP フラグメンテーションを起こさずおくれる最大のデーターパケットサイズに IPv4 の場合は 28 バイト IPv6 の場合は 48 バイト足したものです。例えば以下では IPv4 接続の場合は 1460 と IPv6 接続の場合は 1500 と突き止めます。

```
$ ping -4 -c 1 -s $((1500-28)) -M do www.debian.org
PING  (149.20.4.15) 1472(1500) bytes of data.
ping: local error: message too long, mtu=1460

---  ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

$ ping -4 -c 1 -s $((1460-28)) -M do www.debian.org
PING  (130.89.148.77) 1432(1460) bytes of data.
1440 bytes from klecker-misc.debian.org (130.89.148.77): icmp_seq=1 ttl=50 time=325 ms

---  ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 325.318/325.318/325.318/0.000 ms
$ ping -6 -c 1 -s $((1500-48)) -M do www.debian.org
PING www.debian.org(mirror-csail.debian.org (2603:400a:ffff:bb8::801f:3e)) 1452 data bytes
1460 bytes from mirror-csail.debian.org (2603:400a:ffff:bb8::801f:3e): icmp_seq=1 ttl=47  ↩
    time=191 ms

--- www.debian.org ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 191.332/191.332/191.332/0.000 ms
```

このプロセスはパス MTU (PMTU) 発見 (RFC1191) で、tracepath(8) コマンドで自動化できます。

ネットワーク環境	**MTU**	理由
ダイヤルアップ接続 (IP: PPP)	576	標準
イーサーネット接続 (IP: DHCP または固定)	1500	標準かつデフォールト

Table 5.6: 最適 MTU 値の基本的なガイドライン

これらの基本的なガイドラインに加えて、次を覚えておきます。

- 何らかのトンネル手法 (VPN 等) を使うと、それらのオーバーヘッドのために最適 MTU を更に減らすかもしれません。
- MTU 値は実験的に決定される PMTU 値を越すべきではありません。
- もし他の制約条件を満たすなら、MTU 値は一般的に大きい方がいいです。

最大セグメントサイズ (MSS) はパケットサイズの代替尺度として使われます。MSS と MTU の関係は次です。

- IPv4 では MSS = MTU - 40
- IPv6 では MSS = MTU - 60

注意
iptables(8) (項 5.7 を参照下さい) を使う最適化は MSS を使ってパケットサイズを制約できるのでルーターとして有用です。iptables(8) 中の”TCPMSS” を参照下さい。

5.6.2 WAN TCP の最適化

現代的な高帯域でレイテンシーの大きな WAN では、TCP のスループットは TCP バッファーサイズパラメーターを”TCP tuning” にある手順で調整することで最大化できます。今のところ現在の Debian のデフォールトは高速の 1G bps の FTTP サービスでつながっている私の LAN でも十分機能しています。

5.7 Netfilter インフラ

Netfilter は Linux カーネルのモジュール (項 3.10 を参照下さい) を利用するステートフルファイアーウォールとネットワークアドレス変換 (NAT) のインフラを提供します。

`nftables` パッケージ提供の、比較的新しいユーザースペース nft(8) コマンドは、The netfilter.org ”nftables” project を参照ください。(Linux kernel >= 3.13)

`iptables` や類似パッケージ提供の、比較的古いユーザースペース iptables(8) や類似コマンドは、The netfilter.org ”iptables” project を参照ください。(Linux kernel >= 2.4.x)

シェルから対話形式で手動で netfilter を設定し、その状態を”`nft list ruleset > file`” や iptables-save(8) で保存し、”`nft -f file`” や iptables-restore(8) を使って init スクリプト経由でシステムのリブート時に回復できます。

shorewall のような設定ヘルパースクリプトはこの過程を簡単にします。

Netfilter Documentation (または”`/usr/share/doc/iptables/html/`” 中) の文書を参照下さい。

- Linux Networking-concepts HOWTO

パッケージ	ポプコン	サイズ	説明
nftables	V:301, I:889	189	パケットフィルターと NAT のための管理ツール (Netfilter) ({ip,ip6,arp,eb}tables の後継)
iptables	V:402, I:635	2408	netfilter の管理ツール (IPv4 用の iptables(8)、IPv6 用の ip6tables(8))
arptables	V:0.1, I:1.6	102	netfilter の管理ツール (ARP 用の arptables(8))
ebtables	V:14, I:22	276	netfilter の管理ツール (Ethernet ブリッジング用の ebtables(8))
iptstate	V:0.1, I:1.6	130	netfilter の状態を常時モニター (top(1) と類似)
ufw	V:87, I:110	862	Uncomplicated Firewall (UFW) netfilter ファイアーウォールの管理プログラム
gufw	V:5, I:10	3682	Uncomplicated Firewall (UFW) 用のグラフィカルユーザーインターフェース
firewalld	V:16, I:22	2581	firewalld はネットワークゾーンをサポートする動的管理ファイアーウォールプログラム
firewall-config	V:0.8, I:3.0	1096	firewalld 用のグラフィカルユーザーインターフェース
shorewall-init	V:0.13, I:0.31	88	Shoreline ファイアーウォール初期化
shorewall	V:2.2, I:4.5	3090	Shoreline ファイアーウォール、netfilter 設定ファイル生成システム
shorewall-lite	V:0.03, I:0.05	71	Shoreline ファイアーウォール、netfilter 設定ファイル生成システム (軽装備バージョン)
shorewall6	V:0.7, I:1.2	1334	Shoreline ファイアーウォール、netfilter 設定ファイル生成システム (IPv6 バージョン)
shorewall6-lite	V:0.01, I:0.01	71	Shoreline ファイアーウォール、netfilter 設定ファイル生成システム (IPv6 軽装備バージョン)

Table 5.7: ファイアーウォールツールのリスト

- Linux 2.4 Packet Filtering HOWTO
- Linux 2.4 NAT HOWTO
- nftables HOWTO

Chapter 6

ネットワークアプリケーション

ネットワーク接続を確立した (第 5 章を参照下さい) あとで、各種のネットワークアプリケーションを実行できます。

ティップ
現代的な Debian に特化したネットワークインターフェースのガイドは、The Debian Administrator's Handbook —Network Infrastructure を参照下さい。

ティップ
もしどこかの ISP で"2 段階認証" を有効にした場合、あなたのプログラムから POP や SMTP サービスにアクセスするアプリケーションパスワードを入手する必要があります。事前にあなたのホスト IP を許可する必要があるかもしれません。

6.1 ウェッブブラウザー

多くのウェッブブラウザーパッケージがハイパーテキストトランスファープロトコル (HTTP) を使って遠隔コンテントにアクセスするために存在します。

6.1.1 User-Agent 文字列をスプーフィングする

一部の過剰な制約を課すウエッブサイトにアクセスするには、ウエッブブラウザープログラムが返す User-Agent 文字列をスプーフィングする必要があるかもしれません。以下を参照下さい:

- MDN Web Docs: userAgent
- Chrome Developers: Override the user agent string
- How to change your user agent
- How to Change User-Agent in Chrome, Firefox, Safari, and more
- How to Change Your Browser's User Agent Without Installing Any Extensions
- How to change the User Agent in Gnome Web (epiphany)

パッケージ	ポプコン	サイズ	タイプ	ウェッブブラウザーの説明
chromium	V:37, I:101	324490	X	Chromium、(Google からのオープンオースブラウザー)
firefox	V:15, I:21	317809	, ,	Firefox, (Mozilla からのオープンソースのブラウザー、Debian Unstable でのみ入手可能)
firefox-esr	V:188, I:423	299668	, ,	Firefox ESR, (Firefox 延長サポートリリース)
epiphany-browser	V:2.0, I:9.7	6737	, ,	GNOME、HIG 準拠、Epiphany
konqueror	V:26, I:110	8191	, ,	KDE、Konqueror
dillo	V:0.5, I:3.8	2203	, ,	Dillo, (軽量ブラウザー, FLTK 準拠)
w3m	V:11, I:125	2853	テキスト	w3m
lynx	V:30, I:454	2031	, ,	Lynx
elinks	V:2, I:14	1839	, ,	ELinks
links	V:2, I:19	2344	, ,	Links (テキストのみ)
links2	V:0.8, I:9.2	5497	グラフィクス	Links (X を使わないコンソールグラフィクス)

Table 6.1: ウェッブブラウザーのリスト

6.1.2 ブラウザー拡張

全ての現代的な GUI ブラウザーはソースコードを使うブラウザー拡張をサポートしていて、ウエッブ拡張として標準化されつつあります。

6.2 メールシステム

本セクションは消費者用インターネット接続上の典型的モービルワークステーションにフォーカスします。



注意
もしインターネットと直接メール交換するメールサーバーを設定するなら、このような初歩的文書が不要なぐらいシステムを熟知しているべきです。

6.2.1 E メールの基本

email メッセージは、メッセージのエンベロープ (封筒) と、メッセージのヘッダーと、メッセージの本体との、3 構成要素から成り立っています。

- メッセージエンベロープ中の”To” (宛先) と”From” (差出人) 情報は SMTP が電子メールを配達するのに用いられます。(メッセージエンベロープの”From” 情報はバウンスアドレス、From_、等とも呼ばれます。)
- メッセージヘッダー中の”To” (宛先) と”From” (差出人) 情報は email クライアントが email を表示するのに用いられます。(これらはメッセージエンベロープの情報と共通のことがよくありますが、必ずしもそうとは限りません。)
- ヘッダーやボディーデーターを包含する電子メールのメッセージの書式は、多目的インターネットメール拡張 (MIME) を持ちいて、プレーンな ASCII テキストから他の文字エンコーディングに、またオーディオやビデオや画像アプリケーションプログラムに拡張されています。

フル機能の GUI を使った電子メールクライアントは GUI を使った直感的な設定を使い以下の全機能を提供します。

- コンテントのデータータイプやエンコーディングを扱いメッセージヘッダーやボディーのデーターは多目的インターネットメール拡張 (MIME) を持ちいて解釈されます。
- 旧来の基本アクセス認証か現代的な OAuth 2.0 を用いて ISP の SMTP や IMAP サーバーから認証をうけます。(OAuth 2.0 に関しては、デスクトップ環境経由で設定します。例えば”Settings” -> ”Online Accounts”.)
- メッセージサブミッションポート (587) を聞いている ISP のスマートホスト SMTP サーバーにメッセージを送ります。
- TLS/IMAP4 ポート (993) から ISP のサーバー上に保存されたメッセージを受け取ります。
- 属性によってメールのフィルタリングができます。
- 連絡先、カレンダー、タスク、メモといった追加の機能を提供することがあります。

パッケージ	ポプコン	サイズ	タイプ
evolution	V:27, I:227	521	X GUI プログラム (GNOME、グループウエアスイート)
thunderbird	V:39, I:100	304278	X GUI プログラム (GTK、Mozilla Thunderbird)
kmail	V:43, I:101	25384	X GUI プログラム (KDE)
mutt	V:11, I:78	7378	きっと vim とともに使われるキャラクターターミナルプログラム
mew	V:0.01, I:0.14	2319	(x)emacs の下でキャラクターターミナルプログラム

Table 6.2: メールユーザーエージェント (MUA) のリスト

6.2.2 現代的なメールサービスの制約

スパム（迷惑メール）問題にさらされるのを最小化するために、現代的なメールサービスには様々な制約があります。

- 確実にメールをリモートホストに直接送るために消費者用インターネット接続上で SMTP サーバーを実行するのは現実的ではありません。
- メールは可能な限り真正に見えない限り、送付先に到達する途中のどこかのホストによって黙って拒否されるでしょう。
- 無関係の複数の送信元メールアドレスのメールを、単一のスマートホストを使って確実にリモートホストに送ることを期待するのは現実的ではありません。

なぜなら:

- 消費者用インターネット接続からインターネットへの SMTP ポート (25) 接続はブロックされます。
- 消費者用インターネット接続されたホストへのインターネットからの SMTP ポート (25) 接続はブロックされます。
- 消費者用インターネット接続されたホストからインターネットへの送信メッセージはメッセージサブミッションポート (587) 経由でのみ送れます。
- ドメインキーアイデンティファイドメール (DKIM) や SPF 認証やドメインベースのメッセージ認証、報告および適合 (DMARC) のようなアンチスパムテクニックが email のフィルタリングに広範に使用されています。
- ドメインキーアイデンティファイドメールサービスがあなたのメールをスマートホスト経由で送信する際に提供されているかもしれません。
- E メールアドレスのなりすましを防ぐために、スマートホストによってメッセージヘッダ内の送信元メールアドレスがスマートホストのメールアカウントに書き換えられることがあります。

6.2.3 歴史的なメールサービスへの期待

Debian 上のいくつかのプログラムは、UNIX システム上のメールサービスが歴史的に以下のように機能したため、デフォルト設定でもカスタマイズ設定でも `/usr/sbin/sendmail` コマンドにアクセスして以下のようにして電子メールを送ることを期待します:

- 電子メールがテキストファイルで作成されます。
- 電子メールが `/usr/sbin/sendmail` コマンドに引き渡されます。
- 同一ホスト上の送り先アドレスの場合、`/usr/sbin/sendmail` コマンドは `/var/mail/$username` ファイルに電子メールを追記することで電子メールのローカル配達をします。
  - このような機能を期待しているコマンド: `apt-listchanges`, `cron`, `at`, ...
- リモートホスト上の送り先アドレスの場合、`/usr/sbin/sendmail` コマンドは DNS の MX レコードから見つかる送り先ホストに SMTP を用いて電子メールを配達をします。
  - このような機能を期待しているコマンド: `popcon`, `reportbug`, `bts`, ...

6.2.4 メール転送エージェント (MTA)

Debian のモービルワークステーションは、Debian 12 Bookworm 以降メール転送エージェント (MTA) を使わず、フル機能の GUI を使った電子メールクライアントだけで設定可能です。

Debian は伝統的に、`/usr/sbin/sendmail` コマンドを期待するプログラムのために何らかの MTA プログラムをインストールしました。モービルワークステーション上のそのような MTA は項 6.2.2 や項 6.2.3 に対処しなければいけません。

モービルワークステーションでは、MTA の典型的選択肢はインストールオプションとして”Mail sent by smarthost; received via SMTP or fetchmail” 等を選択した `exim4-daemon-light` か `postfix` です。これらの軽量 MTA は”`/etc/aliases`” を尊重します。

ティップ
`exim4` を設定して複数の送信元メールアドレスに対応する複数のスマートホストを経由してインターネットメールを送ることは簡単ではありません。特定のプログラムためにそのような機能が必要な場合には、複数の送信元アドレスを簡単に設定できる `msmtp` を設定し使用しましょう。そして、MTA は単一送信元アドレスだけのままにしましょう。

6.2.4.1 exim4 設定

スマートホスト経由のインターネットメールに関しては、`exim4-*` パッケージを以下のように (再) 設定します。

```
$ sudo systemctl stop exim4
$ sudo dpkg-reconfigure exim4-config
```

”General type of mail configuration” に関して、” スマートホストでメール送信; SMTP または fetchmail で受信する” を選択します。

”System mail name:” をそのデフォルトである FQDN (項 5.1.1 を参照下さい) に設定します。

”IP-addresses to listen on for incoming SMTP connections:” をそのデフォルトである”127.0.0.1 ; ::1” と設定します。

”Other destinations for which mail is accepted:” の内容を消去します。

”Machines to relay mail for:” の内容を消去します。

” 送出スマートホストの IP アドレスまたはホスト名:” を”smtp.hostname.dom:587” と設定します。

パッケージ	ポプコン	サイズ	説明
exim4-daemon-light	V:197, I:206	1777	Exim4 メール転送エージェント (MTA: Debian のデフォルト)
exim4-daemon-heavy	V:4.5, I:4.8	1955	Exim4 メール転送エージェント (MTA: 柔軟な代替候補)
exim4-base	V:201, I:211	1687	Exim4 文書 (text) と共通ファイル
exim4-doc-html	I:0.94	3809	Exim4 文書 (html)
exim4-doc-info	I:0.51	649	Exim4 文書 (info)
postfix	V:99, I:104	4277	Postfix メール転送エージェント (MTA: セキュアな代替候補)
postfix-doc	I:4.0	5030	Postfix 文書 (html+text)
sasl2-bin	V:4.5, I:9.4	370	Cyrus SASL API の実装 (SMTP AUTH について postfix を補完)
cyrus-sasl2-doc	I:0.68	2140	Cyrus SASL - 文書
msmtp	V:10, I:15	822	軽量 MTA
msmtp-mta	V:7.8, I:9.8	139	軽量 MTA (msmtp の sendmail 互換性拡張)
nullmailer	V:7.2, I:7.6	483	超軽量 MTA、ローカルメール無し
ssmtp	V:3.8, I:5.6	134	超軽量 MTA、ローカルメール無し
sendmail-bin	V:9.4, I:9.7	1954	高機能 MTA (既に慣れている場合)
git-email	V:0.4, I:8.6	1229	パッチの集合の email として送信する git-send-email(1) プログラム

Table 6.3: 基本的なメール転送エージェント関連パッケージのリスト

”Hide local mail name in outgoing mail?” に対して”No” を選択します。(この代わりに、項 6.2.4.3 にある”/etc/email-addre
を使用します。)

”DNS クエリの数を最小限に留めますか (ダイヤルオンデマンド)?” に以下の内のひとつの返答をします。

- ブート時にインターネットに接続されている場合は、”No” とします。
- ブート時にインターネットに接続されていない場合は、”Yes” とします。

”Delivery method for local mail:” を”mbox format in /var/mail/” と設定します。

”Split configuration into small files?:” に対して”Yes” を選択します。

”/etc/exim4/passwd.client” を編集しスマートホストのためのパスワードエントリーを作成します。

```
$ sudo vim /etc/exim4/passwd.client
 ...
$ cat /etc/exim4/passwd.client
^smtp.*\.hostname\.dom:username@hostname.dom:password
```

”/etc/default/exim4” 中で”QUEUERUNNER='queueonly'” や”QUEUERUNNER='nodaemon'” 等と設定しシステムリソースの消費を最小限とした exim4(8) (optional)

以下のようにして exim4 を起動します。

```
$ sudo systemctl start exim4
```

”/etc/exim4/passwd.client” 中のホスト名はエイリアスであってはいけません。真のホスト名は以下のようにして確認できます。

```
$ host smtp.hostname.dom
smtp.hostname.dom is an alias for smtp99.hostname.dom.
smtp99.hostname.dom has address 123.234.123.89
```

エイリアス問題を回避するために”/etc/exim4/passwd.client” の中に正規表現を用いています。もし ISP がエイリアスで示されるホストを移動させても SMTP AUTH はおそらく動きます。

以下のようにすれば exim4 の設定を手動で更新できます。

- ”/etc/exim4/” 中の exim4 設定ファイルの更新。
  - MACRO を設定するために”/etc/exim4/exim4.conf.localmacros” を作成し、”/etc/exim4/exim4.conf.tem を編集します。(非分割設定)
  - ”/etc/exim4/exim4.conf.d” サブディレクトリー中で、新規ファイルを作成したり既存ファイルを編集したりします。(分割設定)
- ”systemctl reload exim4” を実行します。



注意
”DNS クエリの数を最小限に留めますか (ダイヤルオンデマンド)?” という debconf の質問に”No” (デフォールト値) が選ばれシステムがブート時にインターネットに繋がっていない場合、exim4 の起動は長い時間がかかります。

次に示す正式のガイドを読んで下さい: ”/usr/share/doc/exim4-base/README.Debian.gz” と update-exim4.conf(8)



警告
現実的な各種配慮から、ポート 587 上で STARTTLS を用い SMTP を用いるか、ポート 25 上のプレーンの SMTP の代わりのポート 465 上の SMTPS (SSL 使用の SMTP) を用いています。

6.2.4.2 SASL を使う postfix の設定

スマートホスト経由のインターネットメールに関しては postfix 文書と重要マニュアルページを読むことから始めるべきです。

コマンド	機能
postfix(1)	Postfix コントロールプログラム
postconf(1)	Postfix の設定ユーティリティー
postconf(5)	Postfix 設定パラメーター
postmap(1)	Postfix 検索テーブルのメンテナンス
postalias(1)	Postfix エイリアスデーターベースのメンテナンス

Table 6.4: 重要 postfix マニュアルページのリスト

postfix と sasl2-bin パッケージを以下のように (再) 設定します。

```
$ sudo systemctl stop postfix
$ sudo dpkg-reconfigure postfix
```

” スマートホストを使ってインターネット” を選択します。

”SMTP リレーホスト (なければ空):” を”[smtp.hostname.dom]:587” と設定します。

```
$ sudo postconf -e 'smtp_sender_dependent_authentication = yes'
$ sudo postconf -e 'smtp_sasl_auth_enable = yes'
$ sudo postconf -e 'smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd'
$ sudo postconf -e 'smtp_sasl_type = cyrus'
$ sudo vim /etc/postfix/sasl_passwd
```

スマートホストのパスワードエントリーを作成します。

```
$ cat /etc/postfix/sasl_passwd
[smtp.hostname.dom]:587     username:password
$ sudo postmap hush:/etc/postfix/sasl_passwd
```

次に記すように postfix を起動します。

```
$ sudo systemctl start postfix
```

dpkg-reconfigure ダイアログと”/etc/postfix/sasl_passwd” の中で”[” と”]” を使うことで MX レコードを確認せずに指定された hostname その物を直接使うように確実にします。”/usr/share/doc/postfix/html/SASL_REA の中の”Enabling SASL authentication in the Postfix SMTP client” を参照下さい。

6.2.4.3 メールアドレス設定

メールのトランスポートとデリバリーとユーザーのエージェントが使うメールアドレス設定ファイルが少々存在します。

ファイル	機能	アプリケーション
/etc/mailname	(送出) メールのデフォールトのホスト名	Debian 固有、mailname(5)
/etc/email-addresses	送出メールのホスト名の偽装	exim(8) 固有、exim4-config_files(5)
/etc/postfix/generic	送出メールのホスト名の偽装	postfix(1) 固有、postmap(1) コマンド実行後アクティベートされます。
/etc/aliases	受入メールのためのアカウント名のエイリアス	一般的、newaliases(1) コマンド実行後アクティベートされます。

Table 6.5: メールアドレス関連のファイルのリスト

通常”/etc/mailname” ファイル中の **mailname** はホストの IP の一つとして解決できる完全修飾ドメイン名 (FQDN) です。解決できる IP アドレスのあるホスト名を持たない可動ワークステーションの場合には、この **mailname** を”hostname -f” に設定します。(これは exim4-* と postfix の両方に有効な安全な選択肢です。)

ティップ
”/etc/mailname” の内容は多くの MTA 以外のプログラムによってそのデフォールト挙動のために使われます。mutt の場合、~/muttrc ファイル中の”hostname” と”from” 変数を設定して **mailname** の値をオーバーライドします。bts(1) や dch(1) 等の devscripts パッケージ中のプログラムの場合、環境変数の”$DEBFULLNAME” や”$DEBEMAIL” をエキスポートしてその値をオーバーライドします。

ティップ
普通 popularity-contest パッケージは root アカウントからメールを FQDN 付きで送信します。/usr/share/popularity-contest/default.conf に記載された様に /etc/popularity-contest.conf 中に MAILFROM を設定する必要があります。こうしないと、smarthost の SMTP サーバーによってあなたのメールは拒否されます。少々面倒ですが、root からの全メールの発信元を書き替えるより、この方法は安全ですし、他のデーモンや cron スクリプトに関してもこの方法を適用するべきです。

---

mailname を”hostname -f” と設定した時には、次によって MTA で発信元メールアドレスを偽装することが実現できます。

- exim4(8) の場合、exim4-config_files(5) に説明されているように”/etc/email-addresses”
- postfix(1) の場合、generic(5) に説明されているように”/etc/postfix/generic”

postfix の場合、次に記す追加ステップが必要です。

```
# postmap hash:/etc/postfix/generic
# postconf -e 'smtp_generic_maps = hash:/etc/postfix/generic'
# postfix reload
```

あなたのメール設定は以下のようにするとテストできます。

- exim(8) の場合、-brw, -bf, -bF, -bV, …オプションを使用
- postmap(1) の場合、-q オプションを使用

ティップ
Exim には exiqgrep(8) や exipick(8) のようないくつかのユーティリティープログラムが同梱されています。利用可能なコマンドは"dpkg -L exim4-base|grep man8/" を参照下さい。

6.2.4.4 基本的な **MTA** の操作

基本的な MTA 操作が存在します。その一部は sendmail(1) 互換性インターフェース経由で実行する事もできます。

exim コマンド	**postfix** コマンド	説明
sendmail	sendmail	標準入力からメールを読み配送を手配 (-bm)
mailq	mailq	メールキューを状態とキュー ID とともにリスト (-bp)
newaliases	newaliases	エイリアスデーターベースを初期化 (-I)
exim4 -q	postqueue -f	待機メールを排出 (-q)
exim4 -qf	postsuper -r ALL deferred; postqueue -f	全メールを排出
exim4 -qff	postsuper -r ALL; postqueue -f	凍結メールをも排出
exim4 -Mg queue_id	postsuper -h queue_id	キュー ID によってメッセージを凍結
exim4 -Mrm queue_id	postsuper -d queue_id	キュー ID によってメッセージを削除
N/A	postsuper -d ALL	全メッセージを削除

Table 6.6: 基本的 MTA 操作のリスト

ティップ
"/etc/ppp/ip-up.d/*" 中のスクリプトで全メールを排出するのは良い考えかも知れません。

6.3 リーモートアクセスサーバーとユーティリティー (SSH)

セキュアーシェル (SSH) はインターネット経由で接続するセキュアーな方法です。Debian では、OpenSSH と呼ばれるフリーバージョンの SSH が openssh-client と openssh-server パッケージとして利用可能です。

ssh(1) はユーザーにとってより賢明でよりセキュアーな telnet(1) として機能します。telnet コマンドと異なり、ssh コマンドは telnet エスケープ文字 (初期デフォールト CTRL-]) に出会うことで中断される事がありません。

shellinabox は SSH プログラムではありませんが、リモートターミナルアクセスのための興味深い代替策としてここにリストしています。

リモート X クライアントプログラムへの接続のために項 7.9 も参照下さい。

パッケージ	ポプコン	サイズ	ツール	説明
openssh-client	V:709, I:997	3611	ssh(1)	セキュアーシェルクライアント
openssh-server	V:781, I:824	3615	sshd(8)	セキュアーシェルサーバー
ssh-askpass	I:15	103	ssh-askpass(1)	ユーザーに ssh-add 用のパスフレーズを尋ねる (プレーン X)
ssh-askpass-gnome	V:0.4, I:2.9	46	gnome-ssh-askpass(1)	ユーザーに ssh-add 用のパスフレーズを尋ねる (GNOME)
ssh-askpass-fullscreen	V:0.08, I:0.36	41	ssh-askpass-fullscreen(1)	見た目良くユーザーに ssh-add 用のパスフレーズを尋ねる (GNOME)
shellinabox	V:0.60, I:0.88	528	shellinaboxd(1)	ブラウザーアクセス可能な VT100 ターミナルエミュレーターのためのウエッブサーバー

Table 6.7: リモートアクセスサーバーとユーティリティーのリスト



注意
あなたの SSH がインターネットからアクセスできる場合には、項 4.6.3 を参照下さい。

ティップ
リモートのシェルプロセスが回線接続の中断の際にも継続するようにするために screen(1) プログラムを使いましょう (項 9.1.2 を参照下さい)。

6.3.1 SSH の基本

OpenSSH SSH デーモンは SSH プロトコル 2 のみをサポートします。

”`/usr/share/doc/openssh-client/README.Debian.gz`” と ssh(1) と sshd(8) と ssh-keygen(1) と ssh-add(1) と ssh-agent(1) を参照下さい。



警告
OpenSSH サーバーを実行したい場合には、”`/etc/ssh/sshd_not_to_be_run`” が存在してはいけません。
rhost に基づく認証を有効化してはいけない (`/etc/ssh/sshd_config` 中の `HostbasedAuthentication`)。

クライアントから ssh(1) 接続を開始するには以下のようにします。

6.3.2 リモートホストでのユーザ名

ローカルホストとリモートホストで同じユーザ名を使っている場合は”`username@`” と打たなくてもよいです。

たとえローカルとリモートで異なるユーザー名を使う場合でも、”`~/.ssh/config`” を用いるとユーザー名を省略できます。例えば Debian Salsa サービスでのユーザー名が”`foo-guest`” の場合は、”`~/.ssh/config`” が次を含むように設定します。

設定ファイル	設定ファイルの説明
`/etc/ssh/ssh_config`	SSH クライアントのデフォールト、ssh_config(5) 参照下さい
`/etc/ssh/sshd_config`	SSH サーバーのデフォールト、sshd_config(5) 参照下さい
`~/.ssh/authorized_keys`	当該 SSH サーバーの当該アカウント接続用にクライアントが使用するデフォールト公開 SSH キー
`~/.ssh/id_rsa`	ユーザーの秘密 SSH-2 RSA キー
`~/.ssh/id_`*`key-type-name`*	ユーザの秘密 SSH-2 *key-type-name* キー (`ecdsa`, `ed25519` など)

Table 6.8: SSH 設定ファイルのリスト

コマンド	説明
`ssh username@hostname.domain.ext`	デフォルトモードで接続
`ssh -v username@hostname.domain.ext`	デバッグメッセージを有効にしてデフォルトモードで接続
`ssh -o PreferredAuthentications=password username@hostname.domain.ext`	SSH バージョン 2 でパスワードを使うことを強制
`ssh -t username@hostname.domain.ext passwd`	リモートホストでパスワードを更新するために `passwd` プログラムを実行

Table 6.9: SSH クライアント起動例のリスト

```
Host salsa.debian.org people.debian.org
User foo-guest
```

6.3.3 リモートパスワード無しでの接続

”`PubkeyAuthentication`” (SSH-2 プロトコル) を使うと、リモートシステムのパスワードを覚えなくてもよくなります。

リモートシステムの”`/etc/ssh/sshd_config`” 中に対応する項目”`PubkeyAuthentication yes`” を設定します。

次に示すように、ローカルで認証キーを生成しリモートシステム上に公開キーをインストールします。

```
$ ssh-keygen -t rsa
$ cat .ssh/id_rsa.pub | ssh user1@remote "cat - >>.ssh/authorized_keys"
```

ホストを制限したり特定コマンドを実行したりするには”`~/.ssh/authorized_keys`” 中の項目にオプションを追加します。sshd(8) の”AUTHORIZED_KEYS FILE FORMAT” を参照下さい。

6.3.4 外部 SSH クライアントへの対処法

他のプラットフォーム上で利用可能なフリーな SSH クライアントがいくつかあります。

6.3.5 ssh-agent の設定

SSH の認証キーをパスフレーズで保護する方がより安全です。もしパスフレーズが設定されていない場合には”`ssh-keygen -p`” で設定できます。

環境	フリーの **SSH** プログラム
Windows	puTTY (PuTTY: 自由な SSH と Telnet クライアント) (GPL)
Windows (cygwin)	cygwin 中の SSH (Cygwin: Windows 上での Linux 感覚を得よう) (GPL)
Mac OS X	OpenSSH; ターミナルアプリケーションの ssh を使用します (GPL)

Table 6.10: 他のプラットフォーム上で使えるフリーな SSH クライアントのリスト

上記のようにパスワードを使って接続したリモートホスト上の"~/.ssh/authorized_keys" 中にあなたの公開 SSH キー (例えば"~/.ssh/id_rsa.pub") を設定します。

```
$ ssh-agent bash
$ ssh-add ~/.ssh/id_rsa
Enter passphrase for /home/username/.ssh/id_rsa:
Identity added: /home/username/.ssh/id_rsa (/home/username/.ssh/id_rsa)
```

今後、次のコマンドにリモートパスワードは必要ありません。

```
$ scp foo username@remote.host:foo
```

ssh-agent のセッションを終了するのに ^D を押します。

X サーバーの場合、普通の Debian の起動スクリプトは親プロセスとして ssh-agent を実行します。だから ssh-add は 1 回だけ実行すれば十分です。詳細は ssh-agent(1) と ssh-add(1) を参照下さい。

6.3.6 リモートホストからメールを送信する

適切な DNS 設定がなされたサーバに SSH アカウントを持っている場合、リモートサーバから真に送られたメールとして、ワークステーションからメールを送信することができます。

```
$ ssh username@example.org /usr/sbin/sendmail -bm -ti -f "username@example.org" < mail_data ↩
    .txt
```

6.3.7 SMTP/POP3 トンネルをするためのポートフォワーディング

ssh を通して localhost のポート 4025 から remote-server のポート 25 へと、localhost のポート 4110 から remote-server のポート 110 へと接続するパイプを設定するには、ローカルホスト上で以下のように実行します。

```
# ssh -q -L 4025:remote-server:25 4110:remote-server:110 username@remote-server
```

このようにするとインターネット経由で SMTP/POP3 サーバーへとセキュアーに接続できます。リモートホストの"/etc/ssh/sshd_config" 中の"AllowTcpForwarding" エントリーを"yes" と設定します。

6.3.8 SSH 上のリモートシステムをシャットダウンする方法

"shutdown -h now" (項 1.1.8 を参照下さい) を実行しているプロセスを at(1) コマンド (項 9.4.13 を参照下さい) を使って以下のようにして SSH が終了することから守る必要があります。

```
# echo "shutdown -h now" | at now
```

"shutdown -h now" を screen(1) (項 9.1.2 を参照下さい) セッション中で実行しても同様のことができます。

6.3.9 SSH のトラブルシュート

問題があるときは設定ファイルのパーミッションを確認し、ssh を”-v” オプションとともに実行します。

root でファイアーウォールと問題を起こした場合には、”-p” オプションを使います; こうするとサーバーポートの 1 —1023 を使うのを回避します。

リモートサイトへの ssh 接続が急に動作し無くなった際は、システム管理者による変更、特に可能性が高いのはシステムメンテナンス中に”host_key” が変更された結果かもしれません。実際にこういう状況で誰も洒落たハックでリモートホストとしてなりすまそうとしていないことを確認した後に、”host_key” エントリーをローカルホストの”~/.ssh/known_hosts” から削除すると再び接続できるようになります。

6.4 プリントサーバーとユーティリティー

旧来の Unix 的システムでは BSD のラインプリンターデーモン (lpd) が標準で、古典的フリーソフトウェアーの標準プリント出力フォーマットは PostScript (PS) でした。Ghostscript とともに何らかのフィルターシステムを使うことで non-PostScript プリンターへの印刷が可能になっていました。項 11.4.1 を参照下さい。

現代的な Debian システムでは Common UNIX Printing System (CUPS) がデファクトスタンダードで、現代的なフリーソフトの標準プリント出力フォーマットは Portable Document Format (PDF) です。

CUPS は、インターネット印刷プロトコル (IPP) を使います。IPP は新たなクロスプラットフォームの両方向通信能力のあるリモート印刷のデファクト標準となっています。

CUPS システムのファイルフォーマット依存の自動変換機能のおかげで、どんなデーターでも lpr コマンドに供給すると期待される印刷出力が生成されます。(CUPS では、lpr は cups-bsd パッケージをインストールすると有効となります。)

Debian システムには、プリントサーバーやユーティリティーで留意すべきパッケージがいくつかあります。

パッケージ	ポプコン	サイズ	ポート	説明
lpr	V:1.9, I:2.2	378	printer (515)	BSD lpr/lpd (ラインプリンターデーモン)
cups	V:141, I:444	1102	IPP (631)	インターネット印刷 CUPS サーバー
cups-client	V:149, I:455	432	, ,	CUPS 用 System V プリンターコマンド: lp(1) と lpstat(1) と lpoptions(1) と cancel(1) と lpmove(8) と lpinfo(8) と lpadmin(8) 等
cups-bsd	V:33, I:178	131	, ,	CUPS 用 BSD プリンターコマンド: lpr(1) と lpq(1) と lprm(1) と lpc(8) 等
printer-driver-gutenprint	V:10, I:49	1117	非該当	CUPS 用のプリンタードライバー

Table 6.11: プリントサーバーとユーティリティーのリスト

ティップ
CUPS システムはウェッブブラウザーを”http://localhost:631/” に向けることで設定できます。

6.5 他のネットワークアプリケーションサーバー

他のネットワークアプリケーションサーバーを次に示します。

パッケージ	ポプコン	サイズ	プロトコル	説明
inetutils-telnetd	V:0.2, I:1.3	200	TELNET	TELNET サーバー
nfs-kernel-server	V:46, I:53	886	NFS	Unix 式ファイル共有
samba	V:112, I:123	5190	SMB	Windows のファイルとプリンター共有
netatalk	V:0.57, I:0.79	933	ATP	Apple/Mac のファイルとプリンター共有 (AppleTalk)
proftpd-core	V:7.6, I:8.4	5041	FTP	汎用ファイルダウンロード
apache2	V:168, I:205	590	HTTP	汎用ウェッブサーバー
squid	V:8.8, I:9.3	9585	, ,	汎用ウェッブプロキシサーバー
bind9	V:32, I:35	905	DNS	他のホストの IP アドレス
kea	I:0.51	245	DHCP	クライアント自身の IP アドレス

Table 6.12: 他のネットワークアプリケーションサーバー

コモンインターネットファイルシステムプロトコル (CIFS) はサーバーメッセージブロック (SMB) と同じプロトコルで Microsoft Windows で広く使われています。

ティップ
サーバーシステムの統合には、項 4.5.2 を参照下さい。

ティップ
ホスト名の解決は通常 DNS サーバーによって提供されます。ホストの IP アドレスが DHCP によって動的にアサインされる場合には Debian wiki 上の DDNS ページに書かれているようにして `bind9` と `kea` を使いホスト名解決のためのダイナミック DNS が設定できます。

ティップ
Debian アーカイブの全内容のローカルのミラーサーバーを使うより、`squid` 等のプロキシサーバーを使う方がはるかにバンド幅を節約上ではるかに効率的です。

6.6 他のネットワークアプリケーションクライアント

他のネットワークアプリケーションクライアントを次に示します。

6.7 システムデーモンの診断

`telnet` プログラムを使うとシステムデーモンへの手動接続とその診断ができます。

プレーンな POP3 サービスをテストするには、以下のようにします。

```
$ telnet mail.ispname.net pop3
```

一部の ISP が提供する TLS/SSL を有効にした POP3 サービスをテストするには、`telnet-ssl` か `openssl` パッケージによる、TLS/SSL を有効にした `telnet` クライアントが必要です。

パッケージ	ポプコン	サイズ	プロトコル	説明
netcat-traditional	V:61, I:905	139	TCP/IP	TCP/IP 用万能ツール (スイス陸軍ナイフ)
netcat-openbsd	V:23, I:118	105	TCP/IP	IPv6 やプロキシや Unix ソケットサポート付きの TCP/IP 用万能ツール (スイス陸軍ナイフ)
openssl	V:853, I:996	2564	SSL	セキュアソケットレイヤー (SSL) のバイナリーと関連する暗号化ツール
stunnel4	V:7, I:10	24	, ,	万能 SSL ラッパー
inetutils-telnet	V:48, I:778	282	TELNET	TELNET クライアント
nfs-common	V:151, I:188	1259	NFS	Unix 式ファイル共有
smbclient	V:29, I:204	2117	SMB	MS Windows のファイルとプリンター共有
cifs-utils	V:34, I:117	369	, ,	リモートの MS Windows ファイルをマウントやアンマウントするコマンド
wget	V:186, I:982	3784	HTTP と FTP	ウェッブダウンローダー
curl	V:296, I:739	526	, ,	, ,
transmission-gtk	V:10, I:135	6639	BitTorrent	BitTorrent クライアント (GTK)
transmission-qt	V:0.6, I:2.5	6522	, ,	BitTorrent クライアント (Qt)
ktorrent	V:1.3, I:4.8	5031	, ,	BitTorrent クライアント (Qt)
qbittorrent	V:8, I:23	16666	, ,	BitTorrent クライアント (Qt)
bind9-host	V:143, I:940	137	DNS	bind9 由来の host(1) コマンド、”Priority: standard”
bind9-dnsutils	V:116, I:836	521	, ,	bind 由来の dig(1) コマンド、”Priority: standard”
ldap-utils	V:11, I:56	791	LDAP	LDAP サーバーからデーター獲得

Table 6.13: 他のネットワークアプリケーションクライアント

```
$ telnet -z ssl pop.gmail.com 995
```

```
$ openssl s_client -connect pop.gmail.com:995
```

以下の RFC は各システムデーモンに関する必要な知見を提供します。

RFC	説明
rfc1939 と rfc2449	POP3 サービス
rfc3501	IMAP4 サービス
rfc2821 (rfc821)	SMTP サービス
rfc2822 (rfc822)	メールファイルフォーマット
rfc2045	Multipurpose Internet Mail Extensions (MIME)
rfc819	DNS サービス
rfc2616	HTTP サービス
rfc2396	URI 定義

Table 6.14: よく使われる RFC のリスト

”`/etc/services`” の中にポートの使用され方が記載されています。

Chapter 7

GUI システム

7.1 GUI デスクトップ環境

Debian システム上のフル機能の GUI デスクトップ環境にはいくつかの選択肢があります。

タスクパッケージ	ポプコン	サイズ	説明
task-gnome-desktop	p:91	9	GNOME デスクトップ環境
task-xfce-desktop	I:88	9	Xfce デスクトップ環境
task-kde-desktop	I:93	6	KDE Plasma デスクトップ環境
task-mate-desktop	I:30	9	MATE デスクトップ環境
task-cinnamon-desktop	I:35	9	Cinnamon デスクトップ環境
task-lxde-desktop	I:18	9	LXDE デスクトップ環境
task-lxqt-desktop	I:16	9	LXQt デスクトップ環境
task-gnome-flashback-desktop	I:4	6	GNOME Flashback デスクトップ環境

Table 7.1: デスクトップ環境のリスト

ティップ
タスクメタパッケージによって選ばれた依存パッケージは、Debian `unstable`/`testing` 環境下で最新のパッケージ移行状態と同期していないかもしれません。`task-gnome-desktop` の場合、以下のようにパッケージ選択を調整する必要があるかもしれません:

- `sudo aptitude -u` として aptitude(8) をスタートします。
- カーサを”Tasks” に移動し”Enter” を押します。
- カーサを”End-user” に移動し”Enter” を押します。
- カーサを”GNOME” に移動し”Enter” を押します。
- カーサを `task-gnome-desktop` に移動し”Enter” を押します。
- カーサを”Depends” に移動し”m” (手動で manually 選択) を押します。
- カーサを”Recommends” に移動し”m” (手動で manually 選択) を押します。
- カーサを”`task-gnome-desktop` に移動し”-” を押します。(drop)
- パッケージ間コンフリクトを引き起こしている問題あるパッケージを落としながら選択されたパッケージを調整します。
- ”g” を押してインストールを開始します。

本章は Debian のデフォルトデスクトップ環境にフォーカスします: wayland 上の GNOME を提供する `task-gnome-desk`

7.2 GUI 通信プロトコル

GNOME デスクトップで使用されうる GUI 通信プロトコル:

- Wayland (ディスプレイサーバプロトコル) (ネイティブ)
- X Window システムコアプロトコル (`xwayland` 経由)

Wayland アーキテクチャーは X Window アーキテクチャーとどう違うのかという freedesktop.org サイトを確認して下さい。

ユーザー視点からは、相違点は以下のように口語的にまとめられます。

- Wayland は同一ホスト上の GUI 通信プロトコル: 新規、簡単、高速、非 setuid root のバイナリー
- X Window はネットワーク対応可能な GUI 通信プロトコル: 伝統的、複雑、低速、setuid root のバイナリー

Wayland プロトコルを使うアプリケーションにとって、その表示内容へのリモートホストからのアクセスは VNC や RDP によって支えられています。項 7.8 を参照下さい。

現代的な X サーバーは MIT 共有メモリー拡張機能があり、ローカルの X クライアントとローカルの共有メモリーを使って通信します。これはネットワーク透過性の Xlib プロセス間通信チャンネルをバイパスし性能が得られるようにしています。この状況がローカル限定の通信プロトコルの Wayland が作られた背景です。

GNOME ターミナルから起動された `xeyes` プログラムを使うことで、各 GUI アプリケーションが使う GUI コミュニケーションプロトコルが確認できます。

```
$ xeyes
```

- Wayland 表示サーバープロトコルを使う”GNOME ターミナル” のようなアプリケーション上にマウスカーソールがある際には、目玉はマウスカーソールにつれて動きません。
- X Window システムコアプロトコルを使う”xterm” のようなアプリケーション上にマウスカーソールがある際には、X Window アーキテクチャーの分離不十分性がさらされ目玉はマウスカーソールにつれて動きます。

2021 年 4 月の時点で、GNOME や LibreOffice (LO) アプリケーション等の多くの人気ある GUI アプリケーションが Wayland 表示サーバープロトコルに移行しました。`xterm` や `gitk` や `chromium` や `firefox` や `gimp` や `dia` や KDE アプリケーションが、未だに X Window システムコアプロトコルを使っていると見受けます。

注意
Wayland 上の xwayland とネイティブの X Window システムの両方とも、古い X サーバー設定ファイル”`/etc/X11/xorg.conf`” はシステム上に存在するべきではありません。画像や入力デバイスは DRM や KMS や udev によるカーネルによって設定されます。ネイティブの X Window サーバーはこれらを使うように書き換えられました。Linux カーネル文書の”modedb デフォルトビデオモードサポート” を参照下さい。

7.3 GUI インフラストラクチャー

Wayland 環境上の GNOME のための特記すべき GUI インフラパッケージは以下です。

パッケージ	ポプコン	パッケージサイズ	説明
`mutter`	V:1, I:21	218	GNOME の mutter ウィンドウマネージャ **[自動]**
`xwayland`	V:256, I:332	2653	wayland 上で実行される X サーバー **[自動]**
`gnome-remote-desktop`	V:154, I:239	2743	GNOME のための PipeWire を使うリモートデスクトップデーモン **[自動]**
`gnome-tweaks`	V:17, I:230	1145	GNOME のための高度な構成設定
`gnome-shell-extension-prefs`	V:7, I:109	33	GNOME シェル拡張を有効化・無効化するツール

Table 7.2: データーインフラパッケージのリスト

ここで、”**[自動]**” は `task-gnome-desktop` がインストールされた際に自動的にインストールされるパッケージの意味です。

ティップ
`gnome-tweaks` は不可欠の設定ユーティリティーです。例えば:

- ”General” から音量ボリュームの”Over-Amplification” を強制できます。
- ”Keyboard & Mouse” -> ”Keyboard” -> ”Additional Layout Option” から”Caps” が”Esc” になるよう強制できます。

ティップ
GNOME デスクトップ環境の詳細機能は、`Super` キー打鍵後、”`settings`” か”`tweaks`” か”`extensions`” とタイプして起動されるユーティリティーで設定できます。

7.4 GUI アプリケーション

有用な GUI アプリケーションの多くは Debian で利用できるようになりました。対応する機能が GNOME デスクトップ環境下では利用できないため、`scribus` (KDE) のようなソフトウェアを GNOME デスクトップ環境にインストールするのはまったく問題ありません。しかしながら、機能が重複するパッケージをインストールしすぎるとあなたのシステムが散らかってしまいます。

著者の目に止まった GUI アプリケーションのリストを記します。

7.5 ユーザーディレクトリー

デスクトップ環境が使う、”`~/Desktop`”、”`~/Documents`”、...、のようなユーザーディレクトリーのデフォールトの名前はシステムのインストールが使ったロケールに依存します。これらは英語環境の名前に以下でリセットできます:

```
$ LANGUAGE=C xdg-user-dirs-update --force
```

そしてすべてのデーターを新ディレクトリーに手動で移動させます。xdg-user-dirs-update(1) を参照下さい。

それらは”`~/.config/user-dirs.dirs`” を編集することで如何なる名前にも設定できます。user-dirs.dirs(5) を参照下さい。

7.6 フォント

多くの有用なスケーラブルフォントが Debian 上のユーザーに使えるようになっています。ユーザーの関心事は如何に重複を回避するかとか、如何にインストール済みフォントを無効化するかです。こうしないとあなたの GUI アプリケーションのメニューが無用なフォントの選択肢で一杯になります。

Debian システムは FreeType 2.0 ライブラリーを使って多くのスケーラブルフォントフォーマットを画面や印刷用にラスター化します:

- Type 1 (PostScript) フォント、3 次ベジェ曲線を使用 (ほぼ廃れた形式)
- TrueType フォント、2 次ベジェ曲線を使用 (よい選択肢)
- OpenType フォント、3 次ベジェ曲線を使用 (最良の選択肢)

7.6.1 基本的なフォント

文字の大きさやグリフ提供範囲に関する明確な理解に基づき、適切なスケーラブルフォントをユーザーが選択する助けとなることを願って、以下のテーブルを編纂しました。ほとんどのフォントは全てのラテン文字とギリシャ文字とキリル文字をカバーします。最終的にアクティベートされるフォントの選択はあなたの美学によっても影響されます。これらの文字は画面表示にも紙への印刷にも使えます。

ここで:

- ”MCM” は”metric compatible with fonts provided by Microsoft” の意味です。
- ”MCMATC” は”metric compatible with fonts provided by Microsoft: Arial, Times New Roman, Courier New” の意味です。
- ”MCAHTC” は”metric compatible with fonts provided by Adobe: Helvetica, Times, Courier” の意味です。
- フォントタイプ列の数字は同一ポイントサイズのフォントに関する、”M” の概算相対幅を表します。

パッケージ	ポプコン	パッケージサイズ	タイプ	説明
evolution	V:27, I:227	521	GNOME	個人情報管理 (グループウエアと電子メール)
thunderbird	V:39, I:100	304278	GTK	E メールクライアント (Mozilla Thunderbird)
kontact	V:1.0, I:9.7	2416	KDE	個人情報管理 (グループウエアと電子メール)
libreoffice-writer	V:92, I:417	34243	LO	ワードプロセッサ
abiword	V:0.8, I:4.5	3596	GNOME	ワードプロセッサ
calligrawords	V:0.3, I:2.6	7144	KDE	ワードプロセッサ
scribus	V:1, I:12	32942	KDE	PDF ファイルを編集するためのデスクトップパブリッシング
glabels	V:0.4, I:2.4	1295	GNOME	ラベルエディター
libreoffice-calc	V:89, I:414	29622	LO	スプレッドシート
gnumeric	V:2.8, I:9.7	10053	GNOME	スプレッドシート
calligrasheets	V:0.2, I:1.6	14014	KDE	スプレッドシート
libreoffice-impress	V:78, I:412	2501	LO	プレゼンテーション
calligrastage	V:0.1, I:1.6	6237	KDE	プレゼンテーション
libreoffice-base	V:18, I:65	5136	LO	データーベース管理
kexi	V:0.04, I:0.73	7565	KDE	データーベース管理
libreoffice-draw	V:78, I:412	11424	LO	ベクトル画像エディター (ドロー)
inkscape	V:12, I:73	113357	GNOME	ベクトル画像エディター (ドロー)
karbon	V:0.2, I:2.0	4029	KDE	ベクトル画像エディター (ドロー)
dia	V:1, I:16	3846	GTK	フローチャートやダイアグラムエディター
gimp	V:40, I:206	33386	GTK	ビットマップ画像エディター (ペイント)
shotwell	V:15, I:244	6334	GTK	デジタル写真オーガナイザー
digikam	V:1.5, I:8.2	333	KDE	デジタル写真オーガナイザー
darktable	V:3, I:10	36092	GTK	写真家のためのライトボックスと暗室
gnome-video-trimmer	V:0.16, I:0.97	1665	GNOME	ビデオファイルのロスレス切り取り
kdenlive	V:3, I:21	22233	KDE	ノンリニアービデオエディター
shotcut	V:1.2, I:7.5	8488	Qt アプリ	ノンリニアービデオエディター
openshot-qt	V:0.8, I:8.3	196007	Qt アプリ	ノンリニアービデオエディター
flowblade	V:0.2, I:2.1	42213	GTK	ノンリニアービデオエディター
planner	V:0.2, I:4.3	1400	GNOME	プロジェクト管理
calligraplan	V:0.1, I:2.9	23545	KDE	プロジェクト管理
gnucash	V:2.0, I:6.4	29447	GNOME	個人会計
homebank	V:0.3, I:1.5	3277	GTK	個人会計
kmymoney	V:0.4, I:1.8	19313	KDE	個人会計
frescobaldi	V:0.3, I:1.4	14150	Qt アプリ	LilyPond 楽譜テキストエディター
librecad	V:1, I:13	12361	Qt アプリ	コンピューター支援設計 (CAD) システム (2D)
freecad	V:1, I:20	112	Qt アプリ	コンピューター支援設計 (CAD) システム (3D)
kicad	V:3, I:14	226336	GTK	電気回路図と PCB デザインソフトウェアー
xsane	V:9, I:127	1512	GTK	スキャナーのフロントエンド
libreoffice-math	V:73, I:415	1980	LO	数式エディター
calibre	V:7, I:24	76289	KDE	e-book コンバーターとライブラリーの管理
fbreader	V:0.7, I:5.2	4204	GTK	e-book リーダー
evince	V:76, I:280	953	GNOME	文書 (pdf) ビューワー
okular	V:43, I:128	4544	KDE	文書 (pdf) ビューワー

パッケージ	ポプコン	サイズ	**sans**	**serif**	**mono**	フォントについての注釈
fonts-cantarell	V:173, I:284	224	59	-	-	Cantarell (GNOME 3, 画面表示)
fonts-noto	I:146	30	61	63	40	Noto フォント (Google, multi-lingual with CJK)
fonts-dejavu	I:381	34	58	68	40	DejaVu (GNOME 2, MCM:Verdana, 拡張 Bitstream Vera)
fonts-liberation2	V:36, I:143	15	56	60	40	LibreOffice 用の Liberation フォント (Red Hat, MCMATC)
fonts-croscore	V:19, I:35	5260	56	60	40	Chrome OS: Arimo, Tinos と Cousine (Google, MCMATC)
fonts-crosextra-carlito	V:21, I:88	2696	57	-	-	Chrome OS: Carlito (Google, MCM:Calibri)
fonts-crosextra-caladea	V:13, I:81	347	-	55	-	Chrome OS: Caladea (Google, MCM:Cambria) (Latin 文字のみ)
fonts-freefont-ttf	V:77, I:200	14461	57	59	40	GNU FreeFont (拡張 URW Nimbus)
fonts-quicksand	V:210, I:450	392	56	-	-	Debian task-desktop, Quicksand (画面表示, Latin 文字のみ)
fonts-hack	V:33, I:135	2509	-	-	40 P	ソースコードのためにデザインされたタイプフェース Hack (Facebook)
fonts-sil-gentiumplus	V:1, I:27	14345	-	54	-	Gentium SIL
fonts-sil-charis	V:1, I:27	6704	-	59	-	Charis SIL
fonts-urw-base35	V:187, I:533	15558	56	60	40	URW Nimbus (Nimbus Sans, Roman No. 9 L, Mono L, MCAHTC)
fonts-ubuntu	V:2.1, I:4.4	4312	58	-	33 P	Ubuntu フォント (画面表示)
fonts-terminus	V:0.3, I:3.8	451	-	-	33	Cool retro terminal fonts
ttf-mscorefonts-installer	V:1, I:40	85	56?	60	40	Microsoft の非フリーフォントのダウンローダー (以下を参照下さい)

Table 7.4: 特記すべきTrueType やOpenType フォントのリスト

- モノフォントタイプ列の”P” は、”0”/”O” と”1”/”l”/”I” がはっきり区別しやすいというプログラミング用としての使い勝手の良さを表します。
- `ttf-mscorefonts-installer` パッケージはマイクロソフトの”Core fonts for the Web” をダウンロードして、Arial と Times New Roman と Courier New と Verdana と... をインストールします。こうしてインストールされるフォントデーターは non-free のデーターです。

多くのフリーの Latin フォントは、URW Nimbus ファミリーとか Bitstream Vera に、それらへの系統をたどれます。

ティップ
あなたのロケールが、上記のフォントでうまくカバーできないフォントを必要とする場合、aptitude を使って”Tasks” -> ”Localization” の下にリストされたタスクパッケージをチェックしましょう。ローカリゼーションタスク中の”Depends:” か”Recommends:” としてリストされたフォントパッケージが主要候補です。

7.6.2 フォントのラスタ化

Debian は FreeType をフォントをラスタ化に使用します。そのフォント選択インフラは Fontconfig フォント設定ライブラリーで提供されます。

パッケージ	ポプコン	サイズ	説明
`libfreetype6`	V:582, I:996	1098	FreeType フォントラスター化ライブラリー
`libfontconfig1`	V:580, I:801	344	Fontconfig、フォント設定ライブラリー
`fontconfig`	V:477, I:688	415	`fc-*`: Fontconfig の CLI コマンド
`font-manager`	V:2.2, I:6.3	1139	Font Manager: Fontconfig の GUI コマンド
`nautilus-font-manager`	V:0.09, I:0.28	43	Font Manager の Nautilus 拡張

Table 7.5: 有用フォント環境と関連パッケージのリスト

ティップ
`fonts-noto*` のようないくつかのフォントパッケージは多すぎるフォントインストールします。普通の使用状況下では、いくつかのフォントパッケージをインストールはしながら無効化したいかもしれません。いくつかの Unicode コードポイントでは漢統一 (Han unification) のため複数のグリフが期待されていて、Fontconfig ライブラリーが設定がまだだと希望しないグリフが表示されかもしれません。最も気になるケースは CJK 国間の”U+3001 IDEOGRAPHIC COMMA” と”U+3002 IDEOGRAPHIC FULL STOP” です。フォントマネージャー GUI (font-manager) を使いフォントの可用性を設定することによりこのような問題状況は簡単に回避できます。

フォント設定状態は以下のようにしても確認できます。

- fontconfig のフォントデフォルトに関しては”`fc-match(1)`”
- fontconfig で利用可能なフォントに関しては”`fc-list(1)`”

テキストエディターからフォント設定状態を設定できますが、これは簡単ではありません。fonts.conf(5) を参照下さい。

7.7 サンドボックス

Linux 上の主に GUI アプリケーションの多くはバイナリー形式で非 Debian ソースから利用可能です。

- AppImage -- どこででも実行できる Linux アプリ
- FLATHUB -- Linux 用アプリ、ここで使える
- snapcraft -- Linux 用アプリストアー



警告
これらのサイトからのバイナリーはプロプライエタリーな non-free ソフトウェアーパッケージが含まれているかもしれません。

各アプリのアップストリームデベロッパーが当該アプリに使っている本来のライブラリーの組み合わせを Debian が提供するライブラリーと独立にアプリに提供することを可能にするので、Debian を使うフリーソフトウェアー愛好家にとってもこれらのバイナリー形式ディストリビューションには一定の存在意義があります。

外部バイナリーを実行する本質的リスクは、Linux の現代的なセキュリティー機能を利用するサンドボックス環境を使うことで低減できます (項 4.7.5 を参照下さい)。

- AppImage やいくつかのアップストリームサイトからのバイナリーに関しては、手動設定の下で firejail を実行します。
- FLATHUB からのバイナリーに関しては、 Flatpak の中で実行します。(手動設定不要です。)
- snapcraf からのバイナリーに関しては、 Snap の中で実行します。(手動設定不要です。デーモンプログラムとコンパチブルです。)

`xdg-desktop-portal` パッケージは共通のデスクトップ機能への標準化された API を提供します。 xdg-desktop-portal (flatpak) や xdg-desktop-portal (snap) を参照下さい。

このサンドボックス環境技術はアプリが制御されたリソースアクセス下で実行されるスマートフォンの OS 上のアプリと非常に似ています。

ウエッブブラウザーのようないくつかの大きな GUI アプリも、より安全にするために内部的にサンドボックス環境技術を使います。

7.8 リモートデスクトップ

7.9 X サーバ接続

ローカルホスト上の `xwayland` も含めた X サーバーにリモートホスト上のアプリが接続する方法がいくつかあります。

7.9.1 X サーバローカル接続

ローカルの UNIX ドメインソケット経由でローカル接続することで、X コアプロトコルを使うローカルのアプリはローカルの X サーバーにアクセスできます。これはアクセスクッキーを保持する権限ファイルによって許諾されます。権限ファイルの場所は”`$XAUTHORITY`” 環境変数により特定され、X ディスプレーは”`$DISPLAY`” 環境変数により特定されます。普通これらは自動設定されているので、例えば”`gitk`” の場合以下のように特段のアクションは不要です。

パッケージ	ポプコン	サイズ	説明
flatpak	V:108, I:116	9297	デスクトップアプリ用の Flatpak アプリケーションデプロイメントフレームワーク
gnome-software-plugin-flatpak	V:23, I:30	291	GNOME ソフトウェアーへの Flatpak サポート
snapd	V:56, I:59	62738	snap パッケージを有効化するデーモンやツール
gnome-software-plugin-snap	V:1.4, I:2.0	144	GNOME ソフトウェアーへの Snap サポート
xdg-desktop-portal	V:376, I:438	2291	Flatpak や Snap デスクトップインテグレーションポータル
xdg-desktop-portal-gtk	V:340, I:436	715	gtk 用の xdg-desktop-portal バックエンド (GNOME)
xdg-desktop-portal-kde	V:81, I:111	3333	Qt 用の xdg-desktop-portal バックエンド (KDE)
xdg-desktop-portal-wlr	V:2.4, I:6.7	167	wltools 用の xdg-desktop-portal バックエンド (Wayland)
firejail	V:1.1, I:4.2	1725	AppImage に用いる SUID のセキュリティーサンドボックスプログラム firejail

Table 7.6: 特記すべきサンドボックス環境や関連のパッケージのリスト

パッケージ	ポプコン	サイズ	プロトコル	説明
gnome-remote-desktop	V:154, I:239	2743	RDP	GNOME Remote Desktop サーバー
xrdp	V:29, I:31	4757	RDP	xrdp, リモートデスクトッププロトコル (RDP) サーバー
x11vnc	V:10, I:48	1863	RFB (VNC)	x11vnc, リモートフレームバッファープロトコル (VNC) サーバー
tigervnc-standalone-server	V:6, I:15	3176	RFB (VNC)	TigerVNC, リモートフレームバッファープロトコル (VNC) サーバー
gnome-connections	V:7, I:154	1726	RDP, RFB (VNC)	GNOME リモートデスクトップクライアント
remmina	V:16, I:59	1046	RDP, RFB (VNC), SPICE, SSH, ...	Remmina: GTK リモートデスクトップクライアント
krdc	V:2, I:15	4144	RDP, RFB (VNC)	KRDC: KDE リモートデスクトップクライアント
virt-viewer	V:5, I:36	1278	RFB (VNC), SPICE	仮想マシンマネージャーのゲスト OS の GUI 表示クライアント

Table 7.7: 特記すべきリモートアクセスサーバーのリスト

パッケージ	ポプコン	サイズ	コマンド	説明
openssh-server	V:781, I:824	3615	X11-forwarding オプション付きの sshd	SSH サーバ (セキュア)
openssh-client	V:709, I:997	3611	ssh -X	SSH クライアント (セキュア)
xauth	V:197, I:975	81	xauth	X 認証ファイルユーティリティ
x11-xserver-utils	V:376, I:532	559	xhost	X のサーバアクセス制御

Table 7.8: X サーバーへの接続方法のリスト

```
username $ gitk
```

注意
xwayland の場合、XAUTHORITY は"/run/user/1000/.mutter-Xwaylandauth.YVSU30" のような値を取ります。

7.9.2 X サーバリモート接続

X コアプロトコルを使うリモートのアプリからローカルの X サーバーへのアクセスは X11 フォワーディング機能を使ってサポートされます。

- ローカルホスト上で gnome-terminal を開きます。
- ssh(1) を -X オプションとともに実行して以下のようにリモートサイトとの接続を確立する。

```
localname @ localhost $ ssh -q -X loginname@remotehost.domain
Password:
```

- リモートホスト上の"gitk" 等の X アプリケーションコマンドを以下のように実行します。

```
loginname @ remotehost $ gitk
```

ここに書かれた手法はリモート X クライアントがあたかもローカルの UNIX ドメインソケット経由でローカル接続されているかのようにして、リモート X クライアントからの出力を表示できるようにします。

SSH/SSHD に関しては項 6.3 を参照下さい。



警告
X サーバーへの TCP/IP 接続はセキュリティー上の理由で Debian システム上では無効化されています。回避可能であれば、"xhost +" と単純に設定しこれを有効化してはいけないし、また XDMCP 接続を有効化してこれを有効化してもいけません。

7.9.3 X サーバ chroot 接続

X コアプロトコルを使う同一ではあるが認証ファイルにアクセスできない chroot のような環境で実行されるアプリケーションから X サーバーへのアクセスは、例えば"gitk" の場合は以下のようにして User-based アクセスを使うと xhost を使い安全に認証できます。

```
username $ xhost + si:localuser:root ; sudo chroot /path/to
# cd /src
# gitk
# exit
username $ xhost -
```

7.10 クリップボード

テキストのクリップボードへのクリップに関しては、項 1.4.4 を参照下さい。

グラフィクスのクリップボードへのクリップに関しては、項 11.6 を参照下さい。

文字クリップボード (PRIMARY と CLIPBOARD) の操作は、いくつかの CLI コマンドを使うと可能です。

パッケージ	ポプコン	パッケージサイズ	ターゲット	説明
xsel	V:8, I:42	55	X	X 選択のコマンドラインインターフェース (クリップボード)
xclip	V:19, I:78	62	X	X 選択のコマンドラインインターフェース (クリップボード)
wl-clipboard	V:11, I:28	174	Wayland	wl-copy wl-paste: Wayland クリップボードへのコマンドラインインターフェース
gpm	V:8.0, I:8.7	524	Linux コンソール	Linux コンソールのマウスイベントを捉えるデーモン

Table 7.9: 文字クリップボードの操作関連プログラムのリスト

Chapter 8

I18N と L10N

アプリケーションソフトの多言語化 (M17N) とかネイティブ言語サポートは 2 段階で行います。

- 国際化 (I18N): ソフトが複数のロケール (地域) を扱えるようにします。
- 地域化 (L10N): 特定のロケール (地域) を扱えるようにします。

ティップ
M17N、I18N、L10N に対応する英語の multilingualization、internationalization、localization の中の”m” と”n”、”i” と”n”、”l” と”n” の間には 17、18、10 の文字があります。詳細は、国際化と地域化を参照下さい。

8.1 ロケール

国際化をサポートするプログラムの挙動は、ローカル化をサポートする環境変数”$LANG” を使って設定されます。libc ライブラリーによるロカール依存機能の実際のサポートには、locales か locales-all パッケージをインストールする必要があります。locales パッケージは適切に初期化する必要があります。

locales と locales-all パッケージのいずれもインストールされない場合、ロカール機能のサポートは失われ、システムは US 英語のメッセージ使い、データーを **ASCII** として取り扱います。この挙動は”$LANG” が”LANG=” か”LANG=C” か”LANG=POSIX” と設定されたのと同様です。

GNOME や KDE 等の現代的なソフトは多言語化されています。UTF-8 データーを扱えるようにすることで国際化され、gettext(1) インフラで翻訳されたメッセージを提供することで地域化されています。翻訳されたメッセージは別の地域化パッケージとして供給されているかもしれません。

現行の Debian デスクトップ GUI システムは普通 GUI 環境下のロカールを”LANG=xx_YY.UTF-8” と設定します。ここで、”xx” は ISO 639 言語コードで”YY” は ISO 3166 国コードです。これらの値はデスクトップ設定 GUI ダイアログで設定されプログラムの挙動を変えます。項 1.5.2 を参照下さい

8.1.1 UTF-8 ロケールを使う根拠

テキストデータの最も単純な表現は **ASCII** で、英語には十分で 127 未満の文字 (7 ビットで表現可能) を使います。

プレーンな英語のテキストですら非 ASCII 文字を含んでいるかもしれません。例えば微妙に曲がった左右のクォーテーションマークは ASCII 内では利用できません。

```
b'' “b''double quoted textb''” b'' is not "double quoted ASCII"
b'' ‘b''single quoted textb''’ b'' is not 'single quoted ASCII'
```

より多くの文字をサポートするために、多くの言語をサポートする多数の文字集合とエンコーディング体系が使用されてきました (表 11.2 を参照下さい)。

ユニコード文字セットは実質的に人類が知り得る全ての文字を 21 ビットのコードポイント範囲 (16 進表記で 0 から 10FFFF まで) で表記できます。

テキストエンコーディングシステム UTF-8 は、ASCII データー処理システムとほぼ互換な賢明な 8 ビットデーターストリームにユニコードコードポイントを当てはめます。これが **UTF-8** を現代的な好ましい選択肢にします。**UTF** はユニコード変換フォーマット (Unicode Transformation Format) の意味です。ASCII プレーンテキストデーターが UTF-8 データーに変換されるとき、最初の ASCII データーと全く同じ内容とサイズです。UTF-8 ロケールを採用することで失いものはありません。

UTF-8 ロカールの下の互換アプリケーションプログラムを使うと、必要なフォントとインプットメソッドが導入され有効化されていれば、いかなる外国語のテキストデーターの表示や編集ができます。例えば、”LANG=fr_FR.UTF-8” ロケールの下で、gedit(1) (GNOME デスクトップ用のテキストエディター) は、メニューをフランス語で表示しながら中国語の文字の表示や編集ができます。

ティップ
新標準の”en_US.UTF-8” ロケールと旧標準の”C”/”POSIX” ロケールは標準アメリカ英語のメッセージを使いますが、ソート順などでわずかに違います。古い”C” ロケールの挙動を保持しながら、ASCII 文字を扱うだけでなく、UTF-8 でエンコードされた全ての文字を優雅に扱いたい場合は、非標準の”C.UTF-8” ロケールを Debian で使います。

注意
一部のプログラムは I18N をサポートした後でより多くのメモリーを消費するようになります。それらのプログラムは、実行速度最適化のために内部的に UTF-32 (UCS4) で Unicode のサポートをコードされていて、選ばれたロケールに無関係にそれぞれの ASCII 文字データー毎に 4 バイトを消費するからです。ここでも、UTF-8 ロケールを使ったからといって何も失うわけではありません。

8.1.2 ロケールの再設定

システムが特定のロケールにアクセスできるように、ロケールデータをロケールデータベースからコンパイルする必要が有ります。

locales パッケージには、事前にコンパイルしたロケールデーターは同梱されていません。以下のようにして設定する必要があります:

```
# dpkg-reconfigure locales
```

このプロセスは 2 段階あります。

1. バイナリー形式にコンパイルしたい全ての必要なロケールデーターを選択します。(少なくとも 1 つの UTF-8 ロケールが含めて下さい)
2. PAM (項 4.5 を参照下さい) によって使われるように”/etc/default/locale” を作成しシステム全体のデフォールトのロケール値を設定。

”/etc/default/locale” 中に設定されたシステム全体のデフォルトロケール値は GUI アプリケーション用の GUI 設定によりオーバーライドされるかもしれません。

注意
実際に使われる符号化方式は”/usr/share/i18n/SUPPORTED” を確認することで識別できます。だから、”en_US” は”ISO-8859-1” 符号化方式を使います。

`locales-all` パッケージには、全てのロケールデーター用に事前にコンパイルしたロケールデーターが同梱されています。”`/etc/default/locale`” を作成しないので、`locales` パッケージもまだインストール必要があるかもしれません。

ティップ
いくつかの Debian 派生のディストリビューションの `locales` パッケージには、事前にコンパイルしたロケールデーターが同梱されています。そのようなシステム環境を Debian 上で再現するには `locales` と `locales-all` パッケージの両方をインストールする必要があります。

8.1.3 ファイル名の符号化方式

クロスプラットフォームのデーター交換 (項 10.1.7 を参照下さい) のために、特定の符号化方式 (エンコーディング) でファイルシステムをマウントする必要があるかもしれません。例えば、vfat ファイルシステムに関して mount(8) はオプション無しの場合 CP437 とみなします。ファイル名に UTF-8 とか CP932 を使うためには明示的にマウントオプションを提供する必要があります。

注意
GNOME のような現代的なデスクトップ環境の下では、デスクトップアイコンを右クリックし”Drive” タブをクリックし”Setting” を開くようにクリックし”Mount options:” に”utf8” を入力すれば、ホットプラグできる USB フラッシュドライブを自動マウントする時のマウントオプションを設定できます。次回この USB フラッシュドライブをマウントする際には UTF-8 でのマウントが有効です。

注意
もしシステムをアップグレードしたり旧式非 UTF-8 システムからディスクを移動したりする場合には、非 ASCII 文字のファイル名は ISO-8859-1 とか eucJP 等の今は非推奨の歴史的符号化方式で符号化をしているかもしれません。テキスト変換ツールの助力を得て、ファイル名を UTF-8 に変換します。項 11.1 を参照下さい。

Samba は新規クライアント (Windows NT、200x、XP) には Unicode を使いますが、旧式クライアント (DOS、Windows 9x/Me) には CP850 をデフォールトで使います。この旧式クライアントへのデフォールトは”`/etc/samba/smb.conf`” ファイル中の”`dos charset`” を使って例えば日本語なら CP932 等と変更できます。

8.1.4 地域化されたメッセージと翻訳された文書

Debian システム中で表示されるエラーメッセージや標準のプログラムの出力やメニューやマニュアルページ等のテキストメッセージや文書の多くに翻訳があります。ほとんどの翻訳行為のバックエンドツールとして GNU gettext(1) コマンドツールチェインが使われています。

”Tasks” → ”Localization” の下の aptitude(8) リストは地域化されたメッセージをアプリケーションに追加したり翻訳された文書を提供する有用なバイナリーパッケージの徹底的なリストを提供します。

例えば、`manpages-`*`LANG`* パッケージをインストールするとマンページで地域化したメッセージに使えるようになります。*programname* に関するイタリア語のマンページを”`/usr/share/man/it/`” から読むには、次を実行します。

```
LANG=it_IT.UTF-8 man programname
```

GNU gettext は $LANGUAGE 環境変数を使って翻訳言語の優先順位をつけるようにできます。例えば:

```
$ export LANGUAGE="pt:pt_BR:es:it:fr"
```

詳しくは、`info gettext` を参照して”The LANGUAGE variable” セクションを読んで下さい。

8.1.5 ロケールの効果

sort(1) や ls(1) での並べ替え順はロケールの影響を受けます。LANG=en_US.UTF-8 をエクスポートすると辞書順 A->a->B->b...->Z->z で並べ替えられ、一方 LANG=C.UTF-8 をエクスポートすると ASCII バイナリー順 A->B->...->Z->a->b... で並べ替えられます。

ls(1) の日付形式はロケールに影響されます。literal>" の日付形式は違います (項 9.3.4 を参照下さい)。

date(1) の日付形式はロケールに影響されます。例えば以下です:

```
 $ unset LC_ALL
 $ LANG=en_US.UTF-8 date
Thu Dec 24 08:30:00 PM JST 2023
 $ LANG=en_GB.UTF-8 date
Thu 24 Dec 20:30:10 JST 2023
 $ LANG=es_ES.UTF-8 date
jue 24 dic 2023 20:30:20 JST
 $ LC_TIME=en_DK.UTF-8 date
2023-12-24T20:30:30 JST
```

数字の区切り方はロケール毎に異なります。例えば、英語のロケールでは一千点一は"1,000.1" と表示されますが、ドイツ語のロケールでは"1.000,1" と表示されます。スプレッドシートプログラムでこの違いを目にするしれません。

"$LANG" 環境変数の各詳細機能は、"$LC_*" 変数の設定でオーバーライドされます。これらの環境変数は更に"$LC_ALL" 変数の設定でオーバーライドされます。詳細は locale(7) マンページを参照下さい。複雑な設定をするよほどの理由がない限り、これらは使わずただ"$LANG" 変数だけを UTF-8 ロカールの 1 つに設定して使いましょう。

8.2 キーボード入力

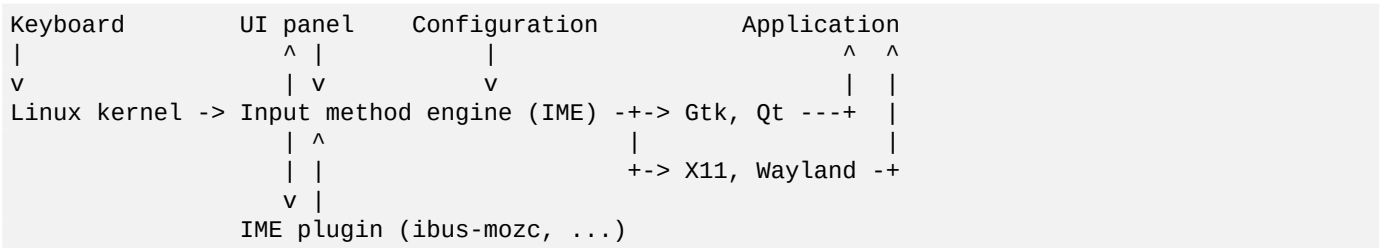
キーボードシステムはシステムの異なるレーヤーで設定できます。

- Linux カーネル: keyboard(5)
- X サーバー: setxkbmap(1), xkeyboard-config(5), 環境変数 XMODIFIERS
- GUI デスクトップ環境: インプットメソッド枠組: ibus, fcitx5
- アプリケーション: インプット元を設定する環境変数: GTK_IM_MODULE, QT_IM_MODULE, QT_IM_MODULES, ...

インプットメソッド枠組み (IM) は以下で構成されます:

- インプットメソッドエンジン (IME): 実際のインプットメソッド
- 設定: IBus や IME プラグインのような他のサービスの設定を取り扱う
- パネル: 言語バーや選択候補テーブルのようなユーザーインターフェース

アプリケーションへの多言語入力は概略以下のように処理されます。



8.2.1 Linux コンソールと X Window 用のキーボードインプット

Debian システムは keyboard-configuration パッケージを使い多くの国際キーボード配列として機能するように設定できます。

```
# dpkg-reconfigure keyboard-configuration
```

Linux コンソールや X Window システムでは、これは”/etc/default/keyboard” 内の設定パラメーターを更新します。多くの欧州言語で用いられるアクセント付き文字を含めた多くの非 ASCII 文字はデッドキーや AltGr キーやコンポーズキーを用い提供されます。

注意
もし ibus がアクティブな場合、たとえ古典的 X 上のデスクトップ環境下でも、あなたの setxkbmap による古典的 X キーボード設定は ibus によってオーバーライドされるかもしれません。im-config を使ってインプットメソッドを”None” と設定すると、インストールされた ibus は無効化できます。詳細は、キーボードに関する Debian Wiki を参照下さい。

8.2.2 Wayland 向けのキーボード入力

X Window プロトコルと違い、Wayland のコアのプロトコルはアクセント付き文字すらサポートしません。GNOME 用の Mutter とか KDE 用の KWin 等の人気のある Wayland コンポジターはテキスト入力用の text-input-unstable-v3 のような拡張プロトコルを実装しています (”current Wayland protocols and their support status” を参照下さい)。

Wayland では、text-input-unstable-v3 プロトコルがインプットメソッドと相性よく機能します (”Wayland input method project post-mortem” を参照下さい)。.

- この text-input-unstable-v3 をサポートする GTK とか Qt のようなライブラリーを用いてほとんどの GUI アプリケーションはビルドされます。
- IBus とか Fcitx (version 5) のような、人気のインプットメソッドエンジン (IME) はこの text-input-unstable-v3 と相性よく機能します
- IME は多くの言語のためのテキスト入力をプラグインを用いてサポートします。
- 最近の IME は、Wayland で欧州言語用のアクセント付き文字のテキスト入力をサポートするために、過去 X Window が提供していた setxkbmap のような”X Keyboard Extension (XKB) (X キーボード拡張機能)” の機能を中に組み込んでいます。

8.2.3 IBus を使うインプットメソッドのサポート

GNOME では”ibus” がそのパッケージ依存関係から自動導入されるデフォルトの IME です。

- 殆どの多言語キーボード入力機能は”GNOME Settings” か”GNOME Tweaks” から設定できます。
- 一部の多言語キーボード入力機能は ibus-setup(1) コマンドから設定する必要があるかもしれません。
- 絵文字キーボード入力は”SUPER-.” をタイプ (Windows キーとピリオドキーの同時タイプ) に続いて、各絵文字のキーワードと SPACE-キーをタイプしてできます。

IBus とそのプラグインパッケージのリストは以下の通りです。

8.2.4 Fcitx を使うインプットメソッドのサポート

Fcitx (version 5) インプトメソッドフレームワークは中国人ユーザーに人気があり”ibus” と互換性があります。

”fcitx5” とそのプラグインパッケージのリストは以下の通りです。

パッケージ	ポプコン	サイズ	サポートされたロケール
ibus	V:211, I:252	1884	dbus を用いるインプットメソッドのフレームワーク
ibus-mozc	V:2.0, I:3.7	872	日本語
ibus-anthy	V:0.45, I:0.98	8740	日本語
ibus-skk	V:0.04, I:0.15	242	日本語
ibus-kkc	V:0.04, I:0.18	206	日本語
ibus-libpinyin	V:1.0, I:4.7	125	中国語 (zh_CN 用)
ibus-chewing	V:0.11, I:0.73	298	中国語 (zh_TW 用)
ibus-libzhuyin	I:0.08	41021	中国語 (zh_TW 用)
ibus-rime	V:0.31, I:0.50	76	中国語 (zh_CN/zh_TW 用)
ibus-cangjie	V:0.01, I:0.09	235	中国語 (zh_HK 用)
ibus-hangul	V:0.3, I:1.9	261	韓国語
ibus-libthai	V:0.00, I:0.05	84	タイ語
ibus-table-thai	I:0.05	59	タイ語
ibus-unikey	V:0.21, I:0.42	286	ベトナム語
keyman	V:0.01, I:0.09	507	多言語: 2000 語以上のための Keyman プラグイン
ibus-table	V:0.06, I:0.85	2482	IBus 用のテーブルプラグイン
ibus-m17n	V:0.2, I:1.7	373	多言語: インド系言語、アラビア語、他

Table 8.1: IBus とプラグイン/パッケージのリスト

パッケージ	ポプコン	サイズ	サポートされたロケール
fcitx5	V:8, I:12	712	”ibus” 互換のインプとメソッドフレームワーク
fcitx5-mozc	V:1.1, I:1.8	1578	日本語
fcitx5-anthy	V:0.06, I:0.24	853	日本語
fcitx5-skk	V:0.08, I:0.16	371	日本語
fcitx5-kkc	V:0.00, I:0.07	418	日本語
fcitx5-chinese-addons	I:8.7	17	中国語 (zh_* 用メタパッケージ)
fcitx5-pinyin	V:4.2, I:9.4	1000	中国語 (zh_CN 用)
fcitx5-chewing	V:0.29, I:0.92	228	中国語 (zh_TW 用)
fcitx5-zhuyin	I:0.07	108535	中国語 (zh_TW 用)
fcitx5-rime	V:0.57, I:0.99	393	中国語 (zh_CN/zh_TW 用)
fcitx5-table-cangjie-large	I:0.12	1292	中国語 (zh_HK 用)
fcitx5-hangul	V:0.13, I:0.28	239	韓国語
fcitx5-libthai	I:0.04	119	タイ語
fcitx5-table-thai	I:0.07	34	タイ語
fcitx5-unikey	V:0.13, I:0.26	624	ベトナム語
fcitx5-m17n	V:0.41, I:0.85	171	多言語: インド系言語、アラビア語、他
fcitx5-table	V:0.3, I:9.2	428	fcitx5 用のテーブルプラグイン
fcitx5-keyman	V:0.04, I:0.05	241	多言語: 2000 語以上のための Keyman プラグイン

Table 8.2: Fcitx5 とプラグインパッケージのリスト

8.2.5 日本語の例

日本語インプットメソッドを英語環境 (”en_US.UTF-8”) 下で起動するのが非常に便利です。IBus を使ってどう実現したかを以下に記します。

1. 日本語インプットツール/パッケージの ibus-mozc (または ibus-anthy) をインストールします。
2. • 一般: ibus-setup(1) を実行 → ”Input Method” を選択 → ”Add” をクリック → ”Japanese” → ”Mozc (または Anthy)” → ”Add” をクリック
   • GNOME: ”Settings” → ”Keyboard” → ”Input Sources” → ”Input Sources” 中の”+” をクリック → ”Japanese” → ”Mozc (または Anthy)” → ”Add” をクリック
3. インプットソースはいくつ選んでも構いません。
4. ユーザーアカウントに再ログインします。
5. GUI ツールバーアイコンを右クリックして各インプットソースを設定します。
6. インプットソース間を、SUPER-SPACE を用いて切り替えます。(SUPER は普通 Windows キーです。)

ティップ
シフト 2 が" (ダブルクォーテーションマーク) の刻印のある物理的な日本語キーボードでアルファベットのみのキーボード環境にアクセスするには、上記の手順で”Japanese” を選びます。シフト 2 が @ (アットマーク) の刻印のある物理的な US 英語キーボードを”Japanese mozc (もしくは anthy)” を使うと日本語が入力できます。

- Wayland だと:
  - im-config パッケージは何もし無いし、安全に削除可能です。
  - 環境変数の設定をする必要は、バックワード互換性のため等を除けばきっと必要ないでしょう。
  - 環境変数を設定する必要のある場合には、”~/.config/environment.d/50-input-method.conf” のようなファイルを作成して環境変数を設定します。
- X Window だと:
  - im-config パッケージをインストールします。
  - im-config(8) のための GUI メニューエントリーは”Input method” です。
  - あるいは、ユーザのシェルから”im-config” を実行します。
  - im-config(8) は実行されるのが root からかどうかによって違った挙動をします。
  - im-config(8) はユーザーからのアクション無しにシステム上で最も好ましいインプットメソッドを有効にします。

8.3 ディスプレー出力

Linux コンソールは限定された文字しか表示できません。(非 GUI コンソール上で非ヨーロッパ言語を表示するには jfbterm(1) のような特別なターミナルプログラムを使う必要があります。)

GUI 環境 (第 7 章) は、必要なフォントデーターがあれば UTF-8 中の全ての文字を表示できます。(オリジナルフォントデーターで使われた符号化方式は面倒を見られているのでユーザーからは見えません。)

8.3.1 ターミナルの設定

Debian システムは `console-setup` パッケージを使い多くの国際化コンソール設定をともに使えるように設定できます。

```
# dpkg-reconfigure console-setup
```

Linux コンソールや X Window システムでは、これは”`/etc/default/console-setup`” 内の設定パラメーターを更新し多くの欧州言語で用いられるアクセント付き文字を含めた多くの非 ASCII 文字を表示するようにします。

文字コンソールと ncurses(3) システム機能を設定するのはいくつかの要素があります。

- ”`/etc/terminfo/*/*`” ファイル (terminfo(5))
- ”`$TERM`” 環境変数 (term(7))
- setterm(1)、stty(1)、tic(1)、toe(1)

もし `xterm` 用の `terminfo` エントリーが非 Debian の `xterm` でうまく機能しない場合には、リモートから Debian システムにログインする時にターミナルタイプ、”`$TERM`”、を”`xterm`” から”`xterm-r6`” のような機能限定版に変更します。詳細は”`/usr/share/doc/ncurses-base/FAQ`” を参照下さい。”`dumb`” は”`$TERM`” の最低機能の共通項です。

8.3.2 東アジア不明瞭文字幅文字

東アジアのロケールでは、箱描画文字やギリシャ文字やキリル文字はあなたが望むよりも広い幅で表示されて、ターミナル出力が揃わなくなるかもしれません (Unicode 標準附属書 #11, 4.2 不明瞭文字参照)。

この問題は回避可能です:

- `gnome-terminal`: Preferences → Profiles → *Profile name* → Compatibility → Ambiguous-wide characters → Narrow
- `ncurses`: 環境変数を `export NCURSES_NO_UTF8_ACS=0` と設定します。

Chapter 9

システムに関するティップ

主にコンソールからシステムを設定や管理する基本的なティップを次に記します。

9.1 コンソールのティップ

コンソール活動を補助するユーティリティープログラムがいくつかあります。

パッケージ	ポプコン	サイズ	説明
mc	V:42, I:172	1590	項 1.3 を参照
bsdutils	V:445, I:1000	338	ターミナルセッションの記録を作成する script コマンド
screen	V:49, I:187	1007	VT100/ANSI ターミナルエミュレーションを使ってのターミナルマルチプレクサ
tmux	V:114, I:172	1653	代替のターミナルマルチプレクサ (代わりに”Control-B” を用いる)
ripgrep	V:17, I:49	5284	自動フィルタリング付きのソースコード中の高速再帰文字検索
fzf	V:10, I:37	5692	ファジーテキストファインダ
fzy	V:0.05, I:0.31	59	ファジーテキストファインダ
rlwrap	V:1, I:11	328	readline 機能のコマンドラインラッパー
ledit	V:0.4, I:7.1	401 401	readline 機能のコマンドラインラッパー
rlfe	V:0.03, I:0.49	45	readline 機能のコマンドラインラッパー

Table 9.1: コンソールの活動をサポートするプログラムのリスト

9.1.1 シェルの活動を綺麗に記録

単に script(1) を使ってシェル活動を記録すると (項 1.4.9 を参照下さい)、コントロール文字の入ったファイルが生成されます。このような事は以下のようにして col(1) を使うことで避けられます。

```
$ script
Script started, file is typescript
```

何なりとします…そして `script` から脱出するために `Ctrl-D` を押します。

```
$ col -bx < typescript > cleanedfile
$ vim cleanedfile
```

シェルの活動を記録する他の方法もあります:

- tee を使う (initramfs 中のブートプロセスで有用):

```
$ sh -i 2>&1 | tee typescript
```

- スクロールバック用バッファーを拡張した gnome-terminal を使います。
- screen(1) を”^A H” で使い (項 9.1.2 を参照下さい) コンソールの記録をします。
- vim を”:terminal” で使ってコンソールの記録します。”C-W N” として TERMINAL モードから NORMAL モードに出ます。”:w typescript” としてバッファーをファイルに書き出します。
- emacs を”M-x shell” か”M-x eshell” か”M-x term” で使ってコンソールの記録します。”C-x C-w” としてバッファーをファイルに書き出します。

9.1.2 screen プログラム

screen(1) は複数のプロセスを 1 つのターミナルウィンドウでうまく動作させるのみならず、接続が中断してもリモートシェルプロセスを生き延びさせる事もできます。screen(1) の使われ方の典型的シナリオは次です。

1. リモート機器にログインします。
2. 単一のコンソール上で screen を起動します。
3. ^A c (”Control-A” に続いて”c”) によって作られた screen のウィンドウ中で複数のプログラムを実行します。
4. ^A n (”Control-A” に続いて”n”) によって、複数の screen のウィンドウ間を切り替えます。
5. 突然ターミナルを離れる必要ができたけれども、接続を継続してあなたのアクティブな作業を失いたくありません。
6. いかなる方法ででも、screen のセッションをデタッチできます。
   - 暴力的にネットワーク接続を引き抜く
   - ^A d (”Control-A” に続いて”d”) とタイプしてリモート接続から手動でログアウト
   - ^A DD (”Control-A” に続いて”DD”) とタイプして screen をデタッチしてログアウト
7. 同じリモート機器に (たとえ異なるターミナルからでも) 再びログインします。
8. screen を”screen -r” として起動します。
9. screen は全アクティブなプログラムが実行されている過去の全 screen ウィンドウを魔法のようにリアタッチします。

ティップ
screen を使うと、切断してもプロセスをアクティブにしておけその後で再接続した時にリアタッチできるので、ダイヤルアップやパケット接続のような計量されたネットワーク接続での接続料金の節約ができます。

---

screen セッションではコマンドキーストローク以外の全てのキーボード入力は現在のウィンドウに送られます。全ての screen コマンドキーストロークは ^A (”Control-A”) と単一キー [プラス何らかのパラメーター] をタイプすることによって入力されます。次に覚えておくべき重要なコマンドキーストロークを記します。

詳細は screen(1) を参照下さい。

代替コマンドの機能については tmux(1) を参照下さい。

キーバインディング	意味
^A ?	ヘルプスクリーンを表示 (キーバインディングを表示)
^A c	新規ウィンドウを作成しそれに切り替える
^A n	次のウィンドウに切り替える
^A p	前のウィンドウに切り替える
^A 0	0 番のウィンドウに切り替える
^A 1	1 番のウィンドウに切り替える
^A w	ウィンドウのリストを表示
^A a	Ctrl-A を現在のウィンドウにキーボード入力として送る
^A h	現在のウィンドウのハードコピーをファイルに書く
^A H	現在のウィンドウのファイルへのロギングを開始/終了する
^A ^X	ターミナルをロック (パスワードで保護)
^A d	ターミナルから screen のセッションをデタッチ
^A DD	screen のセッションをデタッチしてログアウト

Table 9.2: screen キーバインディングのリスト

9.1.3 ディレクトリー間移動

項 1.4.2 にて、ディレクトリー間の俊敏な移動を可能にする 2 つのティップが記述されています: `$CDPATH` と `mc`。

ファジーテキストフィルターを使えば、正確なパスをタイプ無しでも可能です。`fzf` の場合だと、`~/.bashrc` に以下を含めます。

```
FZF_KEYBINDINGS_PATH=/usr/share/doc/fzf/examples/key-bindings.bash
FZF_COMPLETION_PATH=/usr/share/doc/fzf/examples/completion.bash
if [ -f $FZF_KEYBINDINGS_PATH ] && [ -f $FZF_COMPLETION_PATH ]; then
  . $FZF_KEYBINDINGS_PATH
  . $FZF_COMPLETION_PATH
fi
```

たとえば:

- 最小限の努力で非常に奥深いサブディレクトリーまでジャンプできます。最初に"`cd **`" とタイプして `Tab` を押します。すると、候補のパスとともに入力を促されます。例えば、`s/d/b foo` のような部分パス文字列をタイプすると候補パスが絞られます。`cd` が使うパスをカーソールとリターンキーで選択します。
- 最小限の努力でコマンド履歴からより効率的にコマンドを選択できます。コマンドプロンプトで `Ctrl-R` を押します。すると、候補のコマンドとともに入力を促されます。例えば、`vim d` のような部分コマンド文字列をタイプすると候補が絞られます。使うコマンドをカーソールとリターンキーで選択します。

9.1.4 Readline のラッパー

`/usr/bin/dash` のようなコマンドライン履歴編集能力のないコマンドは `rlwrap` もしくはその等価プログラムのもとで透過的にそのような機能を追加できます。

```
$ rlwrap dash -i
```

これは、`bash` のようなフレンドリーな環境下で、`dash` に関する微妙な点をテストする便利なプラットフォームを提供します。

9.1.5 ソースコードツリーのスキャン

`ripgrep` パッケージ中にある rg(1) コマンドは、典型的な状況にあるソースコードツリーをスキャンするための grep(1) コマンド代替の高速コマンドを提供します。現代的なマルチコア CPU をうまく利用するとともに、いくつかのファイルをスキップする合理的なフィルターを自動的に適用します。

9.2 Vim のカスタム化

項 1.4.8 で vim(1) の基本を学んだ後は、vim を以下に使うべきかを理解するために、Bram Moolenaar 氏の”Seven habits of effective text editing (2000)” を読んで下さい。

9.2.1 内部機能を使った vim のカスタマイズ

vim の挙動は”set ...” 等の Ex モードコマンドを通して、その内部機能を有効にすることで大幅に変更できます。

Ex コマンドは、伝統的な”~/.vimrc” または git-friendly な”~/.vim/vimrc” という、ユーザーの vimrc ファイルに含められます。以下は非常に単純な一例です1:

```
""" Generic baseline Vim and Neovim configuration (~/.vimrc)
"""   - For NeoVim, use "nvim -u ~/.vimrc [filename]"
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
let mapleader = ' '             " :h mapleader
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
set nocompatible                " :h 'cp -- sensible (n)vim mode
syntax on                       " :h :syn-on
filetype plugin indent on       " :h :filetype-overview
set encoding=utf-8              " :h 'enc (default: latin1) -- sensible encoding
""" current vim option value can be verified by :set encoding?
set backspace=indent,eol,start  " :h 'bs (default: nobs) -- sensible BS
set statusline=%<%f%m%r%h%w%=%y[U+%04B]%2l/%2L=%P,%2c%V
set listchars=eol:¶,tab:b''▫b''\ ,extends:b''▫b'',precedes:b''▫b'',nbsp:b''▫b''
set viminfo=!,'100,<5000,s100,h " :h 'vi -- bigger copy buffer etc.
""" Pick "colorscheme" from blue darkblue default delek desert elflord evening
""" habamax industry koehler lunaperche morning murphy pablo peachpuff quiet ron
""" shine slate torte zellner
colorscheme industry
""" don't pick "colorscheme" as "default" which may kill SpellUnderline settings
set scrolloff=5                 " :h 'scr -- show 5 lines around cursor
set laststatus=2                " :h 'ls (default 1)  k
""" boolean options can be unset by prefixing "no"
set ignorecase                  " :h 'ic
set smartcase                   " :h 'scs
set autoindent                  " :h 'ai
set smartindent                 " :h 'si
set nowrap                      " :h 'wrap
"set list                        " :h 'list (default nolist)
set noerrorbells                " :h 'eb
set novisualbell                " :h 'vb
set t_vb=                       " :h 't_vb -- termcap visual bell
set spell                       " :h 'spell
set spelllang=en_us,cjk         " :h 'spl -- english spell, ignore CJK
set clipboard=unnamedplus       " :h 'cb -- cut/copy/paste with other app
set hidden                      " :h 'hid
set autowrite                   " :h 'aw
set timeoutlen=300              " :h 'tm
```

vim のキーマップはユーザーの vimrc ファイルで変更可能です。例えば:



注意
よほどいい理由がない限りデフォルトのキーバインディングを変えようとしてはいけません。

1さらに凝ったカスタマイズ例: ”Vim Galore”, ”sensible.vim”, ...

```
""" Popular mappings (imitating LazyVim etc.)
""" Window moves without using CTRL-W which is dangerous in INSERT mode
nnoremap <C-H> <C-W>h
nnoremap <C-J> <C-W>j
nnoremap <C-K> <C-W>k
silent! nnoremap <C-L> <C-W>l
""" Window resize
nnoremap <C-LEFT> <CMD>vertical resize -2<CR>
nnoremap <C-DOWN> <CMD>resize -2<CR>
nnoremap <C-UP> <CMD>resize +2<CR>
nnoremap <C-RIGHT> <CMD>vertical resize +2<CR>
""" Clear hlsearch with <ESC> (<C-L> is mapped as above)
nnoremap <ESC> <CMD>noh<CR><ESC>
inoremap <ESC> <CMD>noh<CR><ESC>
""" center after jump next
nnoremap n nzz
nnoremap N Nzz
""" fast "jk" to get out of INSERT mode (<ESC>)
inoremap  jk <CMD>noh<CR><ESC>
""" fast "<ESC><ESC>" to get out of TERM mode (CTRL-\ CTRL-N)
tnoremap <ESC><ESC> <C-\><C-N>
""" fast "jk" to get out of TERM mode (CTRL-\ CTRL-N)
tnoremap jk <C-\><C-N>
""" previous/next trouble/quickfix item
nnoremap [q <CMD>cprevious<CR>
nnoremap ]q <CMD>cnext<CR>
""" buffers
nnoremap <S-H> <CMD>bprevious<CR>
nnoremap <S-L> <CMD>bnext<CR>
nnoremap [b <CMD>bprevious<CR>
nnoremap ]b <CMD>bnext<CR>
""" Add undo break-points
inoremap  , ,<C-G>u
inoremap  . .<C-G>u
inoremap  ; ;<C-G>u
""" save file
inoremap <C-S> <CMD>w<CR><ESC>
xnoremap <C-S> <CMD>w<CR><ESC>
nnoremap <C-S> <CMD>w<CR><ESC>
snoremap <C-S> <CMD>w<CR><ESC>
""" better indenting
vnoremap < <gv
vnoremap > >gv
""" terminal (Somehow under Linux, <C-/> becomes <C-_> in Vim)
nnoremap <C-_> <CMD>terminal<CR>
"nnoremap <C-/> <CMD>terminal<CR>
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
if ! has('nvim')
""" Toggle paste mode with <SPACE>p for Vim (no need for Nvim)
set pastetoggle=<leader>p
""" nvim default mappings for Vim.  See :h default-mappings in nvim
""" copy to EOL (no delete) like D for d
noremap Y y$
""" sets a new undo point before deleting
inoremap <C-U> <C-G>u<C-U>
inoremap <C-W> <C-G>u<C-W>
""" <C-L> is re-purposed as above
""" execute the previous macro recorded with Q
nnoremap Q @@
""" repeat last substitute and *KEEP* flags
nnoremap & :&&<CR>
```

```
""" search visual selected string for visual mode
xnoremap * y/\V<C-R>"<CR>
xnoremap # y?\V<C-R>"<CR>
endif
```

上記のキーバインディングが適正に機能するには、Backspace キーが”ASCII DEL” を生成し、Delete キーが”Escape sequence” を生成するように、ターミナルプログラムが設定される必要があります。

他のいろいろな設定もユーザーの vimrc ファイルで変更可能です。例えば:

```
""" Use faster 'rg' (ripgrep package) for :grep
if executable("rg")
  set grepprg=rg\ --vimgrep\ --smart-case
  set grepformat=%f:%l:%c:%m
endif
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
""" Retain last cursor position :h '"
augroup RetainLastCursorPosition
  autocmd!
  autocmd BufReadPost *
    \ if line("'\"") > 0 && line ("'\"") <= line("$") |
    \   exe "normal! g'\"" |
    \ endif
augroup END
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
""" Force to use underline for spell check results
augroup SpellUnderline
  autocmd!
  autocmd ColorScheme * highlight SpellBad term=Underline gui=Undercurl
  autocmd ColorScheme * highlight SpellCap term=Underline gui=Undercurl
  autocmd ColorScheme * highlight SpellLocal term=Underline gui=Undercurl
  autocmd ColorScheme * highlight SpellRare term=Underline gui=Undercurl
augroup END
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
""" highlight tailing spaces except when typing as red (set after colorscheme)
highlight TailingWhitespaces ctermbg=red guibg=red
""" \s\+     1 or more whitespace character: <Space> and <Tab>
""" \%#\@<!  Matches with zero width if the cursor position does NOT match.
match TailingWhitespaces /\s\+\%#\@<!$/
```

9.2.2 外部パッケージを使った **vim** のカスタマイズ

興味深い外部プラグインパッケージがあります:

- Vim - 至る所にあるテキストエディター -- Vim と vim スクリプトの公式アップストリームサイト
- VimAwesome -- Vim プラグインのリスト
- vim-scripts -- Debian パッケージ: vim スクリプト集

ユーザーの vimrc ファイルを用いて vim-scripts パッケージ中のプラグインパッケージを有効化できます。例えば:

```
packadd! secure-modelines
packadd! winmanager
" IDE-like UI for files and buffers with <space>w
nnoremap <leader>w         :WMToggle<CR>
```

新しいネイティブの Vim パッケージシステムは”git” や”git submodule” とうまく機能します。そのような設定例のひとつは私の git レポジトリー: dot-vim にあります。これは本質的に以下をします:

- ”git” と”git submodule” を使って、”*name*” のような最新の外部パッケージを ~/.vim/pack/*/opt/*name* に置くようなことをします。
- :packadd! *name* ラインをユーザーの vimrc ファイルに追加することで、これらのパッケージは runtimepath 上に置かれます。
- Vim はその初期化中に runtimepath 上のこのようなパッケージをロードします。
- 初期化の最後に、インストールされたドキュメントのタグが”helptags ALL” で更新されます。

詳しくは、vim を”vim --startuptime vimstart.log” とともに起動し、実際の起動順や各段階の時間を確認しましょう。

vim への外部パッケージを管理したりロードするのに多すぎる手法2があるのには混乱させられます。オリジナルの情報を確認するのが最適の解決法です。

キーストローク	情報
:help package	vim パッケージメカニズムに関する説明
:help runtimepath	runtimepath メカニズムに関する説明
:version	vimrc ファイル用の候補を含めた内部状態
:echo $VIM	vimrc ファイルを見つけるのに用いる”$VIM” 環境変数
:set runtimepath?	全実行時サポートファイルを探す対象のディレクトリーのリスト
:echo $VIMRUNTIME	システムが供給した各種実行時サポートファイルを見つけるのに用いる環境変数”$VIMRUNTIME”

Table 9.3: vim 初期化に関する情報

9.3 データーの記録と表現

9.3.1 ログデーモン

多くの伝統的プログラムは”/var/log/” ディレクトリーの下にそれぞれの活動をテキストファイル形式で記録します。

logrotate(8) が、大量のログファイルを生成するシステム上のログファイルの管理を簡略化するのに使われます。

多くの新規プログラムは”/var/log/journal” ディレクトリーの下に systemd-journald(8) の記録サービスを使ってそれぞれの活動をバイナリファイル形式で記録します。

systemd-cat(1) コマンドを使ってシェルスクリプトから systemd-journald(8) ジャーナルにデーターをログできます。

項 3.5 と項 3.4 を参照下さい。

9.3.2 ログアナライザー

注目すべきログアナライザー (aptitude(8) で”~Gsecurity::log-analyzer”) を次に記します。

注意
CRM114 は TRE 正規表現ライブラリーを使うファジーなフィルターを書く言語インフラを提供します。そのよくある応用はスパムメールのフィルターですが、ログアナライザーとしても使えます。

2vim-pathogen は人気がありました。

パッケージ	ポプコン	サイズ	説明
fail2ban	V:101, I:112	2275	複数回の認証エラーを発生させる IP を使用禁止にします
logwatch	V:9, I:10	2440	綺麗な出力の Perl で書かれたログアナライザー
awstats	V:4.7, I:7.5	6934	強力で機能の多いウェッブサーバーのログアナライザー
analog	V:3, I:81	3739	ウェッブサーバーのログアナライザー
sarg	V:0.61, I:0.67	863	squid の分析レポートジェネレター
pflogsumm	V:1.4, I:3.4	173	Postfix ログ項目サマライザー
fwlogwatch	V:0.11, I:0.15	487	ファイアウォールログアナライザー
squidview	V:0.04, I:0.38	224	squid の access.log ファイルのモニターと分析
swatch	V:0.07, I:0.28	99	正規表現マッチ、ハイライト、フック機能付きログファイルビューワー
crm114	V:0.05, I:0.30	1371	制御可能な正規表現切断機とスパムフィルター (CRM114)
icmpinfo	V:0.01, I:0.34	42	ICMP メッセージの解釈

Table 9.4: システムログアナライザーのリスト

9.3.3 テキストデーターのカスタム化表示

more(1) や less(1) 等のページャーツール (項 1.4.5 を参照下さい) や、ハイライトやフォーマット用のカスタムツール (項 11.1.8 を参照下さい) はテキストデーターを綺麗に表示できますが、汎用エディター (項 1.4.6 を参照下さい) が最も汎用性がありカスタム化が可能です。

ティップ
vim(1) やそのページャーモードのエイリアス view(1) では、"`:set hls`" とするとハイライトサーチが可能になります。

9.3.4 時間と日付のカスタム化表示

"`ls -l`" コマンドによる時間と日付のデフォールトの表示形式はロケール (値は項 1.2.6 を参照下さい) に依存します。"`$LANG`" 変数が最初に参照され、それを"`$LC_TIME`" か"`$LC_ALL`" のエクスポートされた環境変数によりオーバーライドする事ができます。

実際の各ロケールでのデフォールトの表示形式は使われた標準 C ライブラリー (`libc6` パッケージ) のバージョンに依存します。つまり Debian の異なるリリースは異なるデフォールトです。ISO 書式については、ISO 8601 を参照下さい。

ロケール以上にこの時間や日付の表示フォーマットをカスタム化したいと真摯に望むなら、"`--time-style`" 引数か"`$TIME_STYLE`" 値を使って時間スタイル値を設定するべきです (ls(1) と date(1) と"`info coreutils 'ls invocation'`" を参照下さい)。

ティップ
コマンドの別名を使えばコマンドライン上で長いオプションを入力しなくてもよくなります (項 1.5.9 を参照下さい)。

```
alias ls='ls --time-style=+%d.%m.%y %H:%M'
```

時間スタイル値	ロケール	時間と日付の表示
iso	任意	01-19 00:15
long-iso	任意	2009-01-19 00:15
full-iso	任意	2009-01-19 00:15:16.000000000 +0900
locale	C	Jan 19 00:15
locale	en_US.UTF-8	Jan 19 00:15
locale	es_ES.UTF-8	ene 19 00:15
+%d.%m.%y %H:%M	任意	19.01.09 00:15
+%d.%b.%y %H:%M	C または en_US.UTF-8	19.Jan.09 00:15
+%d.%b.%y %H:%M	es_ES.UTF-8	19.ene.09 00:15

Table 9.5: "ls -l" コマンドを時間スタイル値とともに用いた場合の時間と日付の例

9.3.5 着色化されたシェル出力

殆どの現代的なターミナルへのシェル出力は ANSI エスケープコードを使って着色化できます ("/usr/share/doc/xterm を参照下さい)。

例えば、次を試してみて下さい:

```
$ RED=$(printf "\x1b[31m")
$ NORMAL=$(printf "\x1b[0m")
$ REVERSE=$(printf "\x1b[7m")
$ echo "${RED}RED-TEXT${NORMAL} ${REVERSE}REVERSE-TEXT${NORMAL}"
```

9.3.6 着色化されたコマンド

着色化されたコマンドは対話環境で出力を検査するのに便利です。私は、私の"~/.bashrc" に次を含めています。

```
if [ "$TERM" != "dumb" ]; then
    eval "`dircolors -b`"
    alias ls='ls --color=always'
    alias ll='ls --color=always -l'
    alias la='ls --color=always -A'
    alias less='less -R'
    alias ls='ls --color=always'
    alias grep='grep --color=always'
    alias egrep='egrep --color=always'
    alias fgrep='fgrep --color=always'
    alias zgrep='zgrep --color=always'
else
    alias ll='ls -l'
    alias la='ls -A'
fi
```

エイリアスを使うことで色効果を対話コマンド使用時に限定します。こうすると less(1) 等のページャープログラムの下でも色を見られるので、環境変数"export GREP_OPTIONS='--color=auto'" をエキスポートするより都合が良いです。他のプログラムにパイプする際に色を使いたくなければ、先ほどの"~/.bashrc" 例中で代わりに"--color=auto" とします。

ティップ
対話環境でシェルを"TERM=dumb bash" として起動することで、このような着色するエイリアスを無効にできます。

9.3.7 複雑な反復のためにエディターでの活動を記録

複雑な反復のためにエディターでの活動を記録できます。
Vim の場合以下のようにします。

- ”qa”: 名前付きレジスタ”a” にタイプした文字の記録を開始。
- …エディターでの活動
- ”q”: タイプした文字の記録を終了。
- ”@a”: レジスター”a” の内容を実行。

Emacs の場合は以下のようにします。

- ”C-x (”: キーボードマクロの定義開始。
- …エディターでの活動
- ”C-x)”: キーボードマクロの定義終了。
- ”C-x e”: キーボードマクロの実行。

9.3.8 X アプリケーションの画像イメージの記録

xterm の表示を含めた、X アプリケーションの画像イメージを記録するにはいくつか方法があります。

パッケージ	ポプコン	サイズ	screen	コマンド
gnome-screenshot	V:12, I:91	1115	Wayland	GNOME 用のスクリーンショットアプリケーション
flameshot	V:7, I:17	3542	Wayland	強化されたスクリーンショットアプリケーション
gimp	V:40, I:206	33386	Wayland + X	GUI メニュー中のスクリーンショット
x11-apps	V:32, I:440	2461	X	xwd(1)
imagemagick	V:7, I:266	82	X	import(1)
scrot	V:4, I:47	144	X	scrot(1)

Table 9.6: 画像の操作ツールのリスト

9.3.9 設定ファイルの変更記録

DVCS の助力で設定ファイルの変更を記録したり、Btrfs の上でシステムのスナップショットを作成したりする専用のツールがあります。
ローカルスクリプト項 10.2.3 アプローチも一策です。

9.4 プログラム活動の監視と制御と起動

プログラム活動は専用ツールを用いて監視と制御できます。

ティップ
procps パッケージはプログラム活動の監視と制御と起動の基本中の基本を提供します。このすべてを習得するべきです。

パッケージ	ポプコン	サイズ	説明
etckeeper	V:23, I:26	156	Git (デフォールト) か Mercurial か GNU Bazaar を使って設定ファイルとそのメタデーターを保存
timeshift	V:7, I:13	4700	rsync か BTRFS スナップショットを使うシステム回復ユーティリティー
snapper	V:6.3, I:8.4	2461	Linux ファイルシステムスナップショット管理ツール

Table 9.7: 設定の履歴を記録するパッケージのリスト

パッケージ	ポプコン	サイズ	説明
coreutils	V:914, I:1000	19751	nice(1): スケジューリングの優先順位の変更してプログラムを実行
bsdutils	V:445, I:1000	338	renice(1): 実行中プロセスのスケジューリングの優先順位を変更
procps	V:845, I:998	2692	"/proc" ファイルシステムのユーティリティー: ps(1) と top(1) と kill(1) と watch(1) 等
psmisc	V:404, I:705	950	"/proc" ファイルシステムのユーティリティー: killall(1) と fuser(1) と pstree(1) と pstree(1)
time	V:7, I:73	148	time(1): 時間に関するシステムリソース使用状況を報告するためにプログラムを実行
sysstat	V:112, I:163	1904	sar(1)、iostat(1)、mpstat(1)、…: Linux 用のシステムパーフォーマンスツール
isag	V:0.1, I:3.1	109	sysstat の対話型システム活動グラフ化ソフト
lsof	V:463, I:949	492	lsof(8): "-p" オプションを使い実行中のプロセスが開いているファイルをリスト
strace	V:11, I:99	3250	strace(1): システムコールやシグナルを追跡
ltrace	V:1, I:11	420	ltrace(1): ライブラリーコールを追跡
xtrace	V:0.03, I:0.65	342	xtrace(1): X11 のクライアントとサーバーの間の通信を追跡
powertop	V:38, I:231	696	powertop(1): システムの電力消費情報
cron	V:929, I:997	248	cron(8) デーモンからバックグランドでスケジュール通りプロセスを実行
anacron	V:424, I:480	114	1 日 24 時間動作でないシステム用の cron 類似のコマンドスケジューラー
at	V:68, I:87	158	at(1) と batch(1) コマンド: 特定の時間や特定のロードレベル以下でジョブを実行

Table 9.8: プログラム活動の監視と制御のツールのリスト

9.4.1 プロセスの時間計測

コマンドにより呼び出されたプロセスにより使われた時間を表示します。

```
# time some_command >/dev/null
real    0m0.035s        # time on wall clock (elapsed real time)
user    0m0.000s        # time in user mode
sys     0m0.020s        # time in kernel mode
```

9.4.2 スケジューリングの優先度

ナイス値はプロセスのスケジューリングの優先度を制御するのに使われます。

ナイス値	スケジューリングの優先度
19	優先度が最低のプロセス (ナイス)
0	ユーザーにとっての優先度が非常に高いプロセス
-20	root にとっての優先度が非常に高いプロセス (非ナイス)

Table 9.9: スケジューリングの優先度のためのナイス値のリスト

```
# nice  -19 top                                        # very nice
# nice --20 wodim -v -eject speed=2 dev=0,0 disk.img # very fast
```

極端なナイス値はシステムに害を与えるかもしれません。本コマンドは注意深く使用下さい、

9.4.3 ps コマンド

Debian 上の ps(1) コマンドは BSD と SystemV 機能の両方をサポートしプロセスの活動を静的に特定するのに有用です。

スタイル	典型的コマンド	特徴
BSD	`ps aux`	%CPU %MEM を表示
System V	`ps -efH`	PPID を表示

Table 9.10: ps コマンドのスタイルのリスト

ゾンビ (動作していない) 子プロセスに関して、”PPID” フィールドで識別される親プロセス ID を使ってプロセスを停止できます。

pstree(1) コマンドはプロセスの木 (ツリー) を表示します。

9.4.4 top コマンド

Debian 上の top(1) は機能が豊富で、どのプロセスがおかしな動きをしているかを動的に識別することに役立ちます。

それはインタラクティブなフルスクリーンプログラムです。”h”-キーを押すことで使用法のヘルプが得られ、”q”-キーを押すことで終了できます。

9.4.5 プロセスによって開かれているファイルのリスト

プロセス ID (PID)、例えば 1 を使うプロセスによって開かれている全ファイルは以下のようにしてリストできます。

```
$ sudo lsof -p 1
```

PID=1 は通常 `init` プログラムです。

9.4.6 プログラム活動の追跡

プラグラムの活動状況は、システムコールとシグナルは strace(1) で、ライブラリーコールは ltrace(1) で、X11 のクライアントとサーバーの通信は xtrace(1) でプラグラムの活動状況を追跡できます。

`ls` コマンドのシステムコールを以下のようにして追跡できます。

```
$ sudo strace ls
```

ティップ
きれいなトリービューを作る **/usr/share/doc/strace/examples/** にある **strace-graph** スクリプトを使いましょう

9.4.7 ファイルやソケットを使っているプロセスの識別

例えば"`/var/log/mail.log`" 等のファイルを使っているプロセスは fuser(1) によって以下のようにして識別できます。

```
$ sudo fuser -v /var/log/mail.log
                     USER        PID ACCESS COMMAND
/var/log/mail.log:   root       2946 F.... rsyslogd
```

"`/var/log/mail.log`" ファイルが rsyslogd(8) コマンドによって書込みのために開かれている事が分かります。

例えば"`smtp/tcp`" 等のソケットを使っているプロセスは fuser(1) によって以下のようにして識別できます。

```
$ sudo fuser -v smtp/tcp
                     USER        PID ACCESS COMMAND
smtp/tcp:            Debian-exim   3379 F.... exim4
```

SMTP ポート (25) への TCP 接続を処理するためにあなたのシステムでは exim4(8) が実行されている事がこれで分かります。

9.4.8 一定間隔でコマンドを反復実行

watch(1) はプログラムを一定間隔で反復実行しながらフルスクリーンでその出力を表示します。

```
$ watch w
```

こうすると 2 秒毎更新でシステムに誰がログオンしているかを表示します。

9.4.9 ファイルに関してループしながらコマンドを反復実行

例えばグロブパターン"*.ext" へのマッチ等の何らかの条件にマッチするファイルに関してループしながらコマンドを実行する方法がいくつかあります。

- シェルの for-loop 法 (項 12.1.4 を参照下さい):

```
for x in *.ext; do if [ -f "$x"]; then command "$x" ; fi; done
```

- find(1) と xargs(1) の組み合わせ:

```
find . -type f -maxdepth 1 -name '*.ext' -print0 | xargs -0 -n 1 command
```

- コマンド付きの"-exec" オプションを使って find(1):

```
find . -type f -maxdepth 1 -name '*.ext' -exec command '{}' \;
```

- 短いシェルスクリプト付きの"-exec" オプションを使って find(1):

```
find . -type f -maxdepth 1 -name '*.ext' -exec sh -c "command '{}' && echo 'successful'" \;
```

上記の例はスペースを含む等の変なファイル名でも適正に処理できるように書かれています。find(1) に関する高度な使用法の詳細は項 10.1.5 を参照下さい。

9.4.10 GUI からプログラムをスタート

コマンドラインインターフェース (CLI) の場合、$PATH 環境変数で指定されるディレクトリー中で最初にマッチした名前のプログラムが実行されます。項 1.5.3 を参照下さい。

freedesktop.org スタンダード準拠のグラフィカルユーザーインターフェース (GUI) の場合、/usr/share/applications ディレクトリー中の *.desktop ファイルにより各プログラムの GUI メニュー表示に必要なアトリビュートが提供されます。Freedesktop.org の xdg メニューシステムに準拠する各パッケージは"/usr/share/applications/" の下に"*.desktop" で提供されるそのメニューデーターをインストールします。Freedesktop.org 標準に準拠する現代的なデスクトップ環境は xdg-utils パッケージを用いてその環境用のメニューを生成します。"/usr/share/doc/xdg-utils/README" を参照下さい。

例えば chromium.desktop ファイルは、プログラム名の"Name" や、プログラムの実行パスと引数の"Exec" や、使用するアイコンの"Icon" 等の属性 (Desktop Entry Specification 参照) を"Chromium Web Browser" に関して以下のようにして定義します:

```
[Desktop Entry]
Version=1.0
Name=Chromium Web Browser
GenericName=Web Browser
Comment=Access the Internet
Comment[fr]=Explorer le Web
Exec=/usr/bin/chromium %U
Terminal=false
X-MultipleArgs=false
Type=Application
Icon=chromium
Categories=Network;WebBrowser;
MimeType=text/html;text/xml;application/xhtml_xml;x-scheme-handler/http;x-scheme-handler/ ↩
    https;
StartupWMClass=Chromium
StartupNotify=true
```

これは簡略化しすぎた記述ですが、*.desktop ファイルは以下のようにしてスキャンされます。

デスクトップ環境は $XDG_DATA_HOME と $XDG_DATA_DIR 環境変数を設定します。例えば GNOME では:

- $XDG_DATA_HOME が未設定。(デフォルト値の $HOME/.local/share が使われます。)
- $XDG_DATA_DIRS は /usr/share/gnome:/usr/local/share/:/usr/share/ に設定されます。

以上により、ベースディレクトリー (XDG Base Directory Specification 参照) や applications ディレクトリーは以下となります。

- $HOME/.local/share/ → $HOME/.local/share/applications/
- /usr/share/gnome/ → /usr/share/gnome/applications/
- /usr/local/share/ → /usr/local/share/applications/
- /usr/share/ → /usr/share/applications/

*.desktop ファイルはこれらの applications ディレクトリーでこの順番でスキャンされます。

ティップ
ユーザーによるカスタムの GUI メニュー項目は *.desktop ファイルを $HOME/.local/share/applications/ ディレクトリーに追加することで生成できます。

ティップ
"Exec=..." 行はシェルが解釈しません。環境変数を設定する場合には env(1) コマンドを使います。

ティップ
同様に、もしこれらのベースディレクトリーの下の autostart ディレクトリーの中に *.desktop ファイルが作成されれば、*.desktop ファイル中に指定されたプログラムがデスクトップ環境が起動された時点に自動実行されます。Desktop Application Autostart Specification を参照下さい。

ティップ
同様に、もし $HOME/Desktop ディレクトリーの中に *.desktop ファイルが作成され、デスクトップ環境がローンチャーアイコンを表示する機能を有効としていれば、そこに指定されたプログラムがアイコンをクリックした際に実行されます。$HOME/Desktop ディレクトリーの実際の名前はロケール依存であることを承知下さい。xdg-user-dirs-update(1) を参照下さい。

9.4.11 スタートするプログラムのカスタム化

一部のプログラムは他のプログラムを自動的にスタートします。このプロセスをカスタム化する上でのチェックポイントを次に記します。

- アプリケーション設定メニュー:
  - GNOME デスクトップ: "Settings" → "Apps" → "Default Apps"
  - KDE デスクトップ: "Application Launcher" → "System Settings" → "Default Applications"
  - Iceweasel ブラウザー: "Edit" → "Preferences" → "Applications"
  - mc(1): "/etc/mc/mc.ext"

- ”$BROWSER” や”$EDITOR” や”$VISUAL” や”$PAGER” といった環境変数 (environ(7) 参照下さい)
- ”editor” や”view” や”x-www-browser” や”gnome-www-browser” や”www-browser” 等のプログラムに関する update-alternatives(8) システム (項 1.4.7 を参照下さい)
- MIME タイプとプログラムと関係づける、”~/.mailcap” や”/etc/mailcap” ファイルの内容 (mailcap(5) 参照下さい)
- ファイル拡張子と MIME タイプとプログラムと関係づける、”~/.mime.types” や”/etc/mime.types” ファイルの内容 (run-mailcap(1) 参照下さい)

ティップ
update-mime(8) は”/etc/mailcap.order” ファイルを使って”/etc/mailcap” ファイルを更新します (mailcap.order(5) 参照下さい)。

ティップ
debianutils パッケージは、どのエディターやページャーやウェッブブラウザーを呼び出すかに関してそれぞれ賢明な判断をする sensible-browser(1) や sensible-editor(1) や sensible-pager(1) を提供します。これらのシェルスクリプトを読む事をお薦めします。

ティップ
GUI の下で mutt のようなコンソールアプリケーションをあなたの好むアプリケーションとして実行するには、以下のようにして GUI アプリケーションを作成し、前記の方法であなたの好む起動されるアプリケーションとして”/usr/local/bin/mutt-term” を設定します。

```
# cat /usr/local/bin/mutt-term <<EOF
#!/bin/sh
gnome-terminal -e "mutt \$@"
EOF
# chmod 755 /usr/local/bin/mutt-term
```

9.4.12 プロセスの停止

kill(1) を使ってプロセス ID を使ってプロセスを停止 (プロセスへシグナルを送信) します。

killall(1) や pkill(1) プロセスコマンド名や他の属性を使ってプロセスを停止 (プロセスへシグナルを送信) します。

9.4.13 タスク 1 回実行のスケジュール

at(1) コマンドを以下のように実行して 1 回だけのジョブをスケジュールします。

```
$ echo 'command -args'| at 3:40 monday
```

9.4.14 タスク定期実行のスケジュール

cron(8) コマンドを実行して定期的タスクをスケジュールします。crontab(1) と crontab(5) を参照下さい。

例えば foo というノーマルユーザーとして”crontab -e” コマンドを使って”/var/spool/cron/crontabs/foo” という crontab(5) ファイルを作成することでプロセスをスケジュールして実行する事ができます。

crontab(5) ファイルの例を次に記します。

シグナル値	シグナル名	アクション	注釈
0	---	シグナルが送られていません (kill(2) を参照)	プロセスが実行中かチェック
1	SIGHUP	プロセスの終了	ターミナル接続の切断（シグナルがハングアップ）
2	SIGINT	プロセスの終了	キーボードから割り込み (CTRL-C)
3	SIGQUIT	プロセスを終了してコアをダンプ	キーボードから停止 (CTRL-\)
9	SIGKILL	プロセスの終了	ブロック不可能な kill シグナル
15	SIGTERM	プロセスの終了	ブロック可能な終了シグナル

Table 9.11: kill コマンドが良く使うシグナルのリスト

```
# use /usr/bin/sh to run commands, no matter what /etc/passwd says
SHELL=/bin/sh
# mail any output to paul, no matter whose crontab this is
MAILTO=paul
# Min Hour DayOfMonth Month DayOfWeek command (Day... are OR'ed)
# run at 00:05, every day
5  0  *  * *   $HOME/bin/daily.job >> $HOME/tmp/out 2>&1
# run at 14:15 on the first of every month -- output mailed to paul
15 14 1  * *   $HOME/bin/monthly
# run at 22:00 on weekdays(1-5), annoy Joe. % for newline, last % for cc:
0 22 *   * 1-5 mail -s "It's 10pm" joe%Joe,%%Where are your kids?%.%%
23 */2 1 2 *   echo "run 23 minutes after 0am, 2am, 4am ..., on Feb 1"
5  4 *   * sun echo "run at 04:05 every Sunday"
# run at 03:40 on the first Monday of each month
40 3 1-7 * *   [ "$(date +%a)" == "Mon" ] && command -args
```

ティップ
連続稼働していないシステムでは、機器のアップタイム上可能な限り指定間隔に近く定期的にコマンドをスケジュールするために anacron パッケージをインストールします。anacron(8) と anacrontab(5) を参照下さい。

ティップ
スケジュールされたシステムメインテナンススクリプトは、そのようなスクリプトを"/etc/cron.hourly/" か"/etc/cron.daily/" か"/etc/cron.weekly/" か"/etc/cron.monthly/" 中に置くことで root アカウントからそれらを定期的に実行できます。これらのスクリプトの実行時間は"/etc/crontab" と"/etc/anacrontab" でカスタム化できます。

Systemd は cron デーモンを使わずプログラムをスケジュールする低レベル能力があります。例えば、/lib/systemd/sys と /lib/systemd/system/apt-daily.service は、毎日の apt ダウンロード活動を設定しています。systemd.timer(5) を参照下さい。

9.4.15 イベントに合わせたタスクのスケジュール

Systemd は、タイマーイベントのみならずマウントイベントにもプログラムをスケジュールできます。例は、項 10.2.3.3 and 項 10.2.3.2 を参照下さい。

9.4.16 Alt-SysRq キー

Alt-SysRq (PrtScr) に続いて一つのキーを押すとシステムのレスキューコントロールの魔法をできます。

Alt-SysRq に続くキー	アクションの説明
k	全ての現仮想ターミナル上の全てのプロセスを停止 (**K**ll) (SAK)
s	データーが壊れないように全てのマウントされたファイルシステムを **s**ync (同期) します。
u	全てのマウントされたファイルシステムを読出し専用で再マウント (アンマウント、**u**mount)
r	X クラッシュの後でキーボードを **r**aw (生コード発生) モードから復旧

Table 9.12: 特記すべき SAK コマンドキーのリスト

詳しくは、Linux kernel user's and administrator's guide » Linux Magic System Request Key Hacks を参照下さい。

ティップ
SSH ターミナルなどからは、"/proc/sysrq-trigger" に書き込むことで Alt-SysRq 機能が使えます。例えば、リモートのシェルプロンプトから"echo s > /proc/sysrq-trigger; echo u > /proc/sysrq-trigger" とすると、全てのマウントされたファイルシステムを **s**ync (同期) して **u**mount (アンマウント) します。

現在 (2021 年) の Debian amd64 Linux カーネルでは `/proc/sys/kernel/sysrq=438=0b110110110` となっています:

- 2 = 0x2 - コンソールロギングレベルのコントロールを有効化 (ON)
- 4 = 0x4 - キーボード (SAK, unraw) のコントロールを有効化 (ON)
- 8 = 0x8 - プロセス等のデバグダンプを有効化 (OFF)
- 16 = 0x10 - sync コマンドを有効化 (ON)
- 32 = 0x20 - remount read-only を有効化 (ON)
- 64 = 0x40 - プロセスのシグナリング (term, kill, oom-kill) を有効化 (OFF)
- 128 = 0x80 - reboot/poweroff を許可する (ON)
- 256 = 0x100 - 全 RT タスクのナイス設定を許可する (ON)

9.5 システム管理ティップ

9.5.1 だれがシステムを利用している?

だれがシステムを利用しているかは、以下のようにしてチェックできます。

- who(1) は、誰がログオンしているかを表示します。
- w(1) は、誰がログオンしていて何をしているかを表示します。
- last(1) は、最後にログインしたユーザーのリストを表示します。
- lastb(1) は、最後にログイン失敗したユーザーのリストを表示します。

ティップ
"/var/run/utmp" と"/var/log/wtmp" はこのようなユーザー情報を保持します。login(1) と utmp(5) を参照下さい。

9.5.2 全員への警告

wall(1) を使うと、以下のようにしてシステムにログオンしている全員にメッセージを送れます。

```
$ echo "We are shutting down in 1 hour" | wall
```

9.5.3 ハードウエアーの識別

PCI 的デバイス (AGP、PCI-Express、CardBus、ExpressCard、等) では、(きっと"-nn" オプションとともに使う) lspci(8) がハードウエアー識別の良いスタート点です。

この代わりに、"/proc/bus/pci/devices" の内容を読むか、"/sys/bus/pci" の下のディレクトリーツリーを閲覧することでハードウエアーの識別ができます (項 1.2.12 を参照下さい)。

パッケージ	ポプコン	サイズ	説明
pciutils	V:279, I:994	289	Linux PCI ユーティリティー: lspci(8)
usbutils	V:146, I:894	322	Linux USB ユーティリティー: lsusb(8)
nvme-cli	V:30, I:39	2222	Linux 用の NVMe ユーティリティー: nvme(1)
pcmciautils	V:3.8, I:5.7	92	Linux のための PCMCIA ユーティリティー: pccardctl(8)
scsitools	V:0.2, I:2.0	261	SCSI ハードウエアー管理のためのツール集: lsscsi(8)
procinfo	V:0.3, I:5.7	149	"/proc" から得られるシステム情報: lsdev(8)
lshw	V:12, I:87	971	ハードウエアー設定に関する情報: lshw(1)
discover	V:26, I:527	81	ハードウエアー識別システム: discover(8)

Table 9.13: ハードウエアー識別ツールのリスト

9.5.4 ハードウエアー設定

GNOME や KDE のような現代的な GUI のデスクトップ環境ではほとんどのハードウエアー設定が付随する GUI 設定ツールを通じて管理できますが、それらの設定の基本的手法を知っておくのは良い事です。

キーボードやターミナルの設定に関しては、項 8.2 と項 8.3 を参照下さい。

パッケージ	ポプコン	サイズ	説明
console-setup	V:87, I:977	422	Linux コンソールのフォントとキーテーブルユーティリティー
keyd	V:1.7, I:1.8	1437	カーネルの下層の入力基本要素 (evdev, uinput) を用いる、キーボードキーのリマッピングデーモン
keyd-application-mapper	V:0.02, I:0.07	34	キーボードキーのリマッピングデーモン - 特定アプリケーションリマッパー
x11-xserver-utils	V:376, I:532	559	X サーバーユーティリティー: xset(1)、xmodmap(1)
acpid	V:51, I:74	158	Advanced Configuration and Power Interface (ACPI) によって起こるイベントの管理のためのデーモン
acpi	V:6, I:69	49	ACPI デバイス上の情報を表示するユーティリティー
sleepd	V:0.08, I:0.10	84	非使用状況のときにラップトップをスリープさせるデーモン
hdparm	V:111, I:192	246	ハードディスクアクセスの最適化 (項 9.6.9 を参照下さい)
smartmontools	V:249, I:274	2455	S.M.A.R.T. を使ってストレージシステムを制御監視
setserial	V:3.0, I:4.7	104	シリアルポートの管理ツール集
memtest86+	V:1, I:19	12483	メモリーハードウエアー管理のためのツール集
scsitools	V:0.2, I:2.0	261	SCSI ハードウエアー管理のためのツール集
setcd	V:0.06, I:0.59	33	コンパクトデバイスアクセス最適化
big-cursor	I:0.57	26	X のための大きなマウスカーソール

Table 9.14: ハードウエアー設定ツールのリスト

上記で、ACPI は APM より新しい電力管理システムの枠組みです。

ティップ
最近のシステム上の CPU フリーケンシースケーリングは acpi_cpufreq のようなカーネルモジュールで管理されています。

9.5.5 システムとハードウエアーの時間

以下はシステムとハードウエアーの時間を MM/DD hh:mm, CCYY (月/日時: 分, 年) に設定します。

```
# date MMDDhhmmCCYY
# hwclock --utc --systohc
# hwclock --show
```

Debian システムでは時間は地域の時間が普通表示されますが、ハードウエアーとシステムの時間は通常 UTC(GMT) を使います。

ハードウエアーの時間が UTC に設定されていれば”/etc/default/rcS” の中の設定を”UTC=yes” と変更します。

Debian システムが使うタイムゾーンは以下のようにして再設定できます。

```
# dpkg-reconfigure tzdata
```

ネットワーク経由でシステムの時間を更新したい場合には、`systemd-timesyncd` や `ntpsec` や `ntpsec-ntpdate` や `chrony` 等のパッケージを使って NTP サービスを利用することを考えます。

ティップ
systemd の下では、ネットワーク時間同期には上記と代わり `systemd-timesyncd` を使います。詳細は systemd-timesyncd(8) を参照下さい。

次を参照下さい。

- 正確な日時の管理ハウツー
- NTP 公共サービスプロジェクト
- `ntpsec-doc` パッケージ

ティップ
`ntpsec` パッケージ中の ntptrace(1) を使うと、NTP サービスの継がりを第一義的根源まで溯ることができます。

9.5.6 音のインフラ

現在の Linux のためのサウンドカードのためのデバイスドライバーは Advanced Linux Sound Architecture (ALSA) で提供されています。ALSA は過去の Open Sound System (OSS) と互換性のためのエミュレーションモードを提供します。

アプリケーションソフトはサウンドデバイスに直接アクセスするようにばかりでなく標準的なサウンドサーバーシステム経由で間接的にアクセスするように設定されているかもしれません。現在、PulseAudio や JACK や PipeWire がサウンドサーバーシステムとして使われています。最新の状況はサウンドに関する Debian wiki を参照下さい。

各ポピュラーなデスクトップ環境では通常共通のサウンドエンジンがあります。アプリケーションに使われるそれぞれのサウンドエンジンはそれと異なるサウンドサーバーにつなぐようにもできます。

ティップ
”`cat /dev/urandom > /dev/audio`” か speaker-test(1) を使ってスピーカをテストします。(^C で停止)

ティップ
音が出ない場合ですが、あなたのスピーカーが消音された出力につながっているかもしれません。現代的なサウンドシステムには多くの出力があります。`alsa-utils` パッケージ中の alsamixer(1) は音量や消音の設定をするのに便利です。

パッケージ	ポプコン	サイズ	説明
alsa-utils	V:349, I:457	2700	ALSA を設定し使用するユーティリティー
oss-compat	V:0.5, I:8.7	21	ALSA の下で”/dev/dsp not found” エラーを防ぐ OSS 互換性
pipewire	V:330, I:370	140	オーディオとビデオ処理エンジンのマルチメディアサーバー - メタパッケージ
pipewire-bin	V:336, I:370	2254	オーディオとビデオ処理エンジンのマルチメディアサーバー - オーディオサーバーと CLI プログラム
pipewire-alsa	V:175, I:243	197	オーディオとビデオ処理エンジンのマルチメディアサーバー - ALSA 代替オーディオサーバー
pipewire-pulse	V:304, I:350	56	オーディオとビデオ処理エンジンのマルチメディアサーバー - PulseAudio 代替オーディオサーバー
pulseaudio	V:144, I:166	6616	PulseAudio サーバー
libpulse0	V:445, I:578	984	PulseAudio クライアントライブラリー
jackd	V:2, I:13	8	JACK Audio Connection Kit. (JACK) サーバー (低遅延)
libjack0	V:1.7, I:8.3	320	JACK Audio Connection Kit. (JACK) ライブラリー (低遅延)
libphonon4qt5-4	V:16, I:38	572	Phonon: KDE サウンドエンジン

Table 9.15: サウンドパッケージのリスト

9.5.7 スクリーンセーバーの無効化

スクリーンセーバーを無効にするには、以下のコマンドを使います。

環境	コマンド
Linux コンソール	`setterm -powersave off`
X Window (スクリーンセーバー消去)	`xset s off`
X Window (dpms 無効)	`xset -dpms`
X Window (スクリーンセーバーの GUI 設定)	`xscreensaver-command -prefs`

Table 9.16: スクリーンセーバーを無効にするコマンドのリスト

9.5.8 ブザー音の無効化

PC スピーカーのコネクタを外すとブザー音は確実に無効にできます。pcspkr カーネルモジュールを削除すると同じ事ができます。

以下のようにすると bash(1) が使う readline(3) プログラムが警告文字 (ASCII=7) に出会った際にブザー音を発生するのを防げます。

```
$ echo "set bell-style none">> ~/.inputrc
```

9.5.9 メモリー使用状況

メモリー使用状況を確認するのに 2 つのリソースがあります。

- ”/var/log/dmesg” 中にあるカーネルブートメッセージには、利用可能なメモリーの正確な全サイズが書かれています。

- free(1) や top(1) は稼働中システムのメモリーリソース情報を表示します。

以下がその例です。

```
# grep '\] Memory' /var/log/dmesg
[    0.004000] Memory: 990528k/1016784k available (1975k kernel code, 25868k reserved, 931k ↩
     data, 296k init)
$ free -k
             total       used       free     shared    buffers     cached
Mem:        997184     976928      20256          0     129592     171932
-/+ buffers/cache:     675404     321780
Swap:      4545576          4    4545572
```

「dmesg は 990 MB 空いているという一方、free -k は 320 MB 空いていると言っている。600 MB 以上行方不明だ…」と不思議かもれません。

"Mem:" 行の"used" のサイズが大きかったり"free" のサイズが小さかったりについて悩まないでおきましょう。それらの 1 行下 (上記例では 675404 と 321780) を読んで安心して下さい。

1GB=1048576k の DRAM (video システムがこのメモリーの一部を使用) が付いている私の MacBook では以下のようになっています。

報告	サイズ
dmesg 中の全サイズ (Total)	1016784k = 1GB - 31792k
dmesg 中の未使用 (free)	990528k
shell 下での全 (total)	997184k
shell 下での未使用 (free)	20256k (しかし実質は 321780k)

Table 9.17: 報告されるメモリーサイズのリスト

9.5.10 システムのセキュリティーと整合性のチェック

ダメなシステム管理をするとあなたのシステムを外界からの攻撃にさらすことになるかもしれません。

システムのセキュリティーと整合性のチェックには、以下の事から始めるべきです。

- debsums パッケージ、debsums(1) と項 2.5.2 を参照下さい。
- chkrootkit パッケージ、chkrootkit(1) 参照下さい。
- clamav パッケージ類、clamscan(1) と freahclam(1) 参照下さい。
- Debian セキュリティー FAQ.
- Securing Debian Manual.

以下のシンプルなスクリプトを使うと、典型的な間違いの全員書込み可のファイルパーミッションをチェックできます。

```
# find / -perm 777 -a \! -type s -a \! -type l -a \! \( -type d -a -perm 1777 \)
```



注意
debsums パッケージはローカルに保存された MD5 チェックサムを使うので、悪意ある攻撃に対抗するセキュリティー監査ツールとしては完全には信頼できません。

パッケージ	ポプコン	サイズ	説明
logcheck	V:4.2, I:5.2	120	システムログの異常を管理者にメールするデーモン
debsums	V:4, I:29	108	MD5 チェックサムを使ってインストール済みパッケージファイルを検証するユーティリティー
chkrootkit	V:10, I:14	988	ルートキット検出ソフト
clamav	V:8, I:37	20551	Unix 用アンチウィルスユーティリティー - コマンドラインインターフェース
tiger	V:1.0, I:1.3	7800	システムセキュリティーの脆弱性を報告
tripwire	V:1.1, I:1.4	5060	ファイルやディレクトリーの整合性チェックソフト
john	V:1.1, I:7.1	469	アクティブなパスワードクラッキングツール
aide	V:2.1, I:2.7	328	先進的進入検出環境 - 静的ライブラリー
integrit	V:0.07, I:0.16	2759	ファイル整合性確認プログラム
crack	V:0.04, I:0.52	151	パスワード推定プログラム

Table 9.18: システムセキュリティーや整合性確認のためのツールリスト

9.6 データー保存のティップ

Linux live システムとしてあなたのシステムをブート (項 3.2.2 や項 3.2.3) すると、インストール済みシステム上のデーターストレージへの再設定が簡単になります。

注意
ハードディスク (HDD) に関する文言は、SSD / USB フラッシュデバイス / メモリーカード /... のようなストレージデバイスのような他のストレージデバイスにも適用できます。例中の/dev/sda のようなデバイス名は、該当するデバイス名 /dev/nvme0, /dev/mmcblk0, ... に置き換えます。

あるデバイスが GUI デスクトップシステム自動マウントされた場合、それらに操作を加える前に手動でコマンドラインからそのデバイスを umount(8) する必要があるかもしれません。

9.6.1 ディスク空間の利用状況

ディスク空間使用状況は mount と coreutils と xdu パッケージが提供するプログラムで評価できます:

- mount(8) はマウントされたファイルシステム (= ディスク) すべてを報告します。
- df(1) はファイルシステムのディスク空間使用状況を報告します。
- du(1) はディレクトリーツリーのディスク空間使用状況を報告します。

ティップ
du(8) の出力を xdu(1) に”du -k . |xdu” や”sudo du -k -x / |xdu” 等として注ぎ込むとそのグラフィカルでインタラクティブな表現が作成できます。

9.6.2 ディスクパーティション設定

ディスクのパーティションの設定に関して、fdisk(8) は標準と考えられてきていますが、parted(8) も注目に値します。” ディスクパーティションデーター” や” パーティションテーブル” や” パーティションマップ” や” ディスクラベル” は全て同意語です。

古い PC では、ディスクのパーティションデーターが最初のセクターとなる LBA セクター 0 (512 バイト) に保持される、古典的なマスターブートレコード (MBR) 方式が使われています。

Intel ベースの Mac を含むユニファイドエクステンシブルファームウェアインタフェース (UEFI) 付きの一部 PC では、ディスクパーティションデーターを最初のセクター以外に保持する GUID Partition Table (GPT) 方式が使われています。

fdisk(8) はディスクパーティションツールの標準でしたが、parted(8) がそれを置き換えつつあります。

パッケージ	ポプコン	サイズ	説明
util-linux	V:922, I:1000	4826	fdisk(8) と cfdisk(8) を含む雑多なシステムユーティリティー
parted	V:443, I:566	126	GNU Parted ディスクパーティションとリサイズのプログラム
gparted	V:12, I:81	2313	libparted ベースの GNOME パーティションエディター
gdisk	V:20, I:240	967	GPT/MBR ハイブリッドディスク用パーティションエディター
kpartx	V:16, I:23	78	パーティション用のデバイスマッピングを作成するプログラム

Table 9.19: ディスクパーティション管理パッケージのリスト



注意
parted(8) はファイルシステムを生成やリサイズも出きるということですが、そのようなことは mkfs(8) (mkfs.msdos(8) と mkfs.ext2(8) と mkfs.ext3(8) と mkfs.ext4(8) と…) とか resize2fs(8) 等の最もよくメンテされている専用ツールを使って行う方がより安全です。

注意
GPT と MBR 間で切り替えるには、ディスクの最初数ブロックの内容を直接消去し (項 9.8.6 を参照下さい)、"`parted /dev/sdx mklabel gpt`" か"`parted /dev/sdx mklabel msdos`" を使ってそれを設定する必要があります。ここで"`msdos`" が MBR のために使われていることを承知下さい。

9.6.3 UUID を使ってパーティションをアクセス

あなたのパーティションの再設定やリムーバブルストレージメディアのアクティベーション順はパーティションの名前を変えることになるかもしれませんが、それに首尾一貫してアクセスできます。もしディスクが複数ありあなたの BIOS/UEFI がそれに首尾一貫したデバイス名をつけない時にも、これは役に立ちます。

- "`-U`" オプションを使って mount(8) を実行すると"`/dev/sda3`" のようなファイル名を使うのではなく UUID を使ってブロックデバイスをマウントできます。
- "`/etc/fstab`" (fstab(5) 参照下さい) は UUID を使えます。
- ブートローダー (項 3.1.2) もまた UUID を使えます。

ティップ
ブロックスペシャルデバイスの UUID は blkid(8) を使って見極められます。
"`lsblk -f`" を使って UUID や他の情報も調査できます。

9.6.4 LVM2

LVM2 は Linux カーネル用の論理ボリュームマネージャーです。LVM2 を使うと、ディスクパーティションを物理的ハードディスクではなく論理ボリューム上の作成できるようになります。

LVM には以下が必要です。

- Linux カーネルによる device-mapper サポート (Debian カーネルではデフォルト)
- ユーザースペースの device-mapper サポートライブラリー (`libdevmapper*` パッケージ)
- ユーザースペースの LVM2 ツール (`lvm2` パッケージ)

以下のマンページから LVM2 を学び始めましょう。

- lvm(8): LVM2 機構の基本 (全 LVM2 コマンドのリスト)
- lvm.conf(5): LVM2 の設定ファイル
- lvs(8): 論理ボリュームの情報を報告します
- vgs(8): ボリュームグループの情報を報告します
- pvs(8): 物理ボリュームの情報を報告します

9.6.5 ファイルシステム設定

ext4 ファイルシステム用に `e2fsprogs` パッケージは次を提供します。

- 新規の ext4 ファイルシステムを作成するための mkfs.ext4(8)
- 既存の ext4 ファイルシステムをチェックと修理するための fsck.ext4(8)
- ext4 ファイルシステムのスーパーブロックを設定するための tune2fs(8)
- debugfs(8) を使って ext4 ファイルシステムをインタラクティブにデバグします。(削除したファイルを復元する `undel` コマンドがあります。)

mkfs(8) と fsck(8) コマンドは各種ファイルシステム依存プログラム (`mkfs.fstype` や `fsck.fstype`) のフロントエンドとして `e2fsprogs` により提供されています。ext4 ファイルシステム用は、mkfs.ext4(8) と fsck.ext4(8) で、それぞれ mke2fs(8) と e2fsck(8) にシムリンクされています。

Linux によってサポートされる各ファイルシステムでも、類似コマンドが利用可能です。

ティップ
Ext4 ファイルシステムは Linux システムのためのデフォルトのファイルシステムで、特定の使用しない理由がない限りこれを使用することが強く推奨されます。
Btrfs の状態は Debian wiki の btrfs や kernel.org wiki の btrfs に記されています。btrfs ファイルシステムは、ext4 ファイルシステム後継のデフォルトのファイルシステムとなると期待されています。
一部のツールはファイルシステムへのアクセスを Linux カーネルのサポート無しでも可能にします (項 9.8.2 を参照下さい)。

パッケージ	ポプコン	サイズ	説明
e2fsprogs	V:842, I:998	1560	ext2/ext3/ext4 ファイルシステムのためのユーティリティー
btrfs-progs	V:54, I:78	5291	btrfs ファイルシステムのためのユーティリティー
reiserfsprogs	V:12, I:21	473	Reiserfs ファイルシステムのためのユーティリティー
zfsutils-linux	V:32, I:33	1905	OpenZFS ファイルシステムのためのユーティリティー
dosfstools	V:274, I:568	310	FAT ファイルシステムのためのユーティリティー (Microsoft: MS-DOS, Windows)
exfatprogs	V:36, I:466	624	三星によってメンテナンスされている exFAT ファイルシステムのためのユーティリティー
exfat-fuse	V:2, I:37	73	FUSE による exFAT ファイルシステム (Microsoft) の読み書きドライバー。
xfsprogs	V:46, I:82	4651	XFS ファイルシステムのためのユーティリティー (SGI: IRIX)
ntfs-3g	V:197, I:510	1535	FUSE による NTFS ファイルシステム (Microsoft: Windows NT, ⋯) の読み書きドライバー。
jfsutils	V:0.4, I:6.6	1104	JFS ファイルシステムのためのユーティリティー (IBM: AIX, OS/2)
xorriso	V:16, I:66	378	libburnia 提供の、ISO-9660 ファイルシステムと CD/DVD 書き込みユーティリティー
wodim	V:7, I:81	898	cdrkit 提供の、CD/DVD 書き込みコマンド
genisoimage	V:17, I:150	1567	cdrkit 提供の、コマンドライン ISO-9660 ファイルシステムツール
hfsprogs	V:0.3, I:3.0	394	HFS と HFS Plus ファイルシステムのためのユーティリティー (Apple: Mac OS)
zerofree	V:9, I:114	30	ext2/3/4 ファイルシステムのフリーブロックをゼロにセットするプログラム

Table 9.20: ファイルシステム管理用パッケージのリスト

9.6.6 ファイルシステムの生成と整合性チェック

mkfs(8) コマンドは Linux システム上でファイルシステムを生成します。fsck(8) コマンドは Linux システム上でファイルシステムの整合性チェックと修理機能を提供します。

現在 Debian は、ファイルシステム形成後に定期的な `fsck` 無しがデフォルトです。



注意
一般的に `fsck` をマウントされているファイルシステムに実行することは安全ではありません。

ティップ
"`/etc/mke2fs.conf`" 中に"`enable_periodic_fsck`" と設定し、"`tune2fs -c0 /dev/`*`partition_name`*" を実行して最大マウント回数を 0 と設定すれば、リブート時に fsck(8) コマンドを root ファイルシステムを含む全ファイルシステムに安全に実行可能です。mke2fs.conf(5) と tune2fs(8) を参照下さい。
ブートスクリプトから実行される fsck(8) コマンドの結果を"`/var/log/fsck/`" 中のファイルからチェックします。

9.6.7 マウントオプションによるファイルシステムの最適化

"`/etc/fstab`" により静的なファイルシステム設定がなされます。例えば、

```
«file system»                    «mount point» «type» «options»    «dump» «pass»
proc                                      /proc proc   defaults          0 0
UUID=709cbe4c-80c1-56db-8ab1-dbce3146d2f7 /     ext4   errors=remount-ro 0 1
UUID=817bae6b-45d2-5aca-4d2a-1267ab46ac23 none  swap   sw                0 0
/dev/scd0                          /media/cdrom0  udf,iso9660 user,noauto  0 0
```

ティップ
UUID (項 9.6.3 を参照下さい) は、"`/dev/sda3`" や"`/dev/sda3`" 等の普通のブロックデバイス名の代わりにブロックデバイスを指定するのに使えます。

Linux 2.6.30 以来、カーネルは"`relatime`" オプションで提供される挙動をデフォルトとしています。

fstab(5) と mount(8) を参照下さい。

9.6.8 スーパーブロックによるファイルシステムの最適化

tune2fs(8) コマンドを用いてファイルシステムのスーパーブロックによってファイルシステムを最適化できます。

- "`sudo tune2fs -l /dev/sda1`" を実行すると"`/dev/sda1`" 上のファイルシステムスーパーブロックを表示します。
- "`sudo tune2fs -c 50 /dev/sda1`" を実行すると"`/dev/sda1`" 上のファイルシステムのチェック (ブートアップ時の `fsck` 実行) の頻度を 50 回のブート毎に変更します。
- "`sudo tune2fs -j /dev/sda1`" の実行は ext2 から ext3 へと"`/dev/sda1`" 上のファイルシステム変換してファイルシステムにジャーナリングの機能を追加します。(アンマウントしたファイルシステムに対して実行します。)

- ”sudo tune2fs -O extents,uninit_bg,dir_index /dev/sda1 && fsck -pf /dev/sda1”の実行は”/dev/ 上のファイルシステムを ext3 から ext4 に変換します。(アンマウントしたファイルシステムに対して実行します。)

ティップ
tune2fs(8) は、その名前にもかかわらず、ext2 ファイルシステムに機能するだけでなく ext3 とか ext4 ファイルシステムに関しても機能します。

9.6.9 ハードディスクの最適化



警告
ハードディスクの設定はデーターの整合性にとって非常に危険な事なので、その設定をさわる前にお使いのハードウエアーをチェックし hdparm(8) のマンページをチェックします。

例えば”/dev/sda” に対して”hdparm -tT /dev/sda” とするとハードディスクのアクセス速度をテストできます。(E)IDE を使って接続された一部のハードディスクでは、”(E)IDE 32 ビット I/O サポート” を有効にし”using_dma フラグ” を有効にし”interrupt-unmask フラグ” を設定し” 複数 16 セクター I/O” を設定するように、”hdparm -q -c3 -d1 -u1 -m16 /dev/sda” とすると高速化できます (危険です!)。

例えば”/dev/sda” に対して”hdparm -W /dev/sda” とするとハードディスクの書込みキャッシュ機能をテストできます。”hdparm -W 0 /dev/sda” とするとハードディスクの書込みキャッシュ機能を無効にできます。

不良プレスの CDROM を現代的な高速 CD-ROM ドライブで読むには、”setcd -x 2” としてそれを減速して使えば読めるかもしれません。

9.6.10 ソリッドステートドライブの最適化

現在、ソリッドステートドライブ (SSD) は自動検出されます。

揮発性のデーターパスの上に”tmpfs” を /etc/fstab でマウントすることで、不必要なディスクアクセスを減らしてディスクの消耗りを防ぎます。

9.6.11 SMART を用いたハードディスクの破壊の予測

smartd(8) デーモンを使うと SMART に文句を言うハードディスクの監視と記録ができます。

1. BIOS の SMART 機能を有効にします。
2. smartmontools パッケージをインストールします。
3. df(1) を使ってリストすることであなたのハードディスクを識別します。
   - 監視対象のハードディスクを”/dev/sda” と仮定します。
4. SMART 機能が実際に有効となっているかを”smartctl -a /dev/sda” のアウトプットを使ってチェックします。
   - もし有効でない場合には、”smartctl -s on -a /dev/sda” として有効にします。
5. 以下のようにして smartd(8) デーモンを実行します。
   - ”/etc/default/smartmontools” ファイル中の”start_smartd=yes” をアンコメントします。

- "`sudo systemctl restart smartmontools`" として smartd(8) デーモンを再実行します。

ティップ
smartd(8) デーモンは、警告の通知の仕方を含めて `/etc/smartd.conf` ファイルを用いてカスタム化できます。

9.6.12 $TMPDIR 経由で一時保存ディレクトリーを指定

普通アプリケーションは一時保存ディレクトリー"`/tmp`" のもとに一時ファイルを作成します。もし"`/tmp`" が十分なスペースを提供できない場合、行儀のいいプログラムなら `$TMPDIR` 変数を使ってそのような一時保存ディレクトリを指定できます。

9.6.13 LVM を使う使用可能なストレージ空間の拡張

インストール時に論理ボリュームマネージャー (LVM) (Linux 機能) 上に作られたパーティションは、大掛かりなシステムの再設定無しに複数のストレージデバイスにまたがる LVM 上のエクステントを継ぎ足したりその上のエクステントを切り捨てることで簡単にサイズ変更が出きます。

9.6.14 他パーティションをマウントする使用可能なストレージ空間の拡張

空のパーティションがあれば (例えば"`/dev/sdx`")、それを mkfs.ext4(1) を使ってフォーマットし、それをあなたが空間をより必要とするディレクトリーに mount(8) することができます。(元来あったデーター内容はコピーする必要があります。)

```
$ sudo mv work-dir old-dir
$ sudo mkfs.ext4 /dev/sdx
$ sudo mount -t ext4 /dev/sdx work-dir
$ sudo cp -a old-dir/* work-dir
$ sudo rm -rf old-dir
```

ティップ
上記の代わりに、空のディスクイメージファイル (項 9.7.5 を参照下さい) をループデバイスとしてマウントする (項 9.7.3 を参照下さい) 事もできます。実際のディスク使用は実際にデーターを溜め込むとともに成長します。

9.6.15 他ディレクトリーをバインドマウントする使用可能なストレージ空間の拡張

使える空間がある他のパーティション中に空のディレクトリーがあれば (例えば"`/path/to/emp-dir`")、そのディレクトリーを `--bind` オプションを使って、空間を必要としているディレクトリー (例えば"`work-dir`") にマウントすることができます。

```
$ sudo mount --bind /path/to/emp-dir work-dir
```

9.6.16 他ディレクトリーをオーバーレーマウントすることで使用可能なストレージ空間を拡張

Linux カーネル 3.18 以降 (Debian Stetch 9.0 以降) を使うと、他のパーティション中に使える空間 (例えば"/path/to/empty"と"/path/to/work") があれば、その中にディレクトリーを作成し、容量が必要な古いディレクトリー (e.g., "/path/to/old") の上に OverlayFS を使って積み重ねることができます。

```
$ sudo mount -t overlay overlay \
  -olowerdir=/path/to/old-dir,upperdir=/path/to/empty,workdir=/path/to/work
```

ここで、"/path/to/old" 上に書き込むには、読み書きが許可されたパーティション上に"/path/to/empty"と"/path/to/work" があることが必要です。

9.6.17 シムリンクを使う使用可能なストレージ空間の拡張



注意
ここに書かれている事は非推奨です。ソフトウェアーによっては「ディレクトリーへのシムリンク」ではうまく機能しません。上記の「マウントする」アプローチを代わりに使って下さい。

使える空間がある他のパーティション中に空のディレクトリーがあれば (例えば"/path/to/emp-dir")、そのディレクトリーへ ln(8) を使ってシムリンクを作成することができます。

```
$ sudo mv work-dir old-dir
$ sudo mkdir -p /path/to/emp-dir
$ sudo ln -sf /path/to/emp-dir work-dir
$ sudo cp -a old-dir/* work-dir
$ sudo rm -rf old-dir
```



警告
" ディレクトリーへのシムリンク" を"/opt" のようなシステムが管理するディレクトリーに使用してはいけません。システムがアップグレードされる際にそのようなシムリンクは上書きされるかもしれません。

9.7 ディスクイメージ

次に、ディスクイメージの操作を論じます。

9.7.1 ディスクイメージの作成

マウントされていない、例えば 2 番目の SCSI か ATA ドライブの"/dev/sdb"、デバイスのディスクイメージファイル"disk.img" は、以下のひとつで書き込めます。

```
# dd if=/dev/sdb of=disk.img; sync
```

```
# cp /dev/sdb disk.img ; sync
```

```
# cat /dev/sdb > disk.img ; sync
```

プライマリ IDE ディスクの最初のセクターにある伝統的 PC のマスターブートレコード (MBR) (項 9.6.2 を参照下さい) のディスクイメージは、dd(1) を用いれば以下のようにして作れます。

```
# dd if=/dev/sda of=mbr.img bs=512 count=1
# dd if=/dev/sda of=mbr-nopart.img bs=446 count=1
# dd if=/dev/sda of=mbr-part.img skip=446 bs=1 count=66
```

- "mbr.img": パーティションテーブル付きの MBR
- "mbr-nopart.img": パーティションテーブル抜きの MBR。
- "mbr-part.img": MBR のパーティションテーブルのみ。

オリジナルディスクのパーティションのイメージを作る場合には、"/dev/sda" を"/dev/sda1" 等で置き換えます。

9.7.2 ディスクに直接書込み

ディスクイメージファイル"disk.img" は、例えばサイズがマッチする 2 番目の SCSI ドライブ"/dev/sdb" に、、以下のひとつで書き込めます。

```
# dd if=disk.img of=/dev/sdb ; sync
```

```
# cp disk.img /dev/sdb ; sync
```

```
# cat disk.img >/dev/sdb ; sync
```

同様にディスクパーティションイメージファイル"partition.img" はサイズがマッチする例えば"/dev/sdb1" という 2 番目の SCSI ドライブの 1 番目のパーティションに以下のようにして書き込むことができます。

```
# dd if=partition.img of=/dev/sdb1 ; sync
```

9.7.3 ディスクイメージファイルをマウント

単一パーティションイメージを含むディスクイメージ"partition.img" は以下のように loop デバイスを使いマウントしアンマウントできます。

```
# losetup --show -f partition.img
/dev/loop0
# mkdir -p /mnt/loop0
# mount -t auto /dev/loop0 /mnt/loop0
...hack...hack...hack
# umount /dev/loop0
# losetup -d /dev/loop0
```

これは以下のように簡略化出来ます。

```
# mkdir -p /mnt/loop0
# mount -t auto -o loop partition.img /mnt/loop0
...hack...hack...hack
# umount partition.img
```

複数のパーティションを含むディスクイメージ"disk.img" の各パーティションは loop デバイスを使ってマウント出来ます。

```
# losetup --show -f -P disk.img
/dev/loop0
# ls -l /dev/loop0*
brw-rw---- 1 root disk   7,  0 Apr  2 22:51 /dev/loop0
brw-rw---- 1 root disk 259, 12 Apr  2 22:51 /dev/loop0p1
brw-rw---- 1 root disk 259, 13 Apr  2 22:51 /dev/loop0p14
brw-rw---- 1 root disk 259, 14 Apr  2 22:51 /dev/loop0p15
# fdisk -l /dev/loop0
Disk /dev/loop0: 2 GiB, 2147483648 bytes, 4194304 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 6A1D9E28-C48C-2144-91F7-968B3CBC9BD1

Device         Start     End Sectors  Size Type
/dev/loop0p1  262144 4192255 3930112  1.9G Linux root (x86-64)
/dev/loop0p14   2048    8191    6144    3M BIOS boot
/dev/loop0p15   8192  262143  253952  124M EFI System

Partition table entries are not in disk order.
# mkdir -p /mnt/loop0p1
# mkdir -p /mnt/loop0p15
# mount -t auto /dev/loop0p1 /mnt/loop0p1
# mount -t auto /dev/loop0p15 /mnt/loop0p15
# mount |grep loop
/dev/loop0p1 on /mnt/loop0p1 type ext4 (rw,relatime)
/dev/loop0p15 on /mnt/loop0p15 type vfat (rw,relatime,fmask=0002,dmask=0002,allow_utime ↩
    =0020,codepage=437,iocharset=ascii,shortname=mixed,utf8,errors=remount-ro)
...hack...hack...hack
# umount /dev/loop0p1
# umount /dev/loop0p15
# losetup -d /dev/loop0
```

この他、同様の効果は kpartx パッケージの kpartx(8) により作られるデバイスマッパーデバイスを用いて以下のようにして実現も出来ます。

```
# kpartx -a -v disk.img
add map loop0p1 (253:0): 0 3930112 linear 7:0 262144
add map loop0p14 (253:1): 0 6144 linear 7:0 2048
add map loop0p15 (253:2): 0 253952 linear 7:0 8192
# fdisk -l /dev/loop0
Disk /dev/loop0: 2 GiB, 2147483648 bytes, 4194304 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 6A1D9E28-C48C-2144-91F7-968B3CBC9BD1

Device         Start     End Sectors  Size Type
/dev/loop0p1  262144 4192255 3930112  1.9G Linux root (x86-64)
/dev/loop0p14   2048    8191    6144    3M BIOS boot
/dev/loop0p15   8192  262143  253952  124M EFI System

Partition table entries are not in disk order.
# ls -l /dev/mapper/
total 0
crw------- 1 root root 10, 236 Apr  2 22:45 control
lrwxrwxrwx 1 root root       7 Apr  2 23:19 loop0p1 -> ../dm-0
lrwxrwxrwx 1 root root       7 Apr  2 23:19 loop0p14 -> ../dm-1
lrwxrwxrwx 1 root root       7 Apr  2 23:19 loop0p15 -> ../dm-2
```

```
# mkdir -p /mnt/loop0p1
# mkdir -p /mnt/loop0p15
# mount -t auto /dev/mapper/loop0p1 /mnt/loop0p1
# mount -t auto /dev/mapper/loop0p15 /mnt/loop0p15
# mount |grep loop
/dev/loop0p1 on /mnt/loop0p1 type ext4 (rw,relatime)
/dev/loop0p15 on /mnt/loop0p15 type vfat (rw,relatime,fmask=0002,dmask=0002,allow_utime ↩
    =0020,codepage=437,iocharset=ascii,shortname=mixed,utf8,errors=remount-ro)
...hack...hack...hack
# umount /dev/mapper/loop0p1
# umount /dev/mapper/loop0p15
# kpartx -d disk.img
```

9.7.4 ディスクイメージのクリーニング

ディスクイメージファイル"disk.img"は消去済みのファイルを綺麗に無くした綺麗なスパースイメージ"new.img"に以下のようにしてできます。

```
# mkdir old; mkdir new
# mount -t auto -o loop disk.img old
# dd bs=1 count=0 if=/dev/zero of=new.img seek=5G
# mount -t auto -o loop new.img new
# cd old
# cp -a --sparse=always ./ ../new/
# cd ..
# umount new.img
# umount disk.img
```

もし"disk.img"が ext2 か ext3 か ext4 の場合には、zerofree パッケージの zerofree(8) を使うことも出来ます。

```
# losetup --show -f disk.img
/dev/loop0
# zerofree /dev/loop0
# cp --sparse=always disk.img new.img
# losetup -d /dev/loop0
```

9.7.5 空のディスクイメージ作成

5GiB まで成長可能な空のディスクイメージファイル"disk.img"は dd(1) を用いて以下のようにして作成できます。

```
$ dd bs=1 count=0 if=/dev/zero of=disk.img seek=5G
```

ここで dd(1) の利用に代え、特化した fallocate(8) の利用ができます。

loop デバイスを使ってこのディスクイメージ"disk.img"上に ext4 ファイルシステムを作成できます。

```
# losetup --show -f disk.img
/dev/loop0
# mkfs.ext4 /dev/loop0
...hack...hack...hack
# losetup -d /dev/loop0
$ du  --apparent-size -h disk.img
5.0G  disk.img
$ du -h disk.img
83M disk.img
```

”sparse” に関して、そのファイルサイズは 5.0GiB でその実ディスク使用はたったの 83MiB です。この相違は ext4 がスパースファイルを保持できるから可能となっています。

ティップ
スパースファイルによる実際のディスク使用はそこに書かれるデーターとともに成長します。

項 9.7.3 にあるように loop デバイスまたはデバイスマッパーデバイスによりデバイスに同様の操作をすることで、このディスクイメージ”disk.img” を parted(8) または fdisk(8) を使ってパーティションし mkfs.ext4(8) や mkswap(8) 等を使ってファイルシステムを作れます。

9.7.6 ISO9660 イメージファイル作成

ティップ
cdrkit が提供する genisoimage(1) や、Libburnia が提供する xorrisofs(1) はコマンド名以外のコマンドシンタックスを共有しています。

”source_directory” のソースディレクトリーツリーから作られる ISO9660 イメージファイル”cd.iso” は cdrkit が提供する genisoimage(1) を使って以下のようにして作成できます。

```
#  genisoimage -r -J -T -V volume_id -o cd.iso source_directory
```

同様に、ブート可能な ISO9660 イメージファイル”cdboot.iso” は、debian-installer のような”source_directory” にあるディレクトリーツリーから以下のようにして作成できます。

```
#  genisoimage -r -o cdboot.iso -V volume_id \
   -b isolinux/isolinux.bin -c isolinux/boot.cat \
   -no-emul-boot -boot-load-size 4 -boot-info-table source_directory
```

上記では、Isolinux ブートローダー (項 3.1.2 を参照下さい) がブートに使われています。

以下のようにすると CD-ROM デバイスから直接 md5sum 値を計算し ISO9660 イメージを作成できます。

```
$ isoinfo -d -i /dev/cdrom
CD-ROM is in ISO 9660 format
...
Logical block size is: 2048
Volume size is: 23150592
...
# dd if=/dev/cdrom bs=2048 count=23150592 conv=notrunc,noerror | md5sum
# dd if=/dev/cdrom bs=2048 count=23150592 conv=notrunc,noerror > cd.iso
```



警告
正しい結果を得るために上記のように Linux の ISO9660 ファイルシステム先読みバグを注意深く避けなければいけません。

9.7.7 CD/DVD-R/RW に直接書込み

ティップ
DVD i は cdrkit が提供する wodim(1) や、Libburnia が提供する xorrecord(1) にとって、DVD は大きな CD でしかありません。これらコマンドは、コマンド名以外のコマンドシンタックスを共有しています。

使えるデバイスは以下のようにするとみつかります。

```
# wodim --devices
```

そしてブランクの CD-R をドライブに挿入して、例えば”/dev/sda” というこのデバイスに ISO9660 イメージファイル”cd.iso” に wodim(1) を使って以下のようにして書込みます。

```
# wodim -v -eject dev=/dev/sda cd.iso
```

もし CD-R ではなく CD-RW が使われている場合には、次を代わりに実行して下さい。

```
# wodim -v -eject blank=fast dev=/dev/sda cd.iso
```

ティップ
もしあなたのデスクトップシステムが CD を自動的にマウントする場合、wodim(1) を使う前に”sudo unmount /dev/sda” として CD をアンマウントします。

9.7.8 ISO9660 イメージファイルをマウント

もし”cd.iso” の内容が ISO9660 イメージの場合、以下のようにするとそれを”/cdrom” に手動でマウントできます。

```
# mount -t iso9660 -o ro,loop cd.iso /cdrom
```

ティップ
現代的なデスクトップシステムでは ISO9660 フォーマットされた CD のようなリムーバブルメディアを自動的にマウントします (項 10.1.7 を参照下さい)。

9.8 バイナリーデーター

次に、ストレージメディア上のバイナリーデーターを直接操作することを論じます。

9.8.1 バイナリーデーターの閲覧と編集

もっとも基本的なバイナリーファイルを閲覧方法は”od -t x1” コマンドを使うことです。

ティップ
HEX は底が 16 の 16 進フォーマットです。OCTAL は底が 8 の 8 進フォーマットです。ASCII (アスキー) は情報交換用アメリカ標準コードで、普通の英文テキストです。EBCDIC (エビシディック) は IBM メインフレームオペレーティングシステム上で使われる拡張二進化十進数互換コードです。

9.8.2 ファイルシステムをマウントせずに操作

ファイルシステムをマウントせずに読出しや書込みをするツールがあります。

パッケージ	ポプコン	サイズ	説明
coreutils	V:914, I:1000	19751	ファイルをダンプする od(1) がある基本パッケージ (HEX, ASCII, OCTAL, …)
bsdmainutils	V:4, I:112	17	ファイルをダンプする hd(1) があるユーティリティーパッケージ (HEX, ASCII, OCTAL, …)
hexedit	V:0.7, I:7.4	70	バイナリーエディターとビューワー (HEX, ASCII)
ghex	V:0.7, I:7.2	2525	フル機能の 16 進エディター (GNOME)
okteta	V:1, I:11	1591	フル機能の 16 進エディター (KDE4)
ncurses-hexedit	V:0.1, I:1.1	130	バイナリーエディターとビューワー (HEX, ASCII, EBCDIC)

Table 9.21: バイナリーデーターを閲覧や編集するパッケージのリスト

パッケージ	ポプコン	サイズ	説明
mtools	V:7, I:51	404	MSDOS ファイルシステムをマウントせずに使うツール
hfsplus	V:2, I:20	248	HFS+ ファイルシステムをマウントせずに使うツール

Table 9.22: ファイルシステムをマウントせずに操作するパッケージのリスト

9.8.3 データーの冗長性

Linux カーネルが提供するソフトウェアー RAID システムは高いレベルのストレージ信頼性を達成するためにカーネルのファイルシステムのレベルでデーターの冗長性を提供します。

アプリケーションプログラムレベルでストレージの高い信頼性を達成するようにデーター冗長性を付加するツールもあります。

パッケージ	ポプコン	サイズ	説明
par2	V:10, I:115	965	ファイルのチェックと修理のためのパリティーアーカイブセット
dvdisaster	V:0.1, I:1.1	1422	CD/DVD メディアのデーターロス/傷つき/老化の防止
dvbackup	I:0.04	413	MiniDV カメラレコーダを使うバックアップツール

Table 9.23: ファイルにデーターの冗長性を追加するツールのリスト

9.8.4 データーファイルの復元と事故の証拠解析

データーファイルの復元と事故の証拠解析のツールがあります。

ティップ
`e2fsprogs` パッケージ中の debugfs(8) の `list_deleted_inodes` または `undel` コマンドを用いると ext2 ファイルシステム上でファイルのアンデリートができます。

9.8.5 大きなファイルを小さなファイルに分割

単一ファイルでバックアップするにはデーターが大きすぎる場合、そのファイル内容を例えば 2000MiB の断片にしてバックアップし、それらの断片を後日マージしてオリジナルのファイルに戻せます。

パッケージ	ポプコン	サイズ	説明
testdisk	V:2, I:25	529	パーティションのスキャンとディスク復元のためのユーティリティー
magicrescue	V:0.2, I:1.6	257	マジックバイトを探してファイルを復元するユーティリティー
scalpel	V:0.2, I:2.2	89	貿素で高性能なファイル彫刻刀
myrescue	V:0.2, I:1.9	81	破壊したハードディスクからデーターを救出
extundelete	V:0.4, I:7.1	152	ext3/4 ファイルシステム上のファイルの削除復元ユーティリティー
ext4magic	V:0.3, I:3.5	235	ext3/4 ファイルシステム上のファイルの削除復元ユーティリティー
ext3grep	V:0.1, I:1.7	299	ext3 ファイルシステム上のファイルの削除復元ヘルプツール
scrounge-ntfs	V:0.1, I:1.6	49	NTFS ファイルシステム上のデーター復元プログラム
gzrt	V:0.02, I:0.43	34	gzip 復元ツールキット
sleuthkit	V:3, I:24	1130	証拠解析のためのツール (Sleuthkit)
autopsy	V:0.1, I:1.1	1026	SleuthKit のための GUI
foremost	V:0.3, I:3.7	101	データー復元のための証拠解析アプリケーション
guymager	V:0.09, I:0.70	1049	Qt 使用の証拠解析用イメージ作成ソフト
dcfldd	V:0.2, I:2.4	113	証拠解析とセキュリティーのための dd の強化版

Table 9.24: データーファイルの復元と事故の証拠解析のリスト

```
$ split -b 2000m large_file
$ cat x* >large_file
```



注意
名前がかち合わないように”x” で始まるファイル名のファイルが無いようにします。

9.8.6 ファイル内容の消去

ログファイルのようなファイルの内容を消去するためには、rm(1) を使ってファイルを消去しその後新しい空ファイルを作成することは止めます。コマンド実行間にファイルがアクセスされているかもしれないのがこの理由です。以下のようにするのがファイル内容を消去する安全な方法です。

```
$ :>file_to_be_cleared
```

9.8.7 ダミーファイル

以下のコマンドはダミーや空のファイルを作成します。

```
$ dd if=/dev/zero    of=5kb.file bs=1k count=5
$ dd if=/dev/urandom of=7mb.file bs=1M count=7
$ touch zero.file
$ : > alwayszero.file
```

以下のファイルを見つかります。

- ”5kb.file” は 5KB のゼロの連続です。
- ”7mb.file” は 7MB のランダムなデーターです。
- ”zero.file” は 0 バイト長のファイルかもしれません。もしファイルが存在する時は、その mtime を更新しその内容と長さを保持します。
- ”alwayszero.file” は常に 0 バイト長ファイルです。もしファイルが存在する時は mtime を更新しファイル内容をリセットします。

9.8.8 ハードディスクの全消去

”/dev/sda” にある USB フラッシュドライブ等のハードディスク類似デバイス全体のデーターを完全に消すいくつかの方法があります。



注意
コマンドを実行する前にまず USB フラッシュドライブの場所を mount(8) を使ってチェックします。”/dev/sda” によって指し示されるデバイスは、あなたの全システムが存在する、SCSI ハードディスクやシリアル ATA ハードディスクかも知れません。

以下のようにしてデーターを 0 にリセットして全消去します。

```
# dd if=/dev/zero of=/dev/sda
```

以下のようにしてランダムデーターを上書きして全消去します。

```
# dd if=/dev/urandom of=/dev/sda
```

以下のようにしてランダムデーターを非常に効率的に上書きして全消去します。

```
# shred -v -n 1 /dev/sda
```

これに代え badblocks(8) を -t random オプションとともに用いることができる。

Debian インストーラ CD 等の多くのブート可能な Linux の CD のシェルから dd(1) が利用可能ですから、”/dev/sda” や”/dev/sda” 等のシステムハードディスクに対して同類のメディアから消去コマンドを実行することでインストールされたシステムを完全に消去することができます。

9.8.9 ハードディスク未使用部分の全消去

データーの消去はファイルシステムからアンリンクされているだけなので、例えば”/dev/sdb1” のようなハードディスク (USB フラッシュドライブ) 上の使用されていない領域には消去されたデーター自身が含まれているかもしれません。これらに上書きすることで綺麗に消去できます。

```
# mount -t auto /dev/sdb1 /mnt/foo
# cd /mnt/foo
# dd if=/dev/zero of=junk
dd: writing to `junk': No space left on device
...
# sync
# umount /dev/sdb1
```



警告
あなたの USB フラッシュドライブではこれで通常十分です。でもこれは完璧ではありません。消去されたファイル名や属性はファイルシステム中に隠れて残っているかもしれません。

9.8.10 削除されたがまだオープン中のファイルの復活法

ファイルをうっかり消去しても、そのファイルが何らかのアプリケーション (読出しか書込み) によって使われている限り、そのようなファイルを復元出来ます。

例えば、次を試してみて下さい:

```
$ echo foo > bar
$ less bar
$ ps aux | grep ' less[ ]'
bozo    4775  0.0  0.0  92200   884 pts/8    S+   00:18   0:00 less bar
$ rm bar
$ ls -l /proc/4775/fd | grep bar
lr-x------ 1 bozo bozo 64 2008-05-09 00:19 4 -> /home/bozo/bar (deleted)
$ cat /proc/4775/fd/4 >bar
$ ls -l
-rw-r--r-- 1 bozo bozo 4 2008-05-09 00:25 bar
$ cat bar
foo
```

この代わりに、(lsof パッケージがインストールされている時) もう一つのターミナルで以下のように実行します。

```
$ ls -li bar
2228329 -rw-r--r-- 1 bozo bozo 4 2008-05-11 11:02 bar
$ lsof |grep bar|grep less
less 4775 bozo 4r REG 8,3 4 2228329 /home/bozo/bar
$ rm bar
$ lsof |grep bar|grep less
less 4775 bozo 4r REG 8,3 4 2228329 /home/bozo/bar (deleted)
$ cat /proc/4775/fd/4 >bar
$ ls -li bar
2228302 -rw-r--r-- 1 bozo bozo 4 2008-05-11 11:05 bar
$ cat bar
foo
```

9.8.11 全てのハードリンクを検索

ハードリンクのあるファイルは"ls -li" を使って確認できます、

```
$ ls -li
total 0
2738405 -rw-r--r-- 1 root root 0 2008-09-15 20:21 bar
2738404 -rw-r--r-- 2 root root 0 2008-09-15 20:21 baz
2738404 -rw-r--r-- 2 root root 0 2008-09-15 20:21 foo
```

"baz" も"foo" もリンク数が"2" (>1) でハードリンクがある事を示しています。これらの inode 番号は共通の"2738404" です。これはこれらがハードリンクされた同じファイルということを意味します。ハードリンクされた全てのファイルを偶然うまく見つけられない場合は、それを例えば"2738404" という inode で以下のようにして探せます。

```
# find /path/to/mount/point -xdev -inum 2738404
```

9.8.12 見えないディスクスペースの消費

削除されたがオープンされたままのファイルは、普通の du(1) では見えませんが、ディスクスペースを消費します。これらは以下のようにすればそのサイズとともにリストできます。

```
# lsof -s -X / |grep deleted
```

9.9 データー暗号化ティップ

あなたの PC への物理的アクセスがあると、誰でも簡単に root 特権を獲得できあなたの PC の全てのファイルにアクセスできます (項 4.6.4 を参照下さい)。これが意味するところは、あなたの PC が盗まれた場合にログインのパスワードではあなたのプライベートでセンシティブなデーターを守れないと言うことです。それを達成するにはデーターの暗号化技術を適用しなければいけません。GNU プライバシーガード (項 10.3 を参照下さい) はファイルを暗号化できますが、少々手間がかかります。

dm-crypt は、ネイティブ Linux カーネルモジュール経由での自動データー暗号化を、最小限のユーザー努力で device-mapper 経由で可能にします。

パッケージ	ポプコン	サイズ	説明
cryptsetup	V:35, I:77	476	暗号化されたブロックデバイス (dm-crypt / LUKS) のためのユーティリティー
cryptmount	V:1.8, I:2.2	236	ノーマルユーザーによるマウント/アンマウントに焦点を当てた暗号化されたブロックデバイス (dm-crypt / LUKS) のためのユーティリティー
fscrypt	V:0.4, I:1.1	6844	Linux ファイルシステム暗号化 (fscrypt) 用のユーティリティー
libpam-fscrypt	V:0.32, I:0.35	6941	Linux ファイルシステム暗号化 (fscrypt) 用の PAM モジュール

Table 9.25: データー暗号化ユーティリティーのリスト



注意
データーの暗号化には CPU 時間等の負担がかかります。暗号化したデーターはそのパスワードを失うとアクセスできなくなります。暗号化の利益と負担の両天秤をして下さい。

注意
debian-installer (lenny 以降) を使うと、dm-crypt/LUKS と initramfs を使って、全 Debian システムを暗号化したディスク上にインストールできます。

ティップ
ユーザー空間での暗号化ユーティリティーに関しては項 10.3 を参照下さい: GNU プライバシーガード。

9.9.1 dm-crypt/LUKS を使ったリムーバブルディスクの暗号化

例えば"/dev/sdx" にある USB フラッシュドライブのようなリムーバブルストレージデバイスの内容を dm-crypt/LUKS を使って暗号化できます。それを単に以下のようにフォーマットします。

```
# fdisk /dev/sdx
... "n" "p" "1" "return" "return" "w"
# cryptsetup luksFormat /dev/sdx1
...
# cryptsetup open /dev/sdx1 secret
...
# ls -l /dev/mapper/
total 0
crw-rw---- 1 root root  10, 60 2021-10-04 18:44 control
```

```
lrwxrwxrwx 1 root root       7 2021-10-04 23:55 secret -> ../dm-0
# mkfs.vfat /dev/mapper/secret
...
# cryptsetup close secret
```

こうすると暗号化されたディスクは、現代的な GNOME のようなデスクトップ環境では gnome-mount(1) を使ってパスワードを聞く以外は普通のディスクと同様に"`/media/disk_label`" にマウントできます (項 10.1.7 を参照下さい)。全て書込まれるデーターが暗号化されている点が相違点です。パスワード入力はキーリングを使うことで自動化できます (項 10.3.6 を参照下さい)。

異なるファイルシステム、例えば、"`mkfs.ext4 /dev/mapper/sdx1`" として ext4 にメディアをフォーマットするのも一策です。これに替え btrfs が使われた場合には、`udisks2-btrfs` パッケージがインストールされている必要があります。このようなファイルシステムの場合、ファイルのオウナーシップやパーミッションも設定する必要があるかもしれません。

9.9.2 dm-crypt/LUKS で暗号化されたディスクのマウント

例えば、dm-crypt/LUKS を用いて"`/dev/sdc5`" 上に作成された暗号化されたディスクパーティションは以下のようにして"`/mnt`" マウントできます:

```
$ sudo cryptsetup open /dev/sdc5 ninja --type luks
Enter passphrase for /dev/sdc5: ****
$ sudo lvm
lvm> lvscan
  inactive          '/dev/ninja-vg/root' [13.52 GiB] inherit
  inactive          '/dev/ninja-vg/swap_1' [640.00 MiB] inherit
  ACTIVE            '/dev/goofy/root' [180.00 GiB] inherit
  ACTIVE            '/dev/goofy/swap' [9.70 GiB] inherit
lvm> lvchange -a y /dev/ninja-vg/root
lvm> exit
  Exiting.
$ sudo mount /dev/ninja-vg/root /mnt
```

9.10 カーネル

Debian はモジュール化された Linux カーネルをサポートされるアーキテクチャに対してパッケージとしてディストリブートしています。

本ドキュメンテーションを読んでいるなら、あなた自身で Linux カーネルをコンパイルする必要はきっとありません。

9.10.1 カーネル変数

多くの Linux の機能はカーネル変数を使い以下のように設定されます。

- ブートローダーにより初期化されたカーネル変数 (項 3.1.2 を参照下さい)
- 実行時に sysfs によりアクセスできるカーネル変数に関して sysctl(8) を用い変更されたカーネル変数 (項 1.2.12 を参照下さい)
- モジュールがアクティベートされた際の modprobe(8) の引数により設定されるモジュール変数 (項 9.7.3 を参照下さい)

詳細は、"The Linux kernel user's and administrator's guide » The kernel's command-line parameters" を参照下さい。

9.10.2 カーネルヘッダー

ほとんどの普通のプログラムはカーネルヘッダーを必要としませんし、コンパイルするのにそれらを直接用いるとコンパイルがうまくいかないかもしれません。普通のプログラムは Debian システム上では (glibc ソースパッケージから生成される) libc6-dev パッケージが提供する"/usr/include/linux" や"/usr/include/asm" 中のヘッダを使ってコンパイルするべきです。

注意
外部ソースからのカーネルモジュールやオートマウンターデーモン (amd) のようなカーネル固有の一部プログラムをコンパイルする場合、例えば"-I/usr/src/linux-particular-version/include/" 等の対応するカーネルヘッダーへのパスをコマンドラインで指定しなければいけません。

9.10.3 カーネルと関連モジュールのコンパイル

Debian にはカーネルと関連モジュールをコンパイルする独自の方法があります。

パッケージ	ポプコン	サイズ	説明
build-essential	V:20, I:508	17	Debian パッケージをビルドする上で不可欠なパッケージ: make、gcc、…
bzip2	V:163, I:973	113	bz2 ファイルのための圧縮と解凍ユーティリティー
libncurses-dev	V:16, I:148	2380	ncurses のためのデベロッパ用ライブラリーと文書
git	V:442, I:630	52512	git: Linux カーネルによって使われている分散型リビジョンコントロールシステム
fakeroot	V:34, I:508	225	パッケージを非 root としてビルドするための fakeroot 環境を提供
initramfs-tools	V:525, I:989	49	initramfs をビルドするツール (Debian 固有)
dkms	V:81, I:147	235	動的カーネルモジュールサポート (DKMS) (汎用)
module-assistant	V:1, I:11	395	モジュールパッケージ作成用ヘルパーツール (Debian 固有)
devscripts	V:5, I:30	2790	Debian パッケージメンテナ用ヘルパースクリプト (Debian 固有)

Table 9.26: Debian システム上でカーネルの再コンパイルためにインストールする重要パッケージのリスト

項 3.1.2 中で initrd を使う場合、initramfs-tools(7) と update-initramfs(8) と mkinitramfs(8) と initramfs.conf(5) 中の関連情報をしっかり読んで下さい。



警告
Linux カーネルソースをコンパイルする時にソースツリー中のディレクトリー (例えば"/usr/src/linux*") から"/usr/include/linux" や"/usr/include/asm" へのシムリンクを張ってはいけません。(古くなった一部文書はまだこれをすることを提案しています。)

注意
Debian の stable (安定版) システム上で最新の Linux カーネルをコンパイルする際には、Debian の unstable (非安定版) システムからバックポートされた最新のツールが必要かもしれません。
module-assistant(8) (もしくは、その短縮形 m-a) は、単一複数のカスタムカーネル用にモジュールパッケージをユーザーが簡単にビルドするのを援助します。
動的カーネルモジュールサポート (DKMS) は、カーネル全体を変えること無く個別カーネルモジュールをアップグレードできるようにする新しいディストリビューションに依存しない枠組みです。これはアウトオブツリーのモジュールの管理方法です。これはあなたがカーネルをアップグレードする際のモジュールの再構築を簡単にもします。

9.10.4 カーネルソースのコンパイル: Debian カーネルチーム推奨

アップストリームのカーネルソースからカーネルバイナリーパッケージを作成するには、それが提供するターゲットを用いて"deb-pkg" とします。

```
$ sudo apt-get build-dep linux
$ cd /usr/src
$ wget https://mirrors.edge.kernel.org/pub/linux/kernel/v7.x/linux-7.3.5.tar.xz
$ tar --xz -xvf linux-7.3.5.tar.xz
$ cd linux-7.3.5
$ cp /boot/config-7.3.5.config
$ make menuconfig
 ...
$ make deb-pkg
```

ティップ
linux-source-*version* パッケージは Debian パッチがあたった Linux カーネルソースを"/usr/src/linux-*version*.tar.bz2" として提供します。

Debian カーネルソースパッケージから特定のバイナリパッケージをビルドするには、"debian/rules.gen" 中の"binary-arch_*architecture_featureset_flavour*" ターゲットを使います。

```
$ sudo apt-get build-dep linux
$ apt-get source linux
$ cd linux-7.*
$ fakeroot make -f debian/rules.gen binary-arch_i386_none_686
```

詳細は以下参照下さい:

- Debian Wiki: KernelFAQ
- Debian Wiki: Debian カーネル
- Debian Linux カーネルハンドブック: https://kernel-handbook.debian.net

9.10.5 ハードウエアードライバーとファームウエアー

ハードウエアードラーバーとはターゲットシステム上の主 CPU で実行されるコードです。ほとんどのハードウエアードライバーは現在フリーソフトウェアーとして入手可能で main エリアにある普通の Debian カーネルパッケージに含まれています。

- GPU ドラーバー

- Intel GPU ドラーバー (`main`)
- AMD/ATI GPU ドラーバー (`main`)
- NVIDIA GPU ドライバー (nouveau ドライバーは `main`、ベンダーにサポートされたバイナリーのみ提供のドライバーは `non-free` 。)

ファームウエアーとはターゲットシステムに接続されたデバイスにロードされるコードやデーター (例えば CPU マイクロコードや、GPU 上で実行されるレンダリングコードや、FPGA / CPLD データー等々) です。一部のファームウエアーパッケージはフリーソフトウェアーとして入手可能ですが、多くのファームウエアーパッケージはソースの無いバイナリーデーターを含むためにフリーソフトウェアーとして入手不可能です。このようなファームウエアーデーターをインストールすることはデバイスが期待通り動作するのに不可欠です。

- ターゲットデバイス上の揮発性メモリーにロードされるデーターを含むファームウエアーデーターパッケージ。
  - `firmware-linux-free` (`main`)
  - `firmware-linux-nonfree` (`non-free-firmware`)
  - `firmware-linux-*` (`non-free-firmware`)
  - `*-firmware` (`non-free-firmware`)
  - `intel-microcode` (`non-free-firmware`)
  - `amd64-microcode` (`non-free-firmware`)
- ターゲットデバイス上の不揮発性メモリー上のデーターを更新するファームウエアー更新プログラムパッケージ。
  - fwupd (`main`): ファームウエアーデーターを Linux Vendor ファームウエアーサービスからダウンロードする、ファームウエアー更新デーモン
  - gnome-firmware (`main`): fwupd 用の GTK フロントエンド
  - plasma-discover-backend-fwupd (`main`): fwupd 用の Qt フロントエンド

Debian 12 Bookworm 以降、ユーザーに機能的なインストール経験を提供するため `non-free-firmware` パッケージへのアクセスが正規版インストールメディアで提供されていることを承知下さい。`non-free-firmware` エリアに関しては項 2.1.5 を参照下さい。

Linux Vendor ファームウエアーサービスから fwupd がダウンロードし実行中の Linux カーネルにロードするファームウエアーデーターは `non-free` かもしれないことも承知下さい。

9.11 仮想化システム

仮想化されたシステムを利用すると単一ハード上で同時に複数のシステムのインスタンスを実行することが可能となります。

ティップ
Debian wiki: SystemVirtualization を参照下さい。

9.11.1 仮想化やエミュレーションツール

仮想化とエミュレーションツールはいくつかあります。

- games-emulator メタパッケージがインストールするような、完璧なハードウエアーエミュレーションパッケージ
- QEMU のような一部の I/O デバイスエミュレーションを含む、ほぼ CPU レベルのエミュレーション

- Kernel-based Virtual Machine (KVM) のような一部の I/O デバイスエミュレーションを含む、ほぼ CPU レベルの仮想化
- LXC (Linux コンテナー) や Docker や systemd-nspawn(1) ... 等のようなカーネルレベルのサポートの下での OS レベルの仮想化
- chroot のようなシステムライブラリーコールがファイルパスをオーバーライドすることによる OS レベルのファイルシステムアクセス仮想化
- fakeroot のようなシステムライブラリーコールがファイルオウナーシップをオーバーライドすることによる OS レベルのファイルシステムアクセス仮想化
- Wine のような OS API のエミュレーション
- Python 用の virtualenv や venv のようなインタープリターの実行選択や実行時ライブラリーをオーバーライドすることによるインタープリターレベルの仮想化

コンテナ仮想化は項 4.7.5 を使い、また項 7.7 のバックエンド技術です。

仮想化システムを設定する際に役立ついくつかのパッケージを記します。

異なるプラットフォーム仮想化策の詳細な比較は Wikipedia の記事 Comparison of platform virtual machines を参照下さい。

9.11.2 仮想化の業務フロー

注意
lenny 以降の Debian のデフォルトカーネルは KVM をサポートしています。

仮想化のための典型的な業務フローにはいくつかの段階があります。

- 空のファイルシステムの作成 (ファイルツリーもしくはディスクイメージ)。
  - ファイルツリーは”`mkdir -p /path/to/chroot`” として作成できる。
  - raw ディスクイメージファイルは dd(1) を使って作れます (項 9.7.1 と項 9.7.5 を参照下さい)。
  - qemu-img(1) は QEMU によりサポートされたディスクイメージの作成や変換に使えます。
  - raw と VMDK ファイルフォーマットは仮想ツール間の共通フォーマットとして使えます。
- mount(8) を使ってディスクイメージをファイルシステムにマウントする (任意)。
  - raw のディスクイメージファイルに関しては、loop デバイスまたはデバイスマッパーデバイス (項 9.7.3 を参照下さい) としてマウント。
  - QEMU がサポートするディスクイメージファイルに関しては、ネットワークブロックデバイス (項 9.11.3 を参照下さい) としてマウント。
- 必要なシステムデーターを用いて対象のファイルシステムを充足。
  - `debootstrap` や `cdebootstrap` のようなプログラムがこのプロセスを援助します (項 9.11.4 を参照下さい)。
  - OS のインストーラーをフルシステムエミュレーション下で利用。
- 仮想化環境下でプログラムを実行。
  - chroot は、仮想環境の中でプログラムのコンパイルやコンソールアプリケーションの実行やデーモンの実行等をするのに十分な基本的仮想環境を提供します。
  - QEMU: クロスプラットフォームの CPU エミュレーションを提供
  - KVM と共の QEMU はハードウエア補助仮想化によるフルシステムエミュレーションを提供します。
  - VirtualBox はハードウエア補助仮想化の有無によらず i386 と amd64 上でのフルシステムエミュレーションを提供します。

パッケージ	ポプコン	サイズ	説明
coreutils	V:914, I:1000	19751	chroot(8) を含む GNU core utilities
util-linux	V:922, I:1000	4826	chroot(8) を含む GNU core utilitiesunshare(1) を含む雑多なシステムユーティリティー
systemd-container	V:74, I:77	2964	systemd-nspawn(1) を含む systemd の container/nspawn ツール
schroot	V:7.1, I:8.9	2241	Debian バイナリーパッケージを chroot 中で実行する専用ツール
sbuild	V:3.8, I:6.8	160	Debian ソースから Debian バイナリーパッケージをビルドするツール
debootstrap	V:4, I:43	332	基本的な Debian システムのブートストラップ (sh で書かれている)
mmdebstrap	V:8, I:12	613	Debian システムのブートストラップ (Perl で書かれている)
cdebootstrap	V:0.1, I:1.2	113	Debian システムのブートストラップ (C で書かれている)
cloud-image-utils	V:1, I:14	52	クラウドイメージ管理ユーティリティー
cloud-guest-utils	V:5, I:25	67	クラウドゲストユーティリティー
virt-manager	V:12, I:47	2309	仮想マシンマネージャー: 仮想マシンを管理するデスクトップアプリケーション
libvirt-clients	V:49, I:70	1168	libvirt ライブラリー用のプログラム
docker.io	V:51, I:55	80705	docker: Linux コンテナーランタイム
podman	V:36, I:40	86285	podman: OCI ベースのコンテナーを Pod 中で実行するエンジン
podman-docker	V:2.9, I:3.4	271	OCI-ベースのコンテナを Pod 中で実行するエンジン - docker 用のラッパー
incus	V:1.0, I:3.7	24	Incus: システムコンテナーと仮想マシンマネージャー
games-emulator	I:0.18	21	games-emulator: Debian のゲーム用エミュレーター
bochs	V:0.03, I:0.53	8469	Bochs: IA-32 PC エミュレーター
qemu-system	I:19	69	QEMU: フルシステムエミュレーションのバイナリ
qemu-user	V:6.3, I:9.8	472221	QEMU: ユーザーモードエミュレーションのバイナリ
qemu-utils	V:16, I:117	12651	QEMU: ユーティリティー
qemu-system-x86	V:55, I:101	69265	KVM: ハードウエア補助仮想化を利用する x86 ハードウエア上のフル仮想化
virt-manager	V:12, I:47	2309	KVM 仮想マシン、Xen、LXC、... 等の仮想マシン管理のためのデスクトップアプリケーション
gnome-boxes	V:1.1, I:5.8	7070	Boxes: 仮想システムにアクセスするシンプルな GNOME アプリ
xen-tools	V:0.1, I:1.3	719	Debian XEN 仮想サーバーの管理ツール
wine	V:13, I:54	209	Wine: Windows API の実装 (標準スイート)
dosbox	V:1, I:11	3139	DOSBox: Tandy/Herc/CGA/EGA/VGA/SVGA グラフィクス、サウンド、DOS 付きの x86 エミュレーター
lxc	V:11, I:13	1680	Linux コンテナーユーザースペースツール
python3-venv	V:8, I:167	6	仮想 python 環境を作るための venv (システムライブラリー)
python3-virtualenv	V:7, I:35	435	隔離された仮想 python 環境を作るための virtualenv
pipx	V:8, I:52	3261	隔離された環境に python アプリをインストールするための pipx

Table 9.27: 仮想化ツールのリスト

9.11.3 仮想ディスクイメージファイルをマウント。

raw ディスクイメージファイルに関しては、項 9.7 を参照下さい。

他の仮想ディスクイメージに関しては、qemu-nbd(1) を使ってネットワークブロックデバイスプロトコルを用いてそれらをエクスポートし nbd カーネルモジュールを使ってそれらをマウントできます。

qemu-nbd(1) は QEMU がサポートする以下のディスクフォーマットをサポートします: raw、qcow2、qcow、vmdk、vdi、bochs、cow (user-mode Linux の copy-on-write)、parallels、dmg、cloop、vpc、vvfat (virtual VFAT)、host_device。

ネットワークブロックデバイスは loop デバイスと同様の方法でパーティションをサポートします (項 9.7.3 を参照下さい)。"`image.img`" の最初のパーティションは以下のようにするとマウントできます。

```
# modprobe nbd max_part=16
# qemu-nbd -v -c /dev/nbd0 disk.img
...
# mkdir /mnt/part1
# mount /dev/nbd0p1 /mnt/part1
```

ティップ
qemu-nbd(8) に"`-P 1`" オプションを用いると、"`disk.img`" の最初のパーティションだけをエクスポートできます。

9.11.4 Chroot システム

もしターミナルコンソールから新規 Debian 環境を試したい場合、chroot を使うことをお薦めします。これを使うと、ありがちな関連するリスク無しかつブート無しに Debian の `unstable` や `testing` のアプリを実行できます。chroot(8) は最も基本的手法です。



注意
以下の例は親システムと chroot システムが同じ `amd64` CPU アーキテクチャを共有していると仮定しています。

---

debootstrap(8) を使うと chroot(8) 環境を手動で作れますが、少々手間です。

Debian パッケージをソースからビルドする sbuild パッケージは schroot によって管理された chroot 環境を使います。それには sbuild-createchroot(8) という補助スクリプトが同梱されています。それを以下のように実行し、どのように動作するのかを学びましょう。

```
$ sudo mkdir -p /srv/chroot
$ sudo sbuild-createchroot -v --include=eatmydata,ccache unstable /srv/chroot/unstable- ↩
    amd64-sbuild http://deb.debian.org/debian
 ...
```

"`/srv/chroot/unstable-amd64-sbuild`" の下に `unstable` 環境のためのシステムデーターをどのようにして充足するかは debootstrap(8) を見ると分かります。

schroot(1) を使ってこの環境に login できます。

```
$ sudo schroot -v -c chroot:unstable-amd64-sbuild
```

`unstable` 環境下で実行されるシステムシェルをどのようにして作成するかが理解できます。

注意
常に 101 で終了する”/usr/sbin/policy-rc.d” ファイルは、Debian システム上でデーモンプログラムが自動的に起動されることを防ぎます。”/usr/share/doc/init-system-helpers/README.policy-rc.d.gz” を参照下さい。

注意
プログラムによっては機能するために chroot の下で sbuild-createchroot が提供するより多くの親システムのファイルへのアクセスする必要があります。例えば、”/sys” や”/etc/passwd” や”/etc/group” や”/var/run/utmp” や”/var/log/wtmp” 等が bind マウントもしくはコピーされる必要があるかもしれません。

ティップ
sbuild パッケージはそのバックエンドに schroot を使って chroot システムを構築し chroot 内でパッケージをビルドします。それはビルド依存を確認するのに理想的です。詳細は Debian wiki の sbuild や”Guide for Debian Maintainers” 中の sbuild 設定例を参照下さい。

ティップ
systemd-nspawn(1) コマンドは chroot に似た方法で軽量コンテナ中でコマンドや OS を実行したりするのを援助します。それは、namespaces を使ってプロセス木や IPC やホスト名やドメイン名やさらにはネットワーキングやユーザーデーターベースまで完全に仮想化するので、それはより強力です。systemd-nspawn を参照下さい。

9.11.5 複数のデスクトップシステム

仮想化を使って複数のデスクトップシステムを安全に実行するには、Debian 安定版 (stable) システム上で QEMU か KVM を使うことをお薦めします。これらを使うと通常ありがちなリスクに晒されずにまたリブートすること無く Debian テスト版 (testing) や不安定版 (unstable) システムのデスクトップアプリケーションを実行できるようになります。

純粋な QEMU は非常に遅いので、ホストシステムがサポートする際には KVM を使って加速することをお薦めします。

仮想マシンマネージャーは、virt-manager とも呼ばれていて、libvirt 経由で KVM 仮想マシンを管理する便利な GUI ツールです。

QEMU 用の Debian システムを含む仮想ディスクイメージ”virtdisk.qcow2” は debian-installer: 小さな CD を使って以下のように作成できます。

```
$ wget https://cdimage.debian.org/debian-cd/5.0.3/amd64/iso-cd/debian-503-amd64-netinst.iso
$ qemu-img create -f qcow2 virtdisk.qcow2 5G
$ qemu -hda virtdisk.qcow2 -cdrom debian-503-amd64-netinst.iso -boot d -m 256
...
```

ティップ
Ubuntu や Fedra 等の GNU/Linux ディストリビューションを仮想化の下で実行するのは設定ティップを学ぶ非常に良い方法です。他のプロプライエタリな OS もこの GNU/Linux の仮想化の下で上手く実行できます。

更なるティップに関しては Debian wiki: SystemVirtualization を参照下さい。

Chapter 10

データー管理

バイナリーとテキストのデーターを Debian システム上で管理するツールとティップを記します。

10.1 共有とコピーとアーカイブ



警告
競合状態とならないようにするために、アクティブにアクセスされているデバイスやファイルに複数プロセスから調整なく書き込みアクセスをしてはいけません。flock(1) を使ったファイルロック機構がこの回避に使えます。

データーのセキュリティーとそのコントロールされた共有はいくつかの側面があります。

- データーアーカイブの作成
- 遠隔ストレージアクセス
- 複製
- 変更履歴の追跡
- データー共有のアシスト
- 不正なファイルへのアクセスの防止
- 不正なファイルの改変の検出

こういったことは以下の組み合わせを使うことで実現できます。

- アーカイブと圧縮ツール
- コピーと同期ツール
- ネットワークファイルシステム
- リムーバブルストレージメディア
- セキュアーシェル
- 認証システム
- バージョンコントロールシステムツール
- ハッシュや暗号学的暗号化ツール

10.1.1 アーカイブと圧縮ツール

Debian システム上で利用可能なアーカイブと圧縮ツールのまとめを以下に記します。

パッケージ	ポプコン	サイズ	拡張子	コマンド	コメント
tar	V:911, I:1000	3101	.tar	tar(1)	標準アーカイバー (デファクト標準)
cpio	V:388, I:999	1205	.cpio	cpio(1)	Unix System V スタイルのアーカイバー、find(1) とともに使用
binutils	V:201, I:627	1132	.ar	ar(1)	静的ライブラリー生成用のアーカイバー
fastjar	V:0.9, I:9.0	182	.jar	fastjar(1)	Java 用のアーカイバー (zip 類似)
pax	V:4.9, I:8.5	167	.pax	pax(1)	新規 POSIX 標準アーカイバー、tar と cpio の間の妥協点
gzip	V:904, I:1000	262	.gz	gzip(1), zcat(1), …	GNU LZ77 圧縮ユーティリティー (デファクト標準)
bzip2	V:163, I:973	113	.bz2	bzip2(1), bzcat(1), …	gzip(1) より高い圧縮比 (gzip より遅い、類似シンタックス) の Burrows-Wheeler ブロック並び替え圧縮ユーティリティー
lzma	V:0.7, I:9.4	349	.lzma	lzma(1)	gzip(1) より高い圧縮比の LZMA 圧縮ユーティリティー (非推奨)
xz-utils	V:323, I:981	1551	.xz	xz(1), xzdec(1), …	bzip2(1) より高い圧縮比の XZ 圧縮ユーティリティー (gzip より遅いが bzip2 より早い、LZMA 圧縮ユーティリティーの代替)
zstd	V:346, I:850	2308	.zstd	zstd(1), zstdcat(1), …	Zstandard 高速ロスレス圧縮ユーティリティー
7zip	V:86, I:339	7542	.7z	7zr(1), p7zip(1)	高い圧縮比をもつ 7-Zip 圧縮ユーティリティー (LZMA 圧縮、他)
lzop	V:13, I:128	164	.lzo	lzop(1)	gzip(1) より高い圧縮と解凍の速度 (gzip より低い圧縮比、類似シンタックス) の LZO 圧縮ユーティリティー
zip	V:52, I:350	635	.zip	zip(1)	InfoZIP: DOS アーカイブと圧縮ツール
unzip	V:126, I:738	399	.zip	unzip(1)	InfoZIP: DOS アーカイブ解凍と圧縮解凍ツール

Table 10.1: アーカイブと圧縮ツールのリスト



警告
何が起こるかを理解せずに”$TAPE” 変数を設定してはいけません。設定すると tar(1) の挙動が変わります。

- gzip 圧縮された tar(1) アーカイブは”.tgz” とか”.tar.gz” といったファイル拡張子を使います。
- xz 圧縮された tar(1) アーカイブは”.txz” とか”.tar.xz” といったファイル拡張子を使います。
- tar(1) 等の FOSS ツールでのポピュラーな圧縮方法は以下のように変遷しています: gzip → bzip2 → xz

- cp(1) と scp(1) と tar(1) は特殊ファイルに関して一部制約があるかもしれません。cpio(1) は最も汎用性があります。
- cpio(1) は find(1) 等のコマンドとともに使うようにできていて、ファイルの選定部分のスクリプトを独立にテストできるのでバックアップスクリプトを作るのに向いています。
- Libreoffice データーファイルの内部構造は”.jar” ファイルで、unzip で開くことができます。
- デファクトのクロスプラットフォームのアーカイブツールは zip です。最大限のコンパチビリティーのためには”zip -rX” として使って下さい。もし最大ファイルサイズが問題となる際には”-s” オプションも使って下さい。

10.1.2 コピーと同期ツール

Debian システム上で利用可能な単純なコピーとバックアップツールのまとめを以下に記します。

パッケージ	ポプコン	サイズ	ツール	機能
coreutils	V:914, I:1000	19751	GNU cp	ファイルやディレクトリーのローカルコピー (”-a” で再帰的実行)
openssh-client	V:709, I:997	3611	scp	ファイルやディレクトリーのリモートコピー (クライアント、”-r” で再帰実行)
openssh-server	V:781, I:824	3615	sshd	ファイルやディレクトリーのリモートコピー (リモートサーバー)
rsync	V:191, I:538	985		単方向リモート同期とバックアップ
unison	V:3, I:12	14		双方向リモート同期とバックアップ

Table 10.2: コピーと同期ツールのリスト

rsync(1) を使ってのファイルのコピーには他の方法より豊かな機能があります。

- 転送元のファイルと転送先の既存ファイル間の相違のみを送信する差分転送アルゴリズム
- サイズか最終変更時間に変更があったファイルのみを探す (デフォルトで採用される) 急速確認アルゴリズム
- tar(1) 類似の”--exclude” や”--exclude-from” オプション
- 転送先に追加ディレクトリーレベルを作成しなくする「転送元ディレクトリ後スラシュ (/) 付加」文法

ティップ
表 10.14 に記されたバージョンコントロールシステム (VCS) ツールは多方向のコピーと同期のツールとして機能します。

10.1.3 アーカイブの慣用句

”./source” ディレクトリー中の全内容を異なるツールを用いてアーカイブしアーカイブ解凍するいくつかの方法を以下に記します。

GNU tar(1):

```
$ tar -cvJf archive.tar.xz ./source
$ tar -xvJf archive.tar.xz
```

この代わりに、以下のようにも出来ます。

```
$ find ./source -xdev -print0 | tar -cvJf archive.tar.xz --null -T -
```

cpio(1):

```
$ find ./source -xdev -print0 | cpio -ov --null > archive.cpio; xz archive.cpio
$ zcat archive.cpio.xz | cpio -i
```

10.1.4 コピーの慣用句

"./source" ディレクトリー中の全内容を異なるツールを用いてコピーするいくつかの方法を以下に記します。

- ローカルコピー: "./source" ディレクトリー → "/dest" ディレクトリー
- リモートコピー: ローカルホストの"./source" ディレクトリー → "user@host.dom" ホストの"/dest" ディレクトリー

rsync(8):

```
# cd ./source; rsync -aHAXSv . /dest
# cd ./source; rsync -aHAXSv . user@host.dom:/dest
```

「転送元ディレクトリー後スラシュ付加」文法を上記の代わりに使えます。

```
# rsync -aHAXSv ./source/ /dest
# rsync -aHAXSv ./source/ user@host.dom:/dest
```

この代わりに、以下のようにも出来ます。

```
# cd ./source; find . -print0 | rsync -aHAXSv0 --files-from=- . /dest
# cd ./source; find . -print0 | rsync -aHAXSv0 --files-from=- . user@host.dom:/dest
```

GNU cp(1) と openSSH scp(1):

```
# cd ./source; cp -a . /dest
# cd ./source; scp -pr . user@host.dom:/dest
```

GNU tar(1):

```
# (cd ./source && tar cf - . ) | (cd /dest && tar xvfp - )
# (cd ./source && tar cf - . ) | ssh user@host.dom '(cd /dest && tar xvfp - )'
```

cpio(1):

```
# cd ./source; find . -print0 | cpio -pvdm --null --sparse /dest
```

"." を含むすべての例で"." は"foo" で代替でき、ファイルを"./source/foo" ディレクトリーから"/dest/foo" ディレクトリにコピーできます。

"." を含むすべての例で"." を絶対パスの"/path/to/source/foo" で代替でき、"cd ./source;" を削除することができます。これらは使うツール次第で異なる場所にファイルをコピーします。

- "/dest/foo": rsync(8)、GNU cp(1)、scp(1)
- "/dest/path/to/source/foo": GNU tar(1) と cpio(1)

ティップ
rsync(8) や GNU cp(1) には転送先のファイルが新しい場合にスキップする"-u" オプションがあります。

10.1.5 ファイル選択の慣用句

アーカイブやコピーコマンド (項 10.1.3 と項 10.1.4 を参照下さい) のためや xargs(1) (項 9.4.9 を参照下さい) のためにファイルを選択するのに find(1) が使われます。これの操作はそのコマンド引数を使うことで強化できます。

find(1) の基本シンタックスは以下のようにまとめられます。

- 条件の引数は左から右へと評価されます。
- 結果が決まった時点で評価は終了します。
- ” 論理 **OR**” (条件間に”`-o`” で指定) は、” 論理 **AND**” (条件間に”`-a`” または何もなしで指定) より低い優先順位です。
- ” 論理 **NOT**” (条件前に”`!`” で指定) は、” 論理 **AND**” より高い優先順位です。
- ”`-prune`” は常に論理真 **(TRUE)** を返し、ディレクトリーの場合にはこの点以降のファイル探索を停止します。
- ”`-name`” はシェルのグロブ (項 1.5.6 を参照下さい) を使ってファイル名のベースにマッチし、さらに”*” and ”?” 等のメタ文字で最初の”`.`” ともマッチします。(新規の POSIX 機能)
- ”`-regex`” はデフォールトでは emacs スタイルの **BRE** (項 1.6.2 を参照下さい) を用いてフルパスをマッチします。
- ”`-size`” はファイルサイズ (”+” が前に付いた値はより大きい、”-” が前に付いた値はより小さい) に基づいてファイルをマッチします。
- ”`-newer`” はその引数に指定されたファイルより新しいファイルとマッチします。
- ”`-print0`” は常に論理真 **(TRUE)** を返し、フルファイル名を (null 終端処理して) 標準出力へプリントします。

find(1) はしばしば慣用的なスタイルで使われます。

```
# find /path/to \
    -xdev -regextype posix-extended \
    -type f -regex ".*\.cpio|.*~" -prune -o \
    -type d -regex ".*/\.git" -prune -o \
    -type f -size +99M -prune -o \
    -type f -newer /path/to/timestamp -print0
```

これは以下のアクションをすることを意味します。

1. ”`/path/to`” からはじまる全ファイルを探索
2. 探索開始したファイルシステムに探索を全体的に制約し、デフォールトの代わりに **ERE** (項 1.6.2 を参照下さい) を使用
3. 正規表現”`.*\.cpio`” か”`.*~`” にマッチするファイルを処理停止をすることで探索から除外
4. 正規表現”`.*/\.git`” にマッチするディレクトリーを処理停止をすることで探索から除外
5. 9MiB(1048576 バイトの単位) より大きいファイルを処理停止をすることで探索から除外
6. 上記の探索条件に合致し”`/path/to/timestamp`” より新しいファイル名をプリントします

上記例中でファイルを検索から除外するときの”`-prune -o`” の慣用的な使い方を承知下さい。

注意
非 Debian の Unix 的システムでは、一部のオプションは find(1) によってサポートされていないかもしれません。そのような場合には、マッチング方法を調整したり”`-print0`” を”`-print`” で置き換えることを考慮します。これに関連するコマンドも調整する必要があるかもしれません。

10.1.6 アーカイブメディア

重要なデーターアーカイブのためのコンピューターデーターストレージメディアを選ぶ時にはそれらの限界について注意を払うべきです。小さな個人的なバックアップのためには、著者としては名前が知られている会社の CD-R と DVD-R を使い、クールで日陰の乾燥した埃の無い環境に保存しています。(プロ用途ではテープアーカイブメディアに人気があるようです。)

注意
耐火金庫は紙の文書のためのものです。ほとんどのコンピューターデーターストレージメディアは紙よりも耐熱性がありません。著者は複数の安全な場所に保管された複数のセキュアーな暗号化されたコピーに通常頼っています。

ネット上に散見するアーカイブメディアの楽観的なストレージ寿命 (ほとんどベンダー情報由来)。

- 100+ 年: インクと中性紙
- 100 年: オプティカルストレージ (CD/DVD、CD/DVD-R)
- 30 年: 磁気ストレージ (テープ、フロッピー)
- 20 年: 相変化オプティカルストレージ (CD-RW)

これらは取扱いによる機械的故障等は考慮していません。

ネット上に散見するアーカイブメディアの楽観的な書込み回数 (ほとんどベンダー情報由来)。

- 250,000+ 回: ハードディスク
- 10,000+ 回: フラッシュメモリー
- 1,000 回: CD/DVD-RW
- 1 回: CD/DVD-R、紙



注意
ここにあるストレージ寿命や書込み回数の数字はクリチカルなデーターストレージに関する決定に使うべきではありません。製造者によって提供される特定の製品情報を参照下さい。

ティップ
CD/DVD-R や紙は 1 回しか書けないので、本質的に重ね書きで間違ってデーターを消すことを防げます。これは、利点です!

ティップ
もし高速で頻繁な大量のデーターのバックアップをする必要がある場合、高速のネットワーク接続でつながっているリモートホスト上のハードディスクが唯一の現実的なオプションかもしれません。

ティップ
もし書き換え可能なメディアをバックアップに使っている場合には、リードオンリーのスナップショットをサポートする btrfs や zfs のようなファイルシステムを使うのも一策です。

10.1.7 リムーバブルストレージデバイス

リムーバブルストレージデバイスは以下の何れも指します。

- USB フラッシュドライブ
- Hard ディスクドライブ
- 光学ディスクドライブ
- デジタルカメラ
- デジタル音楽プレーヤー

これらは以下の何れかで接続できます。

- USB
- IEEE 1394 / FireWire

GNOME や KDE のような最近のデスクトップ環境は、”/etc/fstab” エントリーにマッチが無いリムーバブルデバイスを自動的にマウントする事ができます。

- udisks2 パッケージは、これらのデバイスをマウントやアンマウントするためのデーモンと関連するユーティリティーを提供します。
- D-bus は、自動的なプロセスを開始するイベントを作成します。
- PolicyKit が必要な特権を提供します。

ティップ
自動的にマウントされたデバイスは、umount(8) によって利用される”uhelper=” マウントオプションが設定されているかもしれません。

ティップ
”/etc/fstab” にリムーバブルメディアデバイスの記載が無い時のみ、現代的なデスクトップ環境下での自動マウントは起こります。

最新のデスクトップ環境下では以下のようにしてカスタマイズ可能なマウント点として”/media/*username/disk_labe*
が選ばれます。

- FAT ファイルシステムでは、mlabel(1) を使います。
- ISO9660 ファイルシステムでは、genisoimage(1) を”-V” オプションとともに使います。
- ext2/ext3/ext4 ファイルシステムでは、tune2fs(8) を”-L” オプションとともに使います。

ティップ
符号化方式 (エンコーディング) の選択をマウントオプションとして与える必要があるかもしれません (項 8.1.3 を参照下さい)。

ティップ
ファイルシステムをアンマウントする際に GUI メニューを使うと、動的に生成された”/dev/sdc” 等のデバイスノード削除するかもしれません。もしそのデバイスノードの削除したくない場合にはシェルのコマンドプロンプトから umount(8) コマンドを使いアンマウントします。

10.1.8 データー共有用のファイルシステム選択

リムーバブルストレージデバイスを使ってデーターを共有する際には、両方のシステムにサポートされた共通のファイルシステムでそれをフォーマットするべきです。ファイルシステム選択のリストを次に示します。

注意
ハードディスク (HDD) に関する文言は、SSD / USB フラッシュデバイス / メモリーカード /... のようなストレージデバイスのような他のストレージデバイスにも適用できます。例中の`/dev/sda` のようなデバイス名は、該当するデバイス名 `/dev/nvme0`, `/dev/mmcblk0`, ... に置き換えます。

ファイルシステム名	典型的使用シナリオ
FAT12	フロッピーディスク上のクロスプラットフォームのデーター共有 (<32MiB)
FAT16	小さなハードディスク類似のデバイス上のクロスプラットフォームのデーター共有 (<2GiB)
FAT32	大きなハードディスク類似のデバイス上のクロスプラットフォームのデーター共有 (<8TiB, MS Windows95 OSR2 以降でサポート有り)
exFAT	大きなハードディスク類似のデバイス上のクロスプラットフォームのデーター共有 (<512TiB, WindowsXP と Mac OS X Snow Leopard 10.6.5 と Linux kernel 5.4 リリース以降でサポート有り)
NTFS	大きなハードディスク類似のデバイス上のクロスプラットフォームのデーター共有 (MS Windows NT 以降でネイティブにサポート、Linux 上では FUSE 経由の NTFS-3G でサポート)
ISO9660	CD-R and DVD+/-R 上の静的データーのクロスプラットフォームの共有
UDF	CD-R や DVD+/-R 上への増分データーの書込み (新規)
MINIX	フロッピーディスク上へのスペース効率の良い unix ファイルデーターのストレージ
ext2	古い Linux システムとハードディスク類似デバイス上のデーターを共有
ext3	古い Linux システムとハードディスク類似デバイス上のデーターを共有
ext4	現行の Linux システムとハードディスク類似デバイス上のデーターを共有
btrfs	現行の Linux システムとハードディスク類似デバイス上のデーターを読み出し専用のスナップショットでの共有

Table 10.3: 典型的な使用シナリオに合わせたリムーバブルストレージデバイスのファイルシステムの選択肢のリスト

ティップ
デバイスレベルの暗号化を使ったクロスプラットフォームのデーター共有に関しては、項 9.9.1 を参照下さい。

FAT ファイルシステムはほとんど全ての現代的なオペレーティングシステムでサポートされていて、ハードディスク類似のメディア経由でのデーター交換目的に非常に有用です。

クロスプラットフォームの FAT ファイルシステムを使ったデーター共有にリムーバブルハードディスク類似デバイスをフォーマットする時の安全な選択肢は次です。

- fdisk(8) か cfdisk(8) か parted(8) (項 9.6.2 を参照下さい) を使ってそれを単一のプライマリパーティションにパーティションしそれを以下のようにマークします。
  - 2GB より小さなメディアには FAT16 となるように”6” とタイプします
  - 大きなメディアには FAT32 (LBA) となるように”c” とタイプします
- 第 1 パーティションを mkfs.vfat(8) を使って以下のようにフォーマットします。
  - FAT16 となるように”`/dev/sda1`” 等とそのデバイス名だけを使います

– FAT32 となるように”`-F 32 /dev/sda1`” 等と明示的なオプション指定とそのデバイス名を使います

FAT とか ISO9660 ファイルシステムを使ってデーターを共有する際の安全への配慮を次に記します。

- tar(1) や cpio(1) を使ってアーカイブファイルに最初にファイルをアーカイブすることで長いファイル名やシンボリックリンクやオリジナルの Unix ファイルパーミッションとオーナー情報を保持します。
- split(1) コマンドを使ってアーカイブファイルを 2GiB 以下の塊に分割してファイルサイズの制約から保護します。
- アーカイブファイルを暗号化してその内容を不正アクセスから保護します。

注意
FAT ファイルシステムはその設計上最大ファイルサイズは `(2^32 - 1) bytes = (4GiB - 1 byte)` です。古い 32 ビット OS 上の一部アプリケーションは、最大ファイルサイズはさらに小さく `(2^31 - 1) bytes = (2GiB - 1 byte)` です。Debian は後者の問題に苦しむことはありません。

注意
Microsoft 自身も 200MB を越すドライブやパーティションに FAT を使うことを勧めていません。マイクロソフトは、彼らの”Overview of FAT, HPFS, and NTFS File Systems” で非効率的なディスク領域の使用等の欠点をハイライトしています。もちろん私たちは Linux では ext4 ファイルシステムを普通使うべきです。

ティップ
ファイルシステムとファイルシステムのアクセスに関しての詳細は、”Filesystems HOWTO” を参照下さい。

10.1.9 ネットワーク経由でのデーター共有

データーをネットワーク経由で他のシステムと共有するときには、共通のサービスを使うべきです。次に一部のヒントを記します。

ネットワークサービス	典型的使用シナリオの説明
Samba を使う SMB/CIFS ネットワーク経由マウントファイルシステム	”Microsoft Windows Network” 経由でのファイル共有、smb.conf(5) と The Official Samba 3.x.x HOWTO and Reference Guide を参照下さい
Linux カーネルを使う NFS ネットワークマウントファイルシステム	”Unix/Linux Network” 経由のファイル共有、exports(5) と Linux NFS-HOWTO 参照下さい。
HTTP サービス	ウェッブサーバー/クライアント間のファイル共有
HTTPS サービス	暗号化されたセキュアーソケットレイヤー (SSL) もしくは Transport Layer Security (TLS) を使ったウェッブサーバー/クライアント間のファイル共有
FTP サービス	FTP サーバー/クライアント間のファイル共有

Table 10.4: 典型的使用シナリオの場合のネットワークサービスの選択のリスト

このようなネットワーク経由でマウントされたファイルシステムやネットワーク経由のファイル転送法はデーター共有のために非常に便利ですが、インセキュアーかもしれませんこれらのネットワーク接続は次に記すようにしてセキュアーにされなければいけません。

- SSL/TLS を使い暗号化
- SSH 経由でそれをトンネル
- VPN 経由でそれをトンネル
- セキュアーファイアウォールの背後に限定

さらに項 6.5 と項 6.6 を参照下さい。

10.2 バックアップと復元

コンピューターはいつか壊れるとか、人間によるエラーがシステムやデーターをへの損害を及ぼすことは皆知っています。バックアップと復元の操作は正しいシステム管理の必須構成要素です。考えうる全ての故障モードはいつかの日にやって来ます。

ティップ
バックアップのシステムは簡単にしておき、システムのバックアップは頻繁にします。バックアップデーターが存在することは、あなたのバックアップ方法が技術的に如何に良いかよりも重要です。

10.2.1 バックアップと復元のポリシー

実際のバックアップと復元の方針を決める上で 3 つの要素があります。

1. 何をバックアップし復元するかを知っていること
   - あなた自身が作成したデーターファイル: "~/" 中のデーター
   - あなた自身が使用したアプリケーションが作成したデーターファイル: "/var/" ("/var/cache/" と"/var/run/" と"/var/tmp/" は除外) 中のデーター
   - システム設定ファイル: "/etc/" 中のデーター
   - ローカルプログラム: "/usr/local/" とか"/opt/" 中のデーター
   - システムインストール情報: 要点 (パーティション、…) をプレーンテキストで書いたメモ
   - 実証済みのデーターセット: 事前の実験復元操作で確認
     - ユーザープロセスでの cron ジョブ: "/var/spool/cron/crontabs" ディレクトリー中のファイルと cron(8) の再スタート。cron(8) と crontab(1) に関しては項 9.4.14 を参照下さい。
     - ユーザープロセスでの systemd タイマージョブ: "~/.config/systemd/user ディレクトリー中のファイル。systemd.timer(5) や systemd.service(5) を参照下さい。
     - ユーザープロセスでの自動スタートジョブ: "~/.config/autostart" ディレクトリー中のファイル。Desktop Application Autostart Specification を参照下さい。
2. バックアップと復元の方法を知っていること
   - セキュアーなデーターのストレージ: 上書きやシステム障害の防止
   - 頻繁なバックアップ: スケジュールされたバックアップ
   - 冗長なバックアップ: データーのミラーリング
   - フルプルーフなプロセス: 簡単な単一コマンドバックアップ
3. 関わっているリスクと費用の評価
   - データー消失時のリスク
     - データーはファイルシステム破壊に耐えるように、できれば異なるディスクやマシン上の、最低限異なるディスクパーティション上に置くべきです。重要データーは読み取り専用ファイルシステムに保存するのが好ましい。1
   - データ侵害の際のデーターのリスク
     - "/etc/ssh/ssh_host_*_key" や"~/.gnupg/*" や"~/.ssh/*" や"/etc/passwd" や"/etc/shadow" や"/etc/fetchmailrc" や"popularity-contest.conf" や"/etc/ppp/pap-secrets" や"/etc/exim4/pa 等の慎重に扱うべきアイデンティティ関連のデーターファイルは暗号化してバックアップする必要があります。2 (項 9.9 を参照下さい。)

1重要データーは上書き事故を防ぐために CD/DVD-R のような 1 回書込みメディアに貯蔵するのが好ましいです。(シェルコマンドラインからストレージメディアにどうして書き込むかについては項 9.8 を参照下さい。GNOME デスクトップの GUI 環境ではメニュー: "Places→CD/DVD Creator" で簡単に書込みできます。)

2このようなデーターの一部は、システムに同一の入力文字列を入力しても再生成できません。

  - たとえ信頼できるシステム上でも、システムの login パスワードや暗号化解除パスフレーズは、いかなるスクリプト中にもハードコードしてはいけません。(項 10.3.6 を参照下さい。)
- 故障モードとその確率
  - ハードウエアー (特に HDD) はいずれ壊れます
  - ファイルシステムは壊れるかもしれないし、その中のデーターは失われるかもしれません
  - セキュリティー侵害に関してリモートストレージシステムは信用できません。
  - 脆弱なパスワードによる保護は簡単に破られます
  - ファイルパーミッションシステムが不正アクセスを許すようになるかもしてません
- バックアップに必要なリソース: 人的、ハードウエアー、ソフトウェアー、…
  - cron ジョブや systemd タイマージョブでする自動スケジュールバックアップ

ティップ
debconf の設定データーは”`debconf-set-selections debconf-selections`” で、dpkg の選択データーは”`dpkg --set-selection <dpkg-selections.list`” で復元できます。

注意
`/proc` や `/sys` や `/tmp` や `/run` 上にある擬似ファイルシステム (項 1.2.12 と項 1.2.13 参照) の内容をバックアップしてはいけません。あなた自身が自分がしていることの意味を余程よく分かっていなければ、これらの内容は巨大で無用なデーターです。

注意
データーをバックアップする際には MTA (項 6.2.4 を参照下さい) 等のアプリケーションデーモンを停止するのも一計です。

10.2.2 バックアップユーティリティーのスイート

Debian システム上で利用可能でバックアップユーティリティーのスイートのなかで際立った選ばれたリストを記します。

バックアップツールにはそれぞれの特別な狙いがあります。

- Mondo Rescue を使うと、普通のインストールプロセスを経ずにバックアップ CD/DVD 等から完全なシステムを迅速に復旧できます。
- Bacula と Amanda と BackupPC は、ネットワーク越しの定期的バックアップに焦点のあるフル機能のバックアップスイートです。
- Duplicity や Borg は典型的ワークステーション向けの比較的簡単なバックアップユーティリティーです。

10.2.3 バックアップのティップ

個人用ワークステーションでは、サーバー環境用に設計されたフル装備バックアップユーティリティー一式はうまく機能しないかもしれません。同時に、既存のワークステーション用のバックアップユーティリティーでは足りない面があるかもしれません。

ここに最小限のユーザー努力でより簡単にバックアップできるようするティップスを示します。これらの技法はどのバックアップユーティリティーにでも使えます。

例示のために、主ユーザーとグループ名を `penguin` とし、バックアップとスナップショット用スクリプト例”`/usr/local/bin/bkss.sh`” を以下として作成します:

パッケージ	ポプコン	サイズ	説明
bacula-common	V:5.7, I:6.4	2501	Bacula: ネットワークバックアップ、復元および検証 - 共通のサポートファイル
bacula-client	V:0.2, I:1.7	199	Bacula: ネットワークバックアップ、復元および検証 - クライアントメタパッケージ
bacula-console	V:0.6, I:2.0	112	Bacula: ネットワークバックアップ、復元および検証 - テキストコンソール
bacula-server	I:0.56	199	Bacula: ネットワークバックアップ、復元および検証 - サーバーメタパッケージ
amanda-common	V:0.5, I:2.0	9952	Amanda: Advanced Maryland Automatic Network Disk Archiver (ライブラリー)
amanda-client	V:0.5, I:2.0	1110	Amanda: Advanced Maryland Automatic Network Disk Archiver (クライアント)
amanda-server	V:0.11, I:0.22	1096	Amanda: Advanced Maryland Automatic Network Disk Archiver (サーバー)
backuppc	V:1.3, I:1.5	3088	BackupPC は高性能でエンタープライズ級の、PC をバックアップするためのシステム (ディスクベース)
duplicity	V:5, I:44	2679	(リモート) 増分バックアップ
deja-dup	V:28, I:41	5274	duplicity の GUI フロントエンド
borgbackup	V:13, I:28	3717	(リモート) 重複回避バックアップ
borgmatic	V:3.4, I:4.4	946	borgbackup のヘルパー
rdiff-backup	V:2.1, I:6.0	1204	(リモート) 増分バックアップ
restic	V:10, I:17	44216	(リモート) 増分バックアップ
backupninja	V:2.0, I:2.4	360	軽量で拡張可のメタバックアップシステム
slbackup	V:0.10, I:0.12	147	(リモート) 増分バックアップ
backup-manager	V:0.39, I:0.68	573	コマンドラインのバックアップツール
backup2l	V:0.34, I:0.46	110	マウントできるメディアのための低メンテナンスのバックアップ/復旧ツール (ディスクベース)

Table 10.5: バックアップスイートのユーティリティーのリスト

```
#!/bin/sh -e
SRC="$1" # source data path
DSTFS="$2" # backup destination filesystem path
DSTSV="$3" # backup destination subvolume name
DSTSS="${DSTFS}/${DSTSV}-snapshot" # snapshot destination path
if [ "$(stat -f -c %T "$DSTFS")" != "btrfs" ]; then
  echo "E: $DESTFS needs to be formatted to btrfs" >&2 ; exit 1
fi
MSGID=$(notify-send -p "bkup.sh $DSTSV" "in progress ...")
if [ ! -d "$DSTFS/$DSTSV" ]; then
  btrfs subvolume create "$DSTFS/$DSTSV"
  mkdir -p "$DSTSS"
fi
rsync -aHxS --delete --mkpath "${SRC}/" "${DSTFS}/${DSTSV}"
btrfs subvolume snapshot -r "${DSTFS}/${DSTSV}" ${DSTSS}/$(date -u --iso=min)
notify-send -r "$MSGID" "bkup.sh $DSTSV" "finished!"
```

ここでは、基本ツールの rsync(8) のみを使いシステムのバックアップが実現され、Btrfs を使うことでストレージスペースが有効利用されています。

ティップ
参考: 当著者は、自作の類似シェルスクリプト”bss: Btrfs Subvolume Snapshot Utility” を自身のワークステーションで使用しています。

10.2.3.1 GUI バックアップ

単一 GUI クリックによるバックアップの設定例をここに示します。

- USB ストレージデバイスをバックアップ用に準備します。
  - ”BKUP” というラベル名のついた btrfs の 1 パーティションがあるように USB ストレージデバイスをフォーマットします。これは暗号化されていても良いです (項 9.9.1 を参照下さい)。
  - これをあなたのシステムに挿入します。デスクトップシステムがそれを”`/media/penguin/BKUP`” をして自動マウントするはずです。
  - 当該ユーザーにより書き込みできるように”`sudo chown penguin:penguin /media/penguin/BKUP`” と実行します。
- 項 9.4.10 中に記載された技法に則り”`~/.local/share/applications/BKUP.desktop`” を以下のように作成します:

```
[Desktop Entry]
Name=bkss
Comment=Backup and snapshot of ~/Documents
Exec=/usr/local/bin/bkss.sh /home/penguin/Documents /media/penguin/BKUP Documents
Type=Application
```

各 GUI クリック毎に、あなたのデーターが”`~/Documents`” から USB ストレージデバイスにバックアップされ、リードオンリーのスナップショットが作成されます。

10.2.3.2 マウントイベントがトリガーするバックアップ

マウントイベントによりトリガーされる自動バックアップの設定例をここに記します。

- 項 10.2.3.1 と同様にし、USB ストレージデバイスをバックアップ用に準備します。

- systemd サービスユニットファイル”~/.config/systemd/user/back-BKUP.service” を以下のように作成します:

```
[Unit]
Description=USB Disk backup
Requires=media-%u-BKUP.mount
After=media-%u-BKUP.mount

[Service]
ExecStart=/usr/local/bin/bkss.sh %h/Documents /media/%u/BKUP Documents
StandardOutput=append:%h/.cache/systemd-snap.log
StandardError=append:%h/.cache/systemd-snap.log

[Install]
WantedBy=media-%u-BKUP.mount
```

- 以下のようにして systemd のユニット設定を有効化します。

```
$ systemctl --user enable bkup-BKUP.service
```

各マウントイベント毎に、あなたのデーターが”~/Documents” から USB ストレージデバイスにバックアップされ、リードオンリーのスナップショットが作成されます。

ここで、systemd が現在メモリー上に保持する systemd のマウント unit 名を、呼び出しているユーザーのサービスマネージャーに問い合わすには、”systemctl --user list-units --type=mount” を使います。

10.2.3.3 タイマーイベントがトリガーするバックアップ

タイマーイベントによりトリガーされる自動バックアップの設定例をここに記します。

- 項 10.2.3.1 と同様にし、USB ストレージデバイスをバックアップ用に準備します。
- systemd タイマーユニットファイル”~/.config/systemd/user/snap-Documents.timer” を以下のように作成します:

```
[Unit]
Description=Run btrfs subvolume snapshot on timer
Documentation=man:btrfs(1)

[Timer]
OnStartupSec=30
OnUnitInactiveSec=900

[Install]
WantedBy=timers.target
```

- systemd サービスユニットファイル”~/.config/systemd/user/snap-Documents.service” を以下のように作成します:

```
[Unit]
Description=Run btrfs subvolume snapshot
Documentation=man:btrfs(1)

[Service]
Type=oneshot
Nice=15
ExecStart=/usr/local/bin/bkss.sh %h/Documents /media/%u/BKUP Documents
IOSchedulingClass=idle
CPUSchedulingPolicy=idle
StandardOutput=append:%h/.cache/systemd-snap.log
StandardError=append:%h/.cache/systemd-snap.log
```

- 以下のようにして systemd のユニット設定を有効化します。

```
$ systemctl --user enable snap-Documents.timer
```

各タイマーイベント毎に、あなたのデーターが"~/Documents" から USB ストレージデバイスにバックアップされ、リードオンリーのスナップショットが作成されます。

ここで、systemd が現在メモリー上に保持する systemd のタイマー unit 名を、呼び出しているユーザーのサービスマネージャーに問い合わすには、"systemctl --user list-units --type=mount" を使います。

現代的なデスクトップシステムでは、この systemd を使うアプローチのほうが at(1) や cron(8) や anacron(8) を使う伝統的 Unix アプローチよりより精緻な制御を提供できます。

10.3 データーセキュリティーのインフラ

データーのセキュリティーのインフラはデーターの暗号化のツールとメッセージダイジェストのツールと署名ツールの組み合わせで提供されます。

パッケージ	ポプコン	サイズ	コマンド	説明
gnupg	V:403, I:853	465	gpg(1)	GNU プライバシーガード - OpenPGP 暗号化と署名ツール
gpgv	V:206, I:911	576	gpgv(1)	GNU プライバシーガード - OpenPGP 署名検証ツール
sq	V:1, I:37	24191	sq(1)	Sequoia-PGP - OpenPGP 暗号化と署名ツール
sqv	V:517, I:595	1919	sqv(1)	Sequoia-PGP - OpenPGP 署名検証ツール
paperkey	V:1, I:12	58	paperkey(1)	OpenPGP の秘密キーから秘密の情報だけを抜粋
cryptsetup	V:35, I:77	476	cryptsetup(8), …	暗号化されたブロックデバイス (dm-crypt / LUKS) のためのユーティリティー
coreutils	V:914, I:1000	19751	md5sum(1)	MD5 メッセージダイジェストを計算やチェック
coreutils	V:914, I:1000	19751	sha1sum(1)	SHA1 メッセージダイジェストを計算やチェック
openssl	V:853, I:996	2564	openssl(1ssl)	"openssl dgst" を使ってメッセージダイジェストを計算やチェック (OpenSSL)
libsecret-tools	V:2, I:14	44	secret-tool(1)	パスワードの保存と読出し (CLI)
seahorse	V:76, I:258	7999	seahorse(1)	キー管理ツール (GNOME)

Table 10.6: データーセキュリティーインフラツールのリスト

Linux カーネルモジュール経由で自動的データー暗号化のインフラを実現する dm-crypt と fscrypt に関しては項 9.9 を参照下さい。

10.3.1 Gnupg のためのキー管理

基本的なキー管理に関する GNU プライバシガードコマンドを次に記します。

トラストコードの意味を次に記します。

以下のようにすると私のキー"9563FC29932C409F1A77F9C1AB8A1522D8234C6A" を作成出来ます。

コマンド	説明
`gpg --gen-key`	新規キーの生成
`gpg --gen-revoke my_user_ID`	my_user_ID に関するリボークキーを生成
`gpg --edit-key user_ID`	インタラクティブにキーを編集、ヘルプは”help”
`gpg -o file --export`	全ての公開キーをファイルにエクスポート
`gpg -o file --export-secret-keys`	全ての秘密キーをファイルにエクスポート
`gpg --import file`	全てのキーをファイルからインポート
`gpg --send-keys user_ID`	user_ID のキーをキーサーバーに送信
`gpg --recv-keys user_ID`	user_ID のキーをキーサーバーから受信
`gpg --list-keys user_ID`	user_ID のキーをリスト
`gpg --list-sigs user_ID`	user_ID の署名をリスト
`gpg --check-sigs user_ID`	user_ID の署名をチェック
`gpg --fingerprint user_ID`	user_ID のフィンガープリントをチェック
`gpg --refresh-keys`	ローカルキーリングを更新

Table 10.7: キー管理のための GNU プライバシガードコマンドのリスト

コード	信用の説明
`-`	所有者への信用未付与/未計算
`e`	信用計算に失敗
`q`	計算用の情報不十分
`n`	このキーを信用不可
`m`	スレスレの信用
`f`	フルに信用
`u`	究極の信用

Table 10.8: トラストコードの意味のリスト

```
$ gpg --gen-key
gpg (GnuPG) 2.4.7; Copyright (C) 2024 g10 Code GmbH
...
GnuPG needs to construct a user ID to identify your key.

Real name: Osamu Aoki
Email address: osamu@debian.org
You selected this USER-ID:
    "Osamu Aoki <osamu@debian.org>"

Change (N)ame, (E)mail, or (O)kay/(Q)uit? o
...
public and secret key created and signed.

pub   ed25519 2026-02-14 [SC] [expires: 2029-02-13]
      9563FC29932C409F1A77F9C1AB8A1522D8234C6A
uid                      Osamu Aoki <osamu@debian.org>
sub   cv25519 2026-02-14 [E] [expires: 2029-02-13]
```

以下のようにすると私のキー”9563FC29932C409F1A77F9C1AB8A1522D8234C6A” をポピュラーなキーサーバー”hkp://keys.gnupg.net” にアップロード出来ます。

```
$ gpg --keyserver hkp://keys.gnupg.net --send-keys 9563FC29932C409F1A77F9C1AB8A1522D8234C6A
```

”~/.gnupg/gpg.conf” (もしくは古い場所”~/.gnupg/options”) 中の良いデフォールトのキーサーバーの設定は次を含みます。

```
keyserver hkp://keys.gnupg.net
```

次によってキーサーバーから知らないキーが獲得できます。

```
$ gpg --list-sigs --with-colons | grep '^sig.*\[User ID not found\]' |\
          cut -d ':' -f 5| sort | uniq | xargs gpg --recv-keys
```

OpenPGP 公開キーサーバー (バージョン 0.9.6 以前) に 2 つ以上サブキーのあるキーを壊すバグがありました。新しい gnupg (>1.2.1-2) パッケージはこのような壊れたサブキーを取り扱えます。gpg(1) の”--repair-pks-subkey-bug” オプションの説明を参照下さい。

SHA-1 をハッシュに使うのは非推奨です。あなたの古い RSA ベースの openPGP キーが SHA-1 を使用していれば、FixKeyWithSha1 を用いて修正しましょう。

ティップ
sq(1) と sqv(1) コマンドは openPGP コマンドの代替セットです。sq user documentation と A Practical Introduction to using sq, Sequoia PGP’s CLI を参照ください。

10.3.2 GnuPG をファイルに使用

基本的なキー管理に関する GNU プライバシガードコマンドを次に記します。

10.3.3 Mutt で GnuPG を使用

インデックスメニュー上で”S” とすれば GnuPG が使えるようにしておきながら、遅い GnuPG が自動的に起動しないように”~/.muttrc” に以下の内容を追加します。

```
macro index S ":toggle pgp_verify_sig\n"
set pgp_verify_sig=no
```

コマンド	説明
`gpg -a -s file`	ファイルを ASCII 文字化した file.asc と署名
`gpg --armor --sign file`	,,
`gpg --clearsign file`	メッセージをクリアサイン
`gpg --clearsign file|mail foo@example.org`	`foo@example.org` にクリアサインされたメッセージをメールする
`gpg --clearsign --not-dash-escaped patchfile`	パッチファイルをクリアサイン
`gpg --verify file`	クリアサインされたファイルを確認
`gpg -o file.sig -b file`	署名を別ファイルで作成
`gpg -o file.sig --detach-sign file`	,,
`gpg --verify file.sig file`	file.sig を使ってファイルを確認
`gpg -o crypt_file.gpg -r name -e file`	file からバイナリー crypt_file.gpg への name 宛公開キー暗号化
`gpg -o crypt_file.gpg --recipient name --encrypt file`	,,
`gpg -o crypt_file.asc -a -r name -e file`	file から ASCII 文字化された crypt_file.asc への name 宛公開キー暗号化
`gpg -o crypt_file.gpg -c file`	file からバイナリー crypt_file.gpg への対称暗号化
`gpg -o crypt_file.gpg --symmetric file`	,,
`gpg -o crypt_file.asc -a -c file`	file から ASCII 文字化された crypt_file.asc への対称暗号化
`gpg -o file -d crypt_file.gpg -r name`	暗号解読
`gpg -o file --decrypt crypt_file.gpg`	,,

Table 10.9: ファイルに使用する GNU プライバシーガードコマンドのリスト

10.3.4 Vim で GnuPG を使用

gnupg のプラグインを使うと”.gpg” や”.asc” や”.pgp” というファイル拡張子のファイルに対して透過的に GnuPG を実行できます。[3]

```
$ sudo aptitude install vim-scripts
$ echo "packadd! gnupg" >> ~/.vim/vimrc
```

10.3.5 MD5 和

md5sum(1) は rfc1321 の方法を使ってダイジェストファイルを作成し各ファイルをそれで確認するユーティリティーを提供します。

```
$ md5sum foo bar >baz.md5
$ cat baz.md5
d3b07384d113edec49eaa6238ad5ff00  foo
c157a79031e1c40f85931829bc5fc552  bar
$ md5sum -c baz.md5
foo: OK
bar: OK
```

注意
MD5 和の計算は GNU プライバシーガード (GnuPG) による暗号学的署名の計算より CPU への負荷がかかりません。通常、一番上のレベルのダイジェストファイルだけがデーターの整合性のために暗号学的に署名されます。

10.3.6 パスワードキーリング

GNOME システム上では、seahorse(1) がキーリング `~/.local/share/keyrings/*` 中にパスワードを管理し保存します。

secret-tool(1) はコマンドラインからパスワードをキーリングに保存できます。

ディスクイメージを LUKS/dm-crypt 暗号化するためのパスフレーズを保存します。

```
$ secret-tool store --label='LUKS passphrase for disk.img' LUKS my_disk.img
Password: ********
```

cryptsetup(8) のような他のプログラムに、こうして保存されたパスワードを読み出し供給できます。

```
$ secret-tool lookup LUKS my_disk.img | \
  cryptsetup open disk.img disk_img --type luks --keyring -
$ sudo mount /dev/mapper/disk_img /mnt
```

ティップ
スクリプト中でパスワードを供給する必要がある際はいつも `secret-tool` を使いパスフレーズを直接ハードコードすることは避けます。

10.4 ソースコードマージツール

ソースコードをマージする多くのツールがあります。以下のコマンドが著者の目に止まりました。

[3] もし、”`~/.vim/vimrc`” に代えて”`~/.vimrc`” を使う場合は、それに合わせて置換して下さい。

パッケージ	ポプコン	サイズ	コマンド	説明
patch	V:100, I:729	243	patch(1)	差分ファイルをオリジナルに適用
vim	V:73, I:330	4415	vimdiff(1)	vim で 2 つのファイルを並べて比較
imediff	V:0.02, I:0.21	329	imediff(1)	対話型フルスクリーンの 2 方/3 方マージツール
meld	V:5, I:23	3996	meld(1)	ファイルを比較やマージ (GTK)
wiggle	V:0.01, I:0.12	191	wiggle(1)	リジェクトされたパッチを適用
diffutils	V:901, I:998	1768	diff(1)	1 行ごとにファイルを比較
diffutils	V:901, I:998	1768	diff3(1)	1 行ごとにファイルを比較やマージ
quilt	V:2, I:16	880	quilt(1)	パッチのシリーズを管理
wdiff	V:5, I:37	580	wdiff(1)	テキストファイル間のワードの相違表示
diffstat	V:8, I:91	86	diffstat(1)	差分ファイルによる変化のヒストグラム作成
patchutils	V:10, I:91	269	combinediff(1)	2 つの積み重ねパッチから 1 つの合計パッチを生成
patchutils	V:10, I:91	269	dehtmldiff(1)	HTML ページから差分ファイルを抽出
patchutils	V:10, I:91	269	filterdiff(1)	差分ファイルから差分ファイルを抽出や削除
patchutils	V:10, I:91	269	fixcvsdiff(1)	CVS により作成された patch(1) が誤解する差分ファイルを修正
patchutils	V:10, I:91	269	flipdiff(1)	古い 2 つのパッチを交換
patchutils	V:10, I:91	269	grepdiff(1)	正規表現にマッチするパッチによって変更されるファイルを表示
patchutils	V:10, I:91	269	interdiff(1)	2 つのユニファイド差分ファイル間の違いを表示
patchutils	V:10, I:91	269	lsdiff(1)	どのファイルがパッチによって変更されるかを表示
patchutils	V:10, I:91	269	recountdiff(1)	ユニファイドコンテキスト差分ファイルのカウントやオフセットを再計算
patchutils	V:10, I:91	269	rediff(1)	手編集された差分ファイルのカウントやオフセットを再計算
patchutils	V:10, I:91	269	splitdiff(1)	増分パッチの分離
patchutils	V:10, I:91	269	unwrapdiff(1)	ワードラップされたパッチを復元
dirdiff	V:0.1, I:1.3	167	dirdiff(1)	ディレクトリーツリー間で相違点の表示と変更のマージ
docdiff	V:0.02, I:0.15	554	docdiff(1)	2 つのファイルをワード毎/文字毎に比較
makepatch	V:0.01, I:0.14	99	makepatch(1)	拡張パッチファイルの生成
makepatch	V:0.01, I:0.14	99	applypatch(1)	拡張パッチファイルの適用

Table 10.10: ソースコードマージツールのリスト

10.4.1 ソースファイル間の相違の抽出

ふたつのソースファイル間の相違を抽出したユニファイド差分ファイルは、以下の要領でファイル位置に対応し"`file.patch0`" か"`file.patch1`" として作成されます。

```
$ diff -u file.old file.new > file.patch0
$ diff -u old/file new/file > file.patch1
```

10.4.2 ソースファイルに更新をマージ

差分ファイル (別名、パッチファイル) はプログラム更新を送るのに使われます。受け取った側はこの更新を別のファイルに以下のようにして適用します。

```
$ patch -p0 file < file.patch0
$ patch -p1 file < file.patch1
```

10.4.3 インタラクティブなマージ

2 つのバージョンのソースコードがある場合、imediff(1) をインタラクティブに使って効率的に 2 方マージを以下のように実行できます。

```
$ imediff -o file.merged file.old file.new
```

3 つのバージョンのソースコードがある場合、imediff(1) をインタラクティブに使って 3 方マージを以下のように実行できます。

```
$ imediff -o file.merged file.yours file.base file.theirs
```

10.5 Git

最近は、ローカルとリモートの両方のコード管理一切が可能な Git がバージョンコントロールシステム (VCS) として最優先の選択肢です。

Debian は Debian Salsa サービス経由でフリーの Git サービスを提供します。その説明文書は https://wiki.debian.org/Salsa にあります。

Git 関連パッケージは以下です。

10.5.1 Git クライアントの設定

Git は使うあなたの名前や email アドレス等を"`~/.gitconfig`" 中のいくつかのグローバル設定に設定したいなら以下のようにします。

```
$ git config --global user.name "Name Surname"
$ git config --global user.email yourname@example.com
```

以下によって Git のデフォルトの動作をカスタマイズしてもいいです。

```
$ git config --global init.defaultBranch main
$ git config --global pull.rebase true
$ git config --global push.default current
```

パッケージ	ポプコン	サイズ	コマンド	説明
git	V:442, I:630	52512	git(1)	git: 高速、スケーラブル、分散型リビジョンコントロールシステム
gitk	V:3, I:25	2045	gitk(1)	GUI による履歴付き Git レポジトリーブラウザー
qgit	V:0.2, I:1.8	1431	qgit(1)	GUI による履歴付き Git レポジトリーブラウザー
git-cola	V:0.7, I:4.4	5331	git-cola(1)	GUI による履歴付き Git レポジトリーブラウザー
tig	V:2, I:11	1245	tig(1)	コンソールによる履歴付き Git レポジトリーブラウザー
lazygit	V:1.0, I:4.1	24167	lazygit(1)	コンソールによる履歴付き Git レポジトリーブラウザー
git-gui	V:1, I:15	2549	git-gui(1)	Git 用の GUI (履歴無し)
git-email	V:0.4, I:8.6	1229	git-send-email(1)	Git からパッチの集合の email として送信
git-buildpackage	V:1.2, I:6.5	2027	git-buildpackage(1)	Git を使って Debian パッケージ化を自動化
dgit	V:0.12, I:0.91	758	dgit(1)	Debian アーカイブと git の相互運用性
imediff	V:0.02, I:0.21	329	git-ime(1)	インタラクティブな git コミット分割ヘルパーツール
stgit	V:0.07, I:0.45	6005	stg(1)	Git 上の quilt (Python)
git-doc	I:9.6	15192	N/A	正式 Git 文書
gitmagic	I:0.43	721	N/A	”Git マジック”、Git に関する分かり易いガイド

Table 10.11: git 関連のパッケージとコマンドのリスト

もしあなたが CVS や Subversion コマンドに慣れ過ぎている場合には、いくつかのコマンドエイリアスを以下のように設定するのも一計です。

```
$ git config --global alias.ci "commit -a"
$ git config --global alias.co checkout
```

あなたのグローバル設定は以下のようにするとチェックできます。

```
$ git config --global --list
```

10.5.2 基本 Git コマンド

Git 操作にはいくつかのデーターが関与します。

- ユーザーから見えるファイルを保持しかつ変更の対象とする、ワーキングツリー
  - 記録すべき変更は、インデックスに明示的に選択して候補として挙げなければいけません。これは、`git add` や `git rm` コマンドです。
- 候補として挙げたファイルを保持するインデックス
  - 候補として挙げたファイルはそれに続くリクエストでローカルのレポジトリーにコミットされるでしょう。これは、`git commit` コマンドです。
- コミットされたファイルを保持するローカルのレポジトリー
  - Git はコミットされたデーターのリンクされた履歴をレポジトリー中にブランチとして整理して保存します。
  - `git push` コマンドによって、ローカルレポジトリーはリモートレポジトリーにデーターを送信できます。

  - `git fetch` や `git pull` コマンドによって、ローカルレポジトリーはリモートレポジトリーからデーターを受信できます。
    * `git pull` コマンドは `git fetch` コマンドの後で `git merge` か `git rebase` コマンドを実行します。
    * ここで、`git merge` は、2 つの別々の履歴のブランチの最後を 1 点にまとめます。(これは、カスタマイズされていないデフォルトの場合の `git pull` で、多くの人に向けてブランチを公開するアップストリームに好適です。)
    * ここで、`git rebase` は、リモートブランチの履歴の後ろにローカルブランチの履歴が繋がった連続履歴の単一ブランチを生成します。(これは、`pull.rebase true` カスタマイゼイションの場合で、残りの我々に好適です。)

- コミットされたファイルを保持するリモートレポジトリー
  - リモートレポジトリーとの通信は SSH か HTTPS 等のセキュアー通信プロトコルを使います。

ワーキングツリーは`.git/` ディレクトリーの外のファイルです。`.git/` ディレクトリーの内部のファイルはインデックス、ローカルレポジトリーデーター、いくつかの git 設定のテキストファイルを保持します。

主要 Git コマンドの概論です。

Git コマンド	機能
`git init`	(ローカル) レポジトリーを作成します
`git clone URL`	リモートレポジトリーをワーキングツリー付きローカルレポジトリーとしてクローンします
`git pull origin main`	ローカルの `main` ブランチをリモートレポジトリーの `origin` を使って更新します
`git add .`	インデックス中にのみ既に存在するファイルに関してワーキングツリー中のファイルをインデックスに追加します
`git add -A .`	ワーキングツリー内の全てのファイルを全てのファイルに関するインデックスに削除を含め加えます
`git rm filename`	ワーキングツリーとインデックスからファイルを削除します
`git commit`	ローカルレポジトリーにインデックス中の挙げられた変更をコミットします
`git commit -a`	ワーキングツリー中の全ての変更をインデックスに追加し、それらをローカルレポジトリーにコミットします (add + commit)
`git push -u origin branch_name`	リモートレポジトリー `origin` をローカルの `branch_name` ブランチで更新します (最初の呼び出し)
`git push origin branch_name`	リモートレポジトリー `origin` をローカルの `branch_name` ブランチで更新します (その後の呼び出し)
`git diff treeish1 treeish2`	*treeish1* コミットと *treeish2* コミットの間の相違を表示します
`gitk`	VCS レポジトリーのブランチヒストリーツリーの GUI 表示をします

Table 10.12: 主要 Git コマンド

10.5.3 Git ティップ

以下は Git ティップです。



警告
たとえ gitk(1) 等の一部ツールが受け付けるからといって、タグ文字列中にスペースを使ってはいけません。他の `git` コマンドで支障が起こるかもしれません。

Git コマンドライン	機能
`gitk --all`	Git 全履歴を閲覧し HEAD を別のコミットにリセットし、パッチをチェリーピックし、タグやブランチを生成するような、Git 履歴への操作を加えます
`git stash`	データーを消失無くクリーンなワーキングツリーを得ます
`git remote -v`	リモートに関する設定をチェックします
`git branch -vv`	ブランチに関する設定をチェックします
`git status`	ワーキングツリーの状態を表示します
`git config -l`	git 設定のリストをします
`git reset --hard HEAD; git clean -x -d -f`	全てのワーキングツリーの変更を元に戻し完全にクリーンアップします
`git rm --cached filename`	`git add filename` で変更された候補のインデックスを元に戻します
`git reflog`	レファレンスログを取得します (削除したブランチからコミットを復元するのに有用です)
`git branch new_branch_name HEAD@{6}`	reflog 情報から新規ブランチを生成します
`git remote add new_remote URL`	URL によって指される `new_remote` リモートレポジトリーを追加します
`git remote rename origin upstream`	リモートレポジトリー名を `origin` から `upstream` に変更します
`git branch -u upstream/branch_name`	リモートレポジトリー `upstream` とそのブランチ名 `branch_name` にリモートトラッキングを設定します。
`git remote set-url origin https://foo/bar.git`	`origin` の URL を変更します
`git remote set-url --push upstream DISABLED`	`upstream` へのプシュを無効化します (`.git/config` を編集して再有効化)
`git remote update upstream`	`upstream` レポジトリー中の全リモートブランチの更新を取得します
`git fetch upstream foo:upstream-foo`	ローカルの (孤立しているかもしれない) `upstream-foo` ブランチを、`upstream` レポジトリー中の `foo` ブランチのコピーとして作成します
`git checkout -b topic_branch ; git push -u topic_branch origin`	新規 `topic_branch` を作成しそれを `origin` にプシュします
`git branch -m oldname newname`	ローカルブランチ名を変更します
`git push -d origin branch_to_be_removed`	リモートブランチを削除します (新手法)
`git push origin :branch_to_be_removed`	リモートブランチを削除します (旧手法)
`git checkout --orphan unconnected`	新 `unconnected` ブランチを生成します
`git rebase -i origin/main`	きれいなブランチ履歴のために、`origin/main` からのコミットを順序変更/削除/押し潰します
`git reset HEAD^; git commit --amend`	最後の 2 コミットを 1 つに押し潰します
`git checkout topic_branch ; git merge --squash topic_branch`	全 `topic_branch` を 1 つのコミットに押し潰します
`git fetch --unshallow --update-head-ok origin '+refs/heads/*:refs/heads/*'`	浅いクローンをブランチのフルクローンに変換します
`git ime`	最後のコミットを分割して一連のファイル毎の小さなコミット等に分割します (`imediff` パッケージが必要)
`git repack -a -d; git prune`	ローカルレポジトリーを単一の梱包に再梱包します (こうすると消去したブランチ等からのデーター復元の可能性を制限するかもしれません)

Table 10.13: Git ティップ



注意
もしリモートレポジトリーにプッシュしたローカルブランチがリベースしたりスクワッシュした場合には、このブランチをプッシュするのはリスクがあるし、`--force` オプションが必要です。これは `main` ブランチでは通常許容されませんが、`main` ブランチにマージする前のトピックブランチでは許容されます。



注意
`git` サブコマンドを直接"`git-xyz`" としてコマンドラインから起動するのは 2006 年初以来非推奨です。

ティップ
`$PATH` で指定されたパス中に実行可能ファイル `git-foo` が存在する場合、ハイフン無しの"`git foo`" をコマンドラインに入力するとこの `git-foo` が起動されます。これは `git` コマンドの機能です。

10.5.4 Git リファレンス

次を参照下さい。

- マンページ: git(1) (`/usr/share/doc/git-doc/git.html`)
- Git ユーザーマニュアル (`/usr/share/doc/git-doc/user-manual.html`)
- git へのチュートリアル導入 (`/usr/share/doc/git-doc/gittutorial.html`)
- git へのチュートリアル導入: 第 2 部 (`/usr/share/doc/git-doc/gittutorial-2.html`)
- 約 20 のコマンドを使って毎日 GIT (`/usr/share/doc/git-doc/giteveryday.html`)
- Git マジック (`/usr/share/doc/gitmagic/html/index.html`)

10.5.5 他のバージョンコントロールシステム

バージョンコントロールシステム (VCS) はリビジョンコントロールシステム (RCS) とかソフトウェアー設定管理 (SCM) という別名もあります。

Debian システム上で利用可能な特記すべき他の非 Git の VCS のまとめを以下に記します。

パッケージ	ポプコン	サイズ	ツール	**VCS** タイプ	コメント
`mercurial`	V:3, I:24	2667	Mercurial	分散型	Python と一部 C で書かれた DVCS
`darcs`	V:0.2, I:2.9	38856	Darcs	分散型	パッチに関して賢い計算をする DVCS (遅い)
`tla`	V:0.03, I:0.60	1022	GNU arch	分散型	主に Tom Lord による DVCS (歴史的)
`bzr`	V:0.1, I:3.8	28	GNU Bazaar	分散型	`tla` に影響され Python で書かれた DVCS
`subversion`	V:9, I:53	5211	Subversion	リモート	” 良く出来た CVS”、新しいリモート VCS の標準 (歴史的)
`cvs`	V:3, I:22	4835	CVS	リモート	リモート VCS の過去の標準 (歴史的)
`tkcvs`	V:0.10, I:0.87	34	CVS, …	リモート	VCS (CVS, Subversion, RCS) レポジトリーツリーの GUI 表示
`rcs`	V:1.6, I:8.8	578	RCS	ローカル	”Unix SCCS の本来あるべき姿” (歴史的)
`cssc`	V:0.01, I:0.27	2044	CSSC	ローカル	Unix SCCS のクローン (歴史的)

Table 10.14: 他のバージョンコントロールシステムツールのリスト

Chapter 11

データー変換

Debian システム上のデーターフォーマット変換のツールとティップを記します。

標準に準拠したツールは非常に良い状態ですが、プロプライエタリデーターフォーマットのサポートは限定的です。

11.1 テキストデーター変換ツール

テキストデーター変換のための以下のパッケージが著者の目に止まりました。

パッケージ	ポプコン	サイズ	キーワード	説明
libc6	V:924, I:1000	5483	文字セット	iconv(1) によるロケール間のテキスト符号化方式変換ソフト (基本的)
recode	V:2, I:12	534	文字セット + 行末文字	ロケール間のテキスト符号化方式変換ソフト (機能豊富、より多いエイリアスと機能)
konwert	V:2, I:42	137	文字セット	ロケール間のテキスト符号化方式変換ソフト (高級機能)
nkf	V:0.4, I:8.3	360	文字セット	日本語のための文字セット翻訳ソフト
tcs	V:0.01, I:0.12	518	文字セット	文字セット翻訳ソフト
unaccent	V:0.03, I:0.26	35	文字セット	アクセント付き文字をアクセントの無しの等価文字に置換
tofrodos	V:1, I:11	50	行末文字	DOS と Unix 間のテキストフォーマット変換ソフト: fromdos(1) と todos(1)
macutils	V:0.03, I:0.38	319	行末文字	Macintosh と Unix 間のテキストフォーマット変換ソフト: frommac(1) and tomac(1)

Table 11.1: テキストデーター変換ツールのリスト

11.1.1 テキストファイルを **iconv** を使って変換

ティップ
iconv(1) は libc6 パッケージの一部として提供されていて、文字の符号化方式変換のために実質的に全ての Unix 的システムで常に利用可能です。

以下のようにするとテキストファイルを iconv(1) を使って変換できます。

```
$ iconv -f encoding1 -t encoding2 input.txt >output.txt
```

符号化方式 (エンコーディング) 値をマッチングする際には、大文字小文字の区別は無く、”-” や”_” を無視します。”iconv -l” コマンドにより、サポートされている符号化方法が確認できます。

符号化方式値	使い方
ASCII	情報交換用米国標準コード (ASCII); アクセント文字無しの 7 ビットコード
UTF-8	全現代的 OS のための現行多言語標準
ISO-8859-1	西欧州言語用の旧標準、ASCII + アクセント文字
ISO-8859-2	東欧州言語用の旧標準、ASCII + アクセント文字
ISO-8859-15	西欧州言語用の旧標準、ユーロ文字付き ISO-8859-1
CP850	コードページ 850、西欧州言語用グラフィック文字付き Microsoft DOS 文字、ISO-8859-1 の変種
CP932	コードページ 932、日本語用 Microsoft Windows スタイル Shift-JIS の変種
CP936	コードページ 936、簡体中国語用 Microsoft Windows スタイル GB2312 か GBK か GB18030 の変種
CP949	コードページ 949、韓国語用 Microsoft Windows スタイル EUC-KR か統一ハングルコードの変種
CP950	コードページ 950、繁体中国語用 Microsoft Windows スタイル Big5 の変種
CP1251	コードページ 1251、キリル文字用 Microsoft Windows スタイル符号化方式
CP1252	コードページ 1252、西欧州言語用 Microsoft Windows スタイル ISO-8859-15 の変種
KOI8-R	キリル文字用の旧ロシアの UNIX 標準
ISO-2022-JP	7 ビットコードのみを用いる日本語 email の標準符号化方式
eucJP	Shift-JIS とはまったく違う、旧日本の UNIX 標準 8 ビットコード
Shift-JIS	日本語のための JIS X 0208 Appendix 1 標準 (CP932 を参照下さい)

Table 11.2: 符号化方式値とその使い方リスト

注意
一部の符号化方式 (エンコーディング) はデーター変換のみサポートされており、ロケール値としては使われません (項 8.1 を参照下さい)。

ASCII や ISO-8859 文字セットのような 1 バイトに収まる文字セットに付いては、文字の符号化方式 (エンコーディング) とは文字セットとほとんど同じ事を意味します。

日本語のための JIS X 0213 や実質的に全ての言語のためのユニコード文字セット (UCS, Unicode, ISO-10646-1) のような多くの文字を含む文字セットの場合には、バイトデーター列に落とし込む多くの符号化手法があります。

- 日本語用には、EUC と ISO/IEC 2022 (別名 JIS X 0202)
- ユニコード用には、UTF-8 と UTF-16/UCS-2 と UTF-32/UCS-4

これらに関しては、文字セットと文字符号化方式の間にはっきりとした区別があります。

コードページは、一部のベンダー固有のコードページで文字符号化テーブルと同義語として使用されています。

注意
ほとんどの符号化システムが 7 ビット文字に関して ASCII と同じコードを共有している事を承知下さい。もちろん例外はありますが。もし古い日本語の C プログラムや URL のデーターをカジュアルにシフト JIS と呼ばれている符号化フォーマットから UTF-8 フォーマットに変換する際には、期待される結果を得るために”shift-JIS” ではなく”CP932” を使います: 0x5C → ”\” と 0x7E → ”~”。こうしないと、これらが間違った文字に変換されます。

ティップ
recode(1) は、十分使えますし、iconv(1) と fromdos(1) と todos(1) と frommac(1) と tomac(1) を組み合わせ以上の機能を提供します。詳しくは”info recode” を参照下さい。

11.1.2 ファイルが UTF-8 であると iconv を使い確認

以下のようにするとテキストファイルが UTF-8 でエンコードされていると iconv(1) を使って確認できます。

```
$ iconv -f utf8 -t utf8 input.txt >/dev/null || echo "non-UTF-8 found"
```

ティップ
最初の非 UTF-8 文字を見つけるには上記例中で”--verbose” オプションを使います。

11.1.3 iconv を使ってファイル名変換

次に、単一ディレクトリー中の旧 OS 下で作成されたファイル名から現代的な UTF-8 のファイル名に符号化方式を変換するスクリプト例を示します。

```
#!/bin/sh
ENCDN=iso-8859-1
for x in *;
 do
 mv "$x" "$(echo "$x" | iconv -f $ENCDN -t utf-8)"
done
```

”$ENCDN” 変数値には、旧 OS 下で用いられたファイル名に用いられた元となる表 11.2 中にあるエンコーディングを指定します。

もっと複雑な場合にはそのようなファイル名を含有するファイルシステム (ディスクドライブ上のパーティション等) を mount(8) オプションに適正な符号化方式 (エンコーディング) (項 8.1.3 を参照下さい) を指定してマウントし、その全内容を他の UTF-8 でマウントされたファイルシステムに”cp -a” コマンドを使ってコピーします。

11.1.4 行末変換

テキストファイルのフォーマット、特に行末 (EOL) コード、はプラットフォーム依存です。

行末 (EOL) フォーマット変換プログラムに関して、fromdos(1) と todos(1) (tofrodos パッケージ参照) と frommac(1) と tomac(1) (macutils パッケージ参照) は非常に便利です。recode(1) もまた役に立ちます。

注意
ほとんどのエディター (例えば vim や emacs や gedit 等) は MSDOS スタイルの行末を透過的に取り扱えます。

プラットフォーム	行末コード	コントロール	**10** 進数	**16** 進数
Debian (unix)	LF	^J	10	0A
MSDOS と Windows	CR-LF	^M^J	13 10	0D 0A
Apple の Macintosh	CR	^M	13	0D

Table 11.3: 異なるプラットフォーム上での行末スタイルのリスト

ティップ
MSDOS と Unix スタイルが混在する行末スタイルを MSDOS スタイルに統一するには、todos(1) を使う代わりに”`sed -e '/\r$/!s/$/\r/'`” を使う方がより好ましいです。(例えば、2 つの MSDOS スタイルファイルを diff3(1) を使ってマージした後。) `todos` は全ての行に CR を追加するというのがこの理由です。

11.1.5 タブ変換

タブコードを変換するための良く使われる専用プログラムがいくつかあります。

機能	**bsdmainutils**	**coreutils**
タブからスペースに展開する	”`col -x`”	`expand`
スペースからタブに逆展開する	”`col -h`”	`unexpand`

Table 11.4: `bsdmainutils` と `coreutils` パッケージ中のタブ変換コマンドのリスト

`indent` パッケージにある indent(1) コマンドは C プログラム中のホワイトスペースを完全にリフォーマットします。

`vim` や `emacs` 等のエディタープログラムもまたタブ変換に使えます。例えば `vim` を使うと、”`:set expandtab`” として”`:%retab`” するコマンドシーケンスでタブ変換が出来ます。これを元に戻すのは、”`:set noexpandtab`” として”`:%retab!`” とするコマンドシーケンスです。

11.1.6 自動変換付きエディター

`vim` プログラムなどのインテリジェントな現代的なエディターは大変良く出来ていていかなる符号化方式やいかなるファイルフォーマットでも機能します。これらのエディターを UTF-8 ロケール下で UTF-8 を扱えるコンソール中で使用することで最良の互換性が得られます。

latin1 (iso-8859-1) 符号化方式で保存された古い西欧州の Unix テキストファイル”`u-file.txt`” は、単純に `vim` を使って以下のようにして編集出来ます。

```
$ vim u-file.txt
```

`vim` 中の符号化方式自動判定機構が、最初は UTF-8 符号化方式を仮定し、それが上手く行かなかった際に latin1 を仮定するから可能です。

latin2 (iso-8859-2) 符号化方式で保存された古いポーランドの Unix テキストファイル”`pu-file.txt`” は、`vim` を使って以下のようにして編集出来ます。

```
$ vim '+e ++enc=latin2 pu-file.txt'
```

eucJP 符号化方式で保存された古い日本の Unix テキストファイル”ju-file.txt” は、vim を使って以下のようにして編集出来ます。

```
$ vim '+e ++enc=eucJP ju-file.txt'
```

shift-JIS 符号化方式 (より正確には: CP932) で保存された古い日本の MS-Windows テキストファイル”jw-file.txt” は、vim を使って以下のようにして編集出来ます。

```
$ vim '+e ++enc=CP932 ++ff=dos jw-file.txt'
```

”++enc” や”++ff” オプションを使ってファイルが開かれた時は、Vim コマンドライン中の”:w” がオリジナルのファイルフォーマットでオリジナルのファイルを上書きします。例えば”:w ++enc=utf8 new.txt” 等と Vim コマンドライン中で保存フォーマットや保存ファイル名を指定することも出来ます。

vim オンラインヘルプ中の mbyte.txt ”multi-byte text support” と、”++enc” に使われるロケール値に関する表 11.2 を参照下さい。

emacs ファミリーのプログラムもまた同様の機能の実行ができます。

11.1.7 プレーンテキスト抽出

以下はウェッブページを読みテキストファイルに落とします。ウェッブから設定を取ってくる時や grep(1) 等の基本的な Unix テキストツールをウェッブページに適用するときに非常に有用です。

```
$ w3m -dump https://www.remote-site.com/help-info.html >textfile
```

同様に、次を用いることで他のフォーマットからプレーンテキストデーターを抽出出来ます。

パッケージ	ポプコン	サイズ	キーワード	機能
w3m	V:11, I:125	2853	html→text	”w3m -dump” コマンドを使う HTML からテキストへの変換ソフト
html2text	V:4, I:66	298	html→text	高度な HTML からテキストへの変換ソフト (ISO 8859-1)
lynx	V:30, I:454	2031	html→text	”lynx -dump” コマンドを使う HTML からテキストへの変換ソフト
elinks	V:2, I:14	1839	html→text	”elinks -dump” コマンドを使う HTML からテキストへの変換ソフト
links	V:2, I:19	2344	html→text	”links -dump” コマンドを使う HTML からテキストへの変換ソフト
links2	V:0.8, I:9.2	5497	html→text	”links2 -dump” コマンドを使う HTML からテキストへの変換ソフト
catdoc	V:16, I:169	682	MSWord→text,TeX	MSWord ファイルをプレーンテキストか TeX に変換
antiword	V:0.9, I:6.1	596	MSWord→text,ps	MSWord ファイルをプレーンテキストか ps に変換
unhtml	V:0.04, I:0.50	40	html→text	HTML ファイルからマークアップタグを削除
odt2txt	V:1, I:17	60	odt→text	OpenDocument テキストからテキストへの変換ソフト

Table 11.5: プレーンテキストデーター抽出ツールのリスト

11.1.8 プレーンテキストデーターをハイライトとフォーマット

以下のようにしてプレーンテキストデーターをハイライトとフォーマット出来ます。

パッケージ	ポプコン	サイズ	キーワード	説明
vim-runtime	V:18, I:347	39473	ハイライト	":source $VIMRUNTIME/syntax/html.vim" を使ってソースコードを HTML に変換するための Vim MACRO
cxref	V:0.02, I:0.20	1191	c→html	C プログラムから latex か HTML への変換ソフト (C 言語)
src2tex	V:0.02, I:0.14	1799	ハイライト	多くのソースコードの TeX への変換ソフト (C 言語)
source-highlight	V:0.4, I:3.0	2131	ハイライト	多くのソースコードを HTML と XHTML と LaTeX と Texinfo と ANSI カラーエスケープシーケンスと DocBook にハイライト付きで変換 (C++)
highlight	V:0.3, I:2.7	1452	ハイライト	多くのソースコードを HTML と XHTML と LaTeX と Tex と AXSL-FO にハイライト付きで変換 (C++)
grc	V:0.9, I:5.4	208	text→color	汎用着色化ソフト (Python)
pandoc	V:12, I:48	209559	text→any	汎用マークアップコンバーター (Haskel)
python3-docutils	V:13, I:51	2012	text→any	ReStructured テキスト文書の XML へのフォーマット化ソフト (Python)
markdown	V:0.4, I:5.5	56	text→html	Markdown テキスト文書の (X)HTML へのフォーマット化ソフト (Perl)
asciidoctor	V:0.4, I:4.4	101	text→any	AsciiDoc テキスト文書の XML/HTML へのフォーマット化ソフト (Ruby)
python3-sphinx	V:7, I:27	3245	text→any	ReStructured テキストを使う文書出版システム (Python)
hugo	V:0.8, I:5.0	66919	text→html	Markdown テキストを使うサイト出版システム (Go)

Table 11.6: プレーンテキストデーターをハイライトするツールのリスト

11.2 XML データー

Extensible Markup Language (XML) は構造化情報を含む文書のためのマークアップ言語です。

XML.COM にある入門情報を参照下さい。

- ”What is XML?”
- ”What Is XSLT?”
- ”What Is XSL-FO?”
- ”What Is XLink?”

11.2.1 XML に関する基本ヒント

XML テキストはちょっと HTML のようにも見えます。これを使うと一つの文書から複数のフォーマットのアウトプット管理できるようになります。簡単な XML システムの一つはここで使っている `docbook-xsl` パッケージです。

各 XML ファイルは以下のような標準的な XML 宣言でスタートします。

```
<?xml version="1.0" encoding="UTF-8"?>
```

XML 要素の基本的シンタックスは以下のようにマークアップされます。

```
<name attribute="value">content</name>
```

空の XML 要素は以下の短縮形を使ってマークアップされます。

```
<name attribute="value" />
```

上記例中の”`attribute="value"`” はオプションです。

XML 中のコメントセクションは以下のようにマークアップされます。

```
<!-- comment -->
```

マークアップを追加する以外に、XML は以下の文字に関して事前定義されたエンティティを使い内容を少し改変する必要があります。

事前定義されたエンティティ	変換先の文字
`"`	`"`：引用符
`'`	`'`：アポストロフィ
`<`	`<`：以下
`>`	`>`：以上
`&`	`&`：アンパサンド

Table 11.7: XML で事前定義されているエントリーのリスト



注意
”<” と”&” はアトリビュートやエレメントには使えません。

注意
例えば”&some-tag;” 等の SGML スタイルのユーザー定義エンティティが使われた場合、他の定義は無効で最初の定義が有効です。エンティティ定義は”<!ENTITY some-tag "entity value">” と表現されます。

注意
XML のマークアップがタグ名の何らかの組み合わせで (あるデーターを内容としてであれアトリビュート値としてであれ) 整合性を持ってされている限り、他の XML の変換は拡張可能スタイルシート言語変換 (XSLT) を使うととっても簡単な作業です。

11.2.2 XML 処理

拡張可能スタイルシート言語 (XSL) のような XML ファイルを処理に利用可能なツールは沢山あります。

基本的に、良くできた XML ファイルを一度作ると、いかなるフォーマットへも拡張可能なスタイルシート言語変換 (XSLT) を使って変換できます。

フォーマットオブジェクト用拡張可能スタイルシート言語 (XSL-FO) がフォーマットのための答えとなるはずです。fop パッケージは Java プログラム言語に依存するため Debian の main アーカイブでは新規です。このため、LaTeX コードが XML から XSLT を使って通常作成され、DVI や PostScript や PDF 等のプリンタブルなファイルが LaTeX システムを使って作成されます。

パッケージ	ポプコン	サイズ	キーワード	説明
docbook-xml	V:15, I:403	2126	xml	DocBook 用 XML ドキュメントタイプ定義 (DTD)
docbook-xsl	V:15, I:141	14823	xml/xslt	DocBook XML を XSLT を使って各種アウトプットへ処理する XSL スタイルシート
xsltproc	V:15, I:69	83	xslt	XSLT コマンドラインプロセスソフト (XML→ XML, HTML, plain text, 他)
xmlto	V:0.5, I:7.7	124	xml/xslt	XSLT を用いて XML から全てへの変換ソフト
fop	V:0.6, I:7.6	282	xml/xsl-fo	Docbook XML ファイルを PDF に変換
dblatex	V:0.9, I:5.3	4635	xml/xslt	XSLT を使って Docbook ファイルを DVI, PostScript, PDF 文書へ変換
dbtoepub	V:0.04, I:0.52	37	xml/xslt	DocBook XML から.epub へのコンバーター

Table 11.8: XML ツールのリスト

XML は標準一般化マークアップ言語 (SGML) のサブセットなので、ドキュメントスタイル構文規程言語 (DSSSL) 等の SGML 用として利用可能な広範なツールで処理できます。

ティップ
GNOME の yelp は DocBook XML ファイルを X 上に体裁良く表示するので時々便利です。

11.2.3 XML データー抽出

他のフォーマットから以下を使うと HTML とか XML のデーターを抽出出来ます。

パッケージ	ポプコン	サイズ	キーワード	説明
openjade	V:1, I:20	1066	dsssl	ISO/IEC 10179: 1996 標準 DSSSL プロセッサ (最新版)
docbook-dsssl	V:0.5, I:7.2	2594	xml/dsssl	DocBook XML を各種出力フォーマットに DSSSL を使って処理するための DSSSL スタイルシート
docbook-utils	V:0.3, I:5.1	287	xml/dsssl	docbook2* コマンドで DSSSL を使って DocBook ファイルを他のフォーマットに (HTML, RTF, PS, man, PDF) 変換するなどのユーティリティー

Table 11.9: DSSL ツールのリスト

パッケージ	ポプコン	サイズ	キーワード	説明
man2html	V:0.1, I:1.2	142	manpage→html	manpage から HTML への変換ソフト (CGI サポート)
doclifter	V:0.00, I:0.05	487	troff→xml	troff から DocBook XML への変換ソフト
texi2html	V:0.2, I:2.6	1847	texi→html	Texinfo から HTML への変換ソフト
info2www	V:0.8, I:1.4	76	info→html	GNU info から HTML への変換ソフト (CGI サポート)
wv	V:0.2, I:2.2	733	MSWord→any	Microsoft Word から HTML や LaTeX 等への文書変換ソフト
unrtf	V:0.3, I:2.7	158	rtf→html	RTF から HTML 等への文書変換ソフト
wp2x	V:0.00, I:0.07	200	WordPerfect→any	WordPerfect 5.0 と 5.1 ファイルから TeX と LaTeX と troff と GML と HTML への変換ソフト

Table 11.10: テキストデーター変換ツールのリスト

11.2.4 XML データーの静的解析

非 XML の HTML ファイルの場合は、これらを整合性ある XML である XHTML に変換できます。XHTML は XML ツールで処理できます。

XML ファイルのシンタックスやファイル中で見かける URL の適正性が確認されるかもしれません。

パッケージ	ポプコン	サイズ	機能	説明
libxml2-utils	V:64, I:204	214	xml↔html↔xhtml	xmllint(1) (シンタクスチェック、リフォーマット、静的解析、他) を含むコマンドライン XML ツール
tidy	V:0.8, I:6.7	79	xml↔html↔xhtml	HTML シンタックスチェックソフトとリフォーマットソフト
weblint-perl	V:0.05, I:0.85	32	静的解析 (lint)	HTML 用のシンタックス最小限の文体チェックソフト
linklint	V:0.05, I:0.45	343	リンクチェック	高速リンクチェックソフトとウェッブサイトメンテツール

Table 11.11: XML 整形印刷ツールのリスト

一度適正な XML が生成されれば、XSLT 技術を使ってマークアップコンテキスト等に基づいてデーターを抽出出来ます。

11.3 タイプセッティング

Unix の troff プログラムは最初 AT&T で開発されました。それはマンページを作成するのに通常使われます。

Donald Knuth 氏によって作成された TeX は非常に強力な組版ツールでデファクト標準です。最初 Leslie Lamport 氏によって書かれた LaTeX は TeX の力への高レベルアクセスを可能にします。

パッケージ	ポプコン	サイズ	キーワード	説明
texlive	V:1, I:26	55	(La)TeX	組版、校正、印刷のための TeX システム
lilypond	V:0.7, I:5.1	18010	(La)TeX	音楽用タイプセッター
groff	V:2, I:22	16868	troff	GNU troff テキストフォーマティングシステム

Table 11.12: タイプ設定ツールのリスト

11.3.1 roff タイプセッティング

伝統的には、roff が主な Unix テキスト処理システムです。roff(7) と groff(7) と groff(1) と grotty(1) と troff(1) と groff_mdoc(7) と groff_man(7) と groff_ms(7) と groff_me(7) と groff_mm(7) と"`info groff`" を参照下さい。

`groff` パッケージをインストールすると"`/usr/share/doc/groff/`" 中に"`-me`" マクロに関する良い入門書や参考書が読めます。

ティップ
"`groff -Tascii -me -`" は ANSI エスケープコードを含むプレーンテキストを生成します。もしマンページのような多くの"^H" や"_" を含む出力が欲しい場合には、この代わりに"`GROFF_NO_SGR=1 groff -Tascii -me -`" を使います。

ティップ
`groff` が生成した”^H” や”_” をテキストから削除するには、それを”`col -b -x`” でフィルターします。

11.3.2 TeX/LaTeX

TeX Live ソフトウェアーディストリビューションは完全な TeX システムを提供します。`texlive` メタパッケージは、ほとんどの一般的タスクに十分な TeX Live パッケージのまともな選択を提供します。

TeX と LaTeX に関する多くの参考書が利用可能です。

- The teTeX HOWTO: The Linux-teTeX Local Guide
- tex(1)
- latex(1)
- texdoc(1)
- texdoctk(1)
- ”The TeXbook”、Donald E. Knuth 著 (Addison-Wesley)
- ”LaTeX - A Document Preparation System”、Leslie Lamport 著 (Addison-Wesley)
- ”The LaTeX Companion”、Goossens と Mittelbach と Samarin 著 (Addison-Wesley)

これはもっとも強力な組版環境です。多くの SGML 処理ソフトはこれをバックエンドのテキスト処理ソフトとしています。多くの人が Emacs や Vim をソースのエディターとして使う一方、`lyx` パッケージが提供する Lyx と `texmacs` パッケージが提供する GNU TeXmacs は洒落た LaTeX の WYSIWYG 編集環境を提供します。

多くのオンラインリソースが利用可能です。

- TEX Live ガイド (”`/usr/share/doc/texlive-doc/texlive/texlive-en/texlive-en.html`”) (`texlive-base` パッケージ)
- A Simple Guide to Latex/Lyx
- Word Processing Using LaTeX

文書が大きくなると、TeX はエラーを発生する事があります。この問題の解決には (正しくは”`/etc/texmf/texmf.d/95No`
を編集し update-texmf(8) を実行することで) ”`/etc/texmf/texmf.cnf`” 中のプールの数を増やし修正しなければいけません。

注意
”The TeXbook” の TeX ソースは www.ctan.org tex-archive site for texbook.tex にあります。このファイルには必要なマクロのほとんど全てが含まれます。この文書は 7 から 10 行をコメントして”`\input manmac \proofmodefalse`” を追加すると tex(1) で処理できると聞いた事があります。オンラインバージョンを使うのではなくこの本 (さらに Donald E. Knuth 氏による全ての本) を購入される事を強く勧めます。しかし、そのソースは TeX の入力の非常に良い例です！

11.3.3 マニュアルページを綺麗に印刷

以下のコマンドでマンページを PostScript で上手く印刷できます。

```
$ man -Tps some_manpage | lpr
```

11.3.4 マニュアルページの作成

プレーンな troff フォーマットでマンページ (マニュアルページ) を書く事は可能ですが、それを作成するヘルパーパッケージがあります。

パッケージ	ポプコン	サイズ	キーワード	説明
docbook-to-man	V:0.6, I:5.2	189	SGML→manpage	DocBook SGML から roff man マクロへの変換ソフト
help2man	V:0.5, I:6.0	542	text→manpage	--help からの自動マンページ生成ソフト
info2man	V:0.01, I:0.18	134	info→manpage	GNU info から POD かマンページへの変換ソフト
txt2man	V:0.05, I:0.59	112	text→manpage	ベタの ASCII テキストからマンページ形式へ変換

Table 11.13: マンページ作成を補助するパッケージのリスト

11.4 印刷可能データー

Debian システム上では印刷可能なデーターは PostScript フォーマットで表現されます。共通 Unix 印刷システム (CUPS) は非 PostScript プリンタ用のラスタ化のバックエンドプログラムとして Ghostscript を使用します。

最近の Debian システム上では印刷可能なデータは PDF フォーマットででも表現されます。

Evince や Okular (項 7.4 参照) のような GUI ビューワーツールや; Chromium のような現代的なブラウザーを使うと、PDF ファイルの内容表示をしたりその入力欄を埋めたりできます。

LibreOffice や Scribus や Inkscape (項 11.6 参照) のようなグラフィックツールを使うと、PDF ファイルの内容を編集できます。

ティップ
PDF ファイルは、GIMP を用いると、解像度 300 dpi 以上を使い PNG フォーマットに変換し読み込めます。これを、LibreOffice のバックグラウンド画像に用いれば最小限の努力で望ましい変更済み印刷が作れます。

11.4.1 Ghostscript

印刷データー処理の核心はラスタ画像を生成する Ghostscript という PostScript (PS) インタープリタです。

ティップ
”gs -h” とすると Ghostscript の設定が表示されます。

11.4.2 2 つの PS や PDF ファイルをマージ

2 つの PostScript (PS) や Portable Document Format (PDF) ファイルは Ghostscript の gs(1) をつかってマージできます。

```
$ gs -q -dNOPAUSE -dBATCH -sDEVICE=pswrite -sOutputFile=bla.ps -f foo1.ps foo2.ps
$ gs -q -dNOPAUSE -dBATCH -sDEVICE=pdfwrite -sOutputFile=bla.pdf -f foo1.pdf foo2.pdf
```

パッケージ	ポプコン	サイズ	説明
ghostscript	V:180, I:554	177	GPL 版 Ghostscript PostScript/PDF インタープリタ
ghostscript-x	V:0, I:12	88	GPL 版 Ghostscript PostScript/PDF インタープリタ - X ディスプレーサポート
libpoppler162	V:16, I:24	5321	xpdf PDF ビューワー派生 PDF レンダリングライブラリー
libpoppler-glib8t64	V:80, I:335	612	PDF レンダリングライブラリー (GLib 準拠共有ライブラリー)
poppler-data	V:170, I:578	13086	PDF レンダリングライブラリー用 CMaps (CJK サポート: Adobe-*)

Table 11.14: Ghostscript PostScript インタープリタのリスト

注意
PDF は、クロスプラットフォームの印刷可能フォーマットとして広範に使われていて、本質的にいくつかの追加機能と拡張がされている、圧縮 PS フォーマットです。

ティップ
コマンドラインの場合、psutils パッケージ中の psmerge(1) 等のコマンドは PostScript 文書を操作するのに便利です。pdftk-java パッケージの pdftk(1) も PDF 文書を操作するのに便利です。

11.4.3 印刷可能データーユーティリティー

印刷可能なデーターに用いる以下のパッケージが著者の目に止まりました。

11.4.4 CUPS を使って印刷

Common Unix Printing System (CUPS) が提供する、lp(1) と lpr(1) コマンドの両方が印刷可能なデーターの印刷をカスタム化するオプションを提供します。

以下のコマンドの内のひとつを使い一つのファイルに対し 3 部の印刷をページ順に揃えてできます。

```
$ lp -n 3 -o Collate=True filename
```

```
$ lpr -#3 -o Collate=True filename
```

さらに、コマンドライン印刷とオプションに書かれているように"-o number-up=2" や"-o page-set=even", "-o page-set=odd" や"-o scaling=200" や"-o natural-scaling=200" 等の印刷オプションを使ってカスタム化できます。

11.5 メールデーター変換

テキストデーター変換のための以下のパッケージが著者の目に止まりました。

ティップ
インターネットメッセージアクセスプロトコルバージョン 4 (IMAP4) サーバーは、プロプライエタリメールシステムのクライアントソフトが IMAP4 サーバーも使えるように設定できる場合、プロプライエタリメールシステムからメールを取り出すのに利用できるかもしれません。

パッケージ	ポプコン	サイズ	キーワード	説明
poppler-utils	V:175, I:475	804	pdf→ps,text…	PDF ユーティリティー: pdftops, pdfinfo, pdfimages, pdftotext, pdffonts
psutils	V:4, I:47	34	ps→ps	PostScript 文書変換ツール
poster	V:0.1, I:1.5	57	ps→ps	PostScript ページから大きなポスターを作る
enscript	V:1, I:12	2138	text→ps, html, rtf	ASCII テキストから PostScript か HTML か RTF か Pretty-Print への変換
a2ps	V:0.6, I:6.5	4109	text→ps	全てを PostScript に変換するソフトと綺麗印刷ソフト
pdftk	V:1, I:18	28	pdf→pdf	PDF 文書変換ツール: pdftk
html2ps	V:0.1, I:1.7	256	html→ps	HTML から PostScript への変換ソフト
gnuhtml2latex	V:0.04, I:0.55	26	html→latex	html から latex への変換ソフト
latex2rtf	V:0.1, I:1.9	495	latex→rtf	LaTeX から MS Word で読める RTF へと文書変換
ps2eps	V:2, I:30	95	ps→eps	PostScript から EPS (カプセル化済み PostScript) への変換ソフト
e2ps	V:0.01, I:0.10	104	text→ps	日本語符号化サポート付きの Text から PostScript への変換ソフト
impose+	V:0.1, I:1.5	118	ps→ps	PostScript ユーティリティー
trueprint	I:0.07	148	text→ps	多くのソースコード (C, C++, Java, Pascal, Perl, Pike, Sh, Verilog) の PostScript への綺麗印刷 (C 言語)
pdf2svg	V:0.2, I:2.8	33	pdf→svg	PDF からスケール可のベクトルグラフィクス (SVG) フォーマットへの変換ソフト
pdftoipe	V:0.02, I:0.44	73	pdf→ipe	PDF から IPE の XML フォーマットへの変換ソフト

Table 11.15: プリントできるデーターのユーティリティーのリスト

パッケージ	ポプコン	サイズ	キーワード	説明
sharutils	V:2, I:26	1436	メール	shar(1) と unshar(1) と uuencode(1) と uudecode(1)
mpack	V:0.7, I:7.1	109	MIME	MIME メッセージの符号化と逆符号化のソフト: mpack(1) と munpack(1)
tnef	V:0.3, I:3.7	103	ms-tnef	Microsoft のみのフォーマットの”application/ms-tnef” タイプの MIME アタッチメントを開梱
uudeview	V:0.2, I:1.6	105	mail	以下のフォーマットのエンコーダーとデコーダー: uuencode, xxencode, BASE64, quoted printable, BinHex

Table 11.16: メールデーター変換を補助するパッケージのリスト

11.5.1 メールデーターの基本

メイル (SMTP) データーは 7 ビットデーター列に限定されるべきです。だからバイナリーデーターや 8 ビットテキストデーターは Multipurpose Internet Mail Extensions (MIME) を用いたり文字セット (表 11.2 を参照下さい) を選択して 7 ビットのフォーマットにエンコードされます。

標準のメールストレージフォーマットは RFC2822 (RFC822 の更新版) により定義される mbox フォーマットです。mbox(5) (`mutt` パッケージが提供) を参照下さい。

欧州言語の場合､ほとんど 8 ビット文字が無いので ISO-8859-1 文字セットとともに"`Content-Transfer-Encoding: quoted-printable`" がメールに通常使われます。欧州のテキストが UTF-8 符号化された場合、ほとんどが 7 ビット文字なので"`Content-Transfer-Encoding: quoted-printable`" が大体使われます。

日本語には､テキストを 7 ビットにしておくために伝統的に"`Content-Type: text/plain; charset=ISO-2022-JP`" がメールに通常使われます。しかし、古い Microsoft システムは適正な宣言無しに Shift-JIS でメールデーターを送るかもしれません。日本語のテキストが UTF-8 で符号化される場合、多くの 8 ビットデーターを含むので Base64 が大体使われます。他のアジアの言語でも状況は同様です。

注意
もし IMAP4 サーバーと話せる非 Debian クライアントからあなたの非 Unix メールデーターがアクセス出きるなら、あなた自身の IMAP4 サーバーを実行することでメールデーターを引き出せるかもしれません。

注意
もし他のメールストレージフォーマットを使っている場合、mbox フォーマットに移動するのが良い第一歩です。mutt(1) のような汎用クライアントプログラムはこれに非常に便利です。

メールボックスの内容は procmail(1) と formail(1) を使って各メッセージに分割できます。

各メールメッセージは `mpack` パッケージにある munpack(1) (または他の専用ツール) を使って開梱して MIME 符号化された内容を取り出せます。

11.6 画像データーツール

gimp(1) のような GUI プログラムは非常に強力ですが、imagemagick(1) 等のコマンドラインツールはスクリプトでイメージ操作を自動化するのに非常に便利です。

デジタルカメラのファイルフォーマットのデファクト標準は、追加のメタデーター付きの JPEG 画像ファイルフォーマットである交換可能な画像ファイルフォーマット (EXIF) です。EXIF は日付や時間やカメラ設定等の情報を保持できます。

Lempel-Ziv-Welch (LZW) ロス無しデーター圧縮特許の期限は切れました。LZW データー圧縮を使う画像交換フォーマット (GIF) ユーティリティーは Debian システム上で自由に利用可能となりました。

ティップ
リムーバブル記録メディア付きのどのデジタルカメラやスキャナーも、カメラファイルシステム用デザインルールに準拠し FAT ファイルシステムを使っているので USB ストレージ読取り機を経由すれば Linux で必ず機能します。項 10.1.7 を参照下さい。

11.6.1 画像データーツール (メタパッケージ)

以下のメタパッケージは aptitude(8) を使って画像データーツールを探す良いスタート地点です。"Packages overview for Debian PhotoTools Maintainers" も別のスタート地点です。

パッケージ	ポプコン	サイズ	キーワード	説明
education-graphics	0.30	26	svg,jpeg,…	画像や絵画芸術教育用のメタパッケージ
open-font-design-toolkit	0.04	9	ttf,ps,…	オープンフォントデザイン用のメタパッケージ

Table 11.17: 画像データーツールのリスト (メタパッケージ)

ティップ
aptitude(8) の正規表現”~Gworks-with::image” (項 2.2.6 を参照下さい) を使ってさらなる画像ツールを探します。

11.6.2 画像データーツール (GUI)

以下の GUI の画像データー変換、編集、整理用パッケージが著者の目に止まりました。

11.6.3 画像データーツール (CLI)

以下の CLI の画像データー変換、編集、整理用パッケージが著者の目に止まりました。

11.7 その他のデーター変換

多くのデーター変換プログラムがあります。aptitude(8) で”~Guse::converting” という正規表現 (項 2.2.6 を参照下さい) を使い以下のプログラムが私の目に止まりました。

RPM フォーマットからのデーター抽出もまた以下のようにするとできます。

```
$ rpm2cpio file.src.rpm | cpio --extract
```

パッケージ	ポプコン	サイズ	キーワード	説明
gimp	V:40, I:206	33386	image(bitmap)	GNU イメージ操作プログラム
xsane	V:9, I:127	1512	image(bitmap)	GTK に基づく SANE (Scanner Access Now Easy) 用の X11 フロントエンド
scribus	V:1, I:12	32942	ps/pdf/SVG/…	Scribus DTP エディター
libreoffice-draw	V:78, I:412	11424	image(vector)	LibreOffice office スイート - ドロー
inkscape	V:12, I:73	113357	image(vector)	SVG (スケーラブルベクトルグラフィクス) エディター
dia	V:1, I:16	3846	image(vector)	ダイアグラムエディター (Gtk)
xfig	V:0.5, I:8.4	7951	image(vector)	X11 下でインテラクティブ生成するソフト
gocr	V:0.4, I:3.4	549	image→text	フリー OCR ソフト
eog	V:20, I:121	10542	image(Exif)	画像ビューアープログラム Eye of GNOME
gthumb	V:3, I:11	5131	image(Exif)	画像ビューアー兼ブラウザー (GNOME)
geeqie	V:3, I:10	3854	image(Exif)	GTK を用いた画像ビューアー
shotwell	V:15, I:244	6334	image(Exif)	デジタル写真オーガナイザー (GNOME)
gwenview	V:40, I:113	6051	image(Exif)	画像ビューア (KDE)
kamera	I:112	992	image(Exif)	KDE アプリケーション用デジタルカメラサポート
digikam	V:1.5, I:8.2	333	image(Exif)	デジタル写真管理アプリケーション
darktable	V:3, I:10	36092	image(Exif)	写真家のための仮想ライトボックスと暗室
hugin	V:0.5, I:5.2	6885	image(Exif)	パノラマ写真合成機
librecad	V:1, I:13	12361	DXF,…	2D CAD データーエディター
freecad	V:1, I:20	112	DXF,…	3D CAD データーエディター
blender	V:3, I:20	110904	blend,TIFF,VRML,…	バージョン等用の 3D コンテントエディター
mm3d	V:0.03, I:0.23	4123	ms3d,obj,dxf	OpenGL 準拠の 3D モデルエディター
fontforge	V:0.5, I:5.2	4736	ttf,ps,…	PS と TrueType と OpenType のフォント用フォントエディター
xgridfit	V:0.02, I:0.07	878	ttf	TrueType フォントをグリッドフィッティングとヒンティング用のプログラム

Table 11.18: 画像データーツール (GUI) のリスト

パッケージ	ポプコン	サイズ	キーワード	説明
imagemagick	V:7, I:266	82	image(bitmap)	画像操作プログラム
graphicsmagick	V:1.0, I:7.8	5988	image(bitmap)	画像操作プログラム (imagemagick のフォーク)
netpbm	V:27, I:275	8470	image(bitmap)	画像変換ツール
libheif-examples	V:0.4, I:3.7	534	heif→jpeg(JPEG)	High Efficiency Image File Format (HEIF) を JPEG や PNG や Y4M フォーマットに heif-convert(1) コマンドで変換
icoutils	V:3, I:31	221	png↔ico(bitmap)	MS Windows のアイコンやカーソルと PNG フォーマット間の変換 (favicon.ico)
pstoedit	V:2, I:36	1075	ps/pdf→image(vector)	PostScript と PDF ファイルから編集可能なベクトルグラフィクスへの変換ソフト (SVG)
libwmf-bin	V:5, I:77	149	Windows/画像 (vector)	Windows メタファイル (ベクトル画像データー) 変換ツール
fig2sxd	V:0.02, I:0.14	158	fig→sxd(vector)	XFig ファイルを OpenOffice.org Draw フォーマットに変換
unpaper	V:2, I:16	417	image→image	OCR 用のスキャンしたページの後処理ツール
tesseract-ocr	V:9, I:35	2411	image→text	HP の商用 OCR エンジンの基づくフリーの OCR ソフトウェアー
tesseract-ocr-eng	V:9, I:35	4032	image→text	OCR エンジンデーター: tesseract-ocr の英文用言語ファイル
ocrad	V:0.3, I:2.2	608	image→text	フリー OCR ソフト
exif	V:2, I:50	334	image(Exif)	JPEG ファイル中の EXIF 情報を表示するコマンドラインユーティリティー
exiv2	V:1, I:18	436	image(Exif)	EXIF/IPTC メタデーター操作ツール
exiftran	V:1, I:11	81	image(Exif)	デジタルカメラの jpeg 画像を変換
exiftags	V:0.2, I:2.5	309	image(Exif)	デジタルカメラの JPEG ファイルから Exif タグを読むユーティリティー
exifprobe	V:0.2, I:2.0	506	image(Exif)	デジタル写真からメタデーターを読み出す
dcraw	V:0.7, I:7.0	428	image(Raw)→ppm	生のデジタルカメラ画像のデコード
findimagedupes	V:0.10, I:0.94	76	image→fingerprint	視覚的に類似画像と重複画像の検出
ale	V:0.01, I:0.14	850	image→image	忠実度を上げたりモザイクを作成するための画像のマージ
imageindex	V:0.1, I:1.1	143	image(Exif)→html	イメージから静的な HTML ギャラリーを生成
outguess	V:0.1, I:1.0	230	jpeg,png	普遍的 Steganographic ツール
jpegoptim	V:0.5, I:5.6	59	jpeg	JPEG ファイルの最適化
optipng	V:2, I:38	187	png	PNG ファイルのロスレス最適化
pngquant	V:1, I:11	62	png	PNG ファイルのロッシー最適化

Table 11.19: 画像データーツールのリスト (CLI)

パッケージ	ポプコン	サイズ	キーワード	説明
alien	V:1, I:12	150	rpm/tgz→deb	外来のパッケージの Debian パッケージへの変換ソフト
freepwing	I:0.01	447	EB→EPWING	"Electric Book" (日本で人気) から単一の JIS X 4081 フォーマット (EPWING V1 のサブセット) への変換ソフト
calibre	V:7, I:24	76289	any→EPUB	e-book コンバーターとライブラリーの管理

Table 11.20: その他のデーター変換ツールのリスト

Chapter 12

プログラミング

Debian システム上でプログラミングを学ぶ人がパッケージ化されたソースコードを読み込めるようになるための指針を示します。以下はプログラムに関する特記すべきパッケージと対応する文書パッケージです。

オンラインリファレンスは `manpages` と `manpages-dev` パッケージをインストールした後で”`man name`” とタイプすると利用可能です。GNU ツールのオンラインリファレンスは該当する文書パッケージをインストールした後で”`info program_name`” とタイプすると使えます。一部の GFDL 文書は DFSG に準拠していないと考えられているので `main` アーカイブに加えて `contrib` や `non-free` アーカイブを含める必要があるかもしれません。

バージョンコントロールシステムツールの使用を考えましょう。項 10.5 を参照下さい。



警告
”`test`” を実行可能なテストファイルの名前に用いてはいけません。”`test`” はシェルのビルトインです。



注意
ソースから直接コンパイルしたソフトウェアープログラムは、システムプログラムとかち合わないように、”`/usr/local`” か”`/opt`” の中にインストールします。

12.1 シェルスクリプト

シェルスクリプトは実行ビットがセットされたテキストファイルで、以下に示すフォーマットのコマンドを含んでいます。

```
#!/bin/sh
 ... command lines
```

最初の行はこのファイル内容を読み実行するシェルインタープリタを指定します。

シェルスクリプトを読むのは Unix 的なシステムがどのように機能しているのかを理解する最良の方法です。ここでは、シェルプログラムに関する指針や心がけを記します。失敗から学ぶために” シェルの失敗” (https://www.greenend.org.uk/rjk/2001/04/shell.html) を参照下さい。

シェル対話モード (項 1.5 と項 1.6 を参照下さい) と異なり、シェルスクリプトは変数や条件文やループを繁用します。

12.1.1 POSIX シェル互換性

多くのシステムスクリプトは POSIX シェル (表 1.13 を参照下さい) のどれで解釈されるか分かりません。

- デフォールトの非インタラクティブなシェル”/usr/bin/sh” は /usr/bin/dash をさしているシムリンクで、多くのシステムプログラムで使われます。
- デフォルトのインタラクティブなシェルは /usr/bin/bash です。

全ての POSIX シェル間でポータブルとするために **bashisms** や **zshisms** を使うシェルスクリプトを書くのを避けます。checkbashisms(1) を使うとこれがチェックできます。

推薦: **POSIX**	回避すべき: **bashism**
`if [ "$foo" = "$bar" ] ; then …`	`if [ "$foo" == "$bar" ] ; then …`
`diff -u file.c.orig file.c`	`diff -u file.c{.orig,}`
`mkdir /foobar /foobaz`	`mkdir /foo{bar,baz}`
`funcname() { …}`	`function funcname() { …}`
8 進表記: ”\377”	16 進表記: ”\xff”

Table 12.1: 典型的 bashizms のリスト

”echo” コマンドはその実装がシェルビルトインや外部コマンド間で相違しているので以下の注意点を守って使わなければいけません。

- ”-n” 以外のどのコマンドオプション使用も避けます。
- 文字列中にエスケープシーケンスはその取扱いに相違があるので使用を避けます。

注意
”-n” オプションは実は POSIX シンタックスではありませんが、一般的に許容されています。

ティップ
出力文字列にエスケープシーケンスを埋め込む必要がある場合には、”echo” コマンドの代わりに”printf” コマンドを使います。

12.1.2 シェル変数

特別なシェルパラメーターがシェルスクリプト中ではよく使われます。

覚えておくべき基本的なパラメーター展開を次に記します。

ここで、これら全てのオペレーターのコロン”:” は実際はオプションです。

- ”:” 付き = 演算子は存在と非ヌル文字列をテストします
- ”:” 無し = 演算子は存在のみをテストします

シェル変数	変数値
$0	シェルまたはシェルスクリプトの名前
$1	最初 (1 番目) のシェル引数
$9	9 番目のシェル引数
$#	シェル引数の数
"$*"	"$1 $2 $3 $4 …"
"$@"	"$1" "$2" "$3" "$4" …
$?	最新のコマンドの終了状態
$$	このシェルスクリプトの PID
$!	最近スタートしたバックグラウンドジョブの PID

Table 12.2: シェル変数のリスト

パラメーター式形	**var** が設定されているときの値	**var** が設定されていないときの値
${var:-string}	”$var”	”string”
${var:+string}	”string”	”null”
${var:=string}	”$var”	”string” (合わせて”var=string” を実行)
${var:?string}	”$var”	”string” を **stderr** に出力 (エラーとともに exit する)

Table 12.3: シェル変数展開のリスト

パラメーター置換形	結果
${var%suffix}	最短のサフィックスパターンを削除
${var%%suffix}	最長のサフィックスパターンを削除
${var#prefix}	最短のプレフィックスパターンを削除
${var##prefix}	最長のプレフィックスパターンを削除

Table 12.4: 重要なシェル変数置換のリスト

12.1.3 シェル条件式

各コマンドは条件式に使えるエグジットステイタスを返します。

- 成功: 0 (” 真”)
- エラー: 非 0 (” 偽”)

注意
シェル条件文の文脈において”0” は” 真” を意味しますが、C 条件文の文脈では”0” は” 偽” を意味します。

注意
”[” は、”]” までの引数を条件式として評価する、test コマンドと等価です。

覚えておくべき基本的な条件文の慣用句は次です。

- ”*command* && 成功したらこの *command* も実行 || true”
- ”*command* || もし *command* が成功しないとこのコマンドも実行 || true”
- 以下のような複数行のスクリプト断片

```
if [ conditional_expression ]; then
 if_success_run_this_command
else
 if_not_success_run_this_command
fi
```

ここで、シェルが”-e” フラグ付きで起動された際にシェルスクリプトがこの行で誤って exit しないようにするために、末尾の”|| true” が必要です。

式	論理真を返す条件
-e *file*	*file* が存在する
-d *file*	*file* が存在しディレクトリーである
-f *file*	*file* が存在し通常ファイルである
-w *file*	*file* が存在し書込み可能である
-x *file*	*file* が存在し実行可能である
file1 -nt *file2*	*file1* が *file2* よりも新しい (変更)
file1 -ot *file2*	*file1* が *file2* よりも古い (変更)
file1 -ef *file2*	*file1* と *file2* は同デバイス上の同 inode 番号

Table 12.5: 条件式中のファイル比較演算子

条件式中の算術整数比較演算子は”-eq” と”-ne” と”-lt” と”-le” と”-gt” と”-ge” です。

12.1.4 シェルループ

POSIX シェル中で使われるループの慣用句があります。

- ”for x in foo1 foo2 …; do コマンド; done” は”foo1 foo2 …” リストの項目を変数”x” に代入し” コマンド” を実行してループします。

式	論理真を返す条件
`-z str`	*str* の長さがゼロ
`-n str`	*str* の長さが非ゼロ
`str1 = str2`	*str1* と *str2* は等しい
`str1 != str2`	*str1* と *str2* は等しく無い
`str1 < str2`	*str1* のソート順が *str2* より前 (ロケール依存)
`str1 > str2`	*str1* のソート順が *str2* より後 (ロケール依存)

Table 12.6: 条件式中での文字列比較演算子のリスト

- "`while 条件; do コマンド; done`" は" 条件" が真の場合" コマンド" を繰り返します。
- "`until 条件; do コマンド; done`" は" 条件" が真でない場合" コマンド" を繰り返します。
- "`break`" によってループから脱出できます。
- "`continue`" によってループ初めに戻り次の反復実行を再開します。

ティップ
C 言語のような数字の繰り返しは"`foo1 foo2 ...`" 生成に seq(1) 使って実現します。

ティップ
項 9.4.9 を参照下さい。

12.1.5 シェル環境変数

普通の人気あるシェルコマンドプロンプトがあなたのスクリプト実行環境下使えないかもしれません。

- "`$USER`" には"`$(id -un)`" を使います
- "`$UID`" には"`$(id -u)`" を使います
- "`$HOME`" に関して、"`$(getent passwd "$(id -u)"|cut -d ":" -f 6)`" を使います (これは項 4.5.2 上でも機能します。)

12.1.6 シェルコマンドライン処理シーケンス

シェルはおおよそ以下のシーケンスでスクリプトを処理します。

- シェルは 1 行読み込みます。
- シェルは、もし"…" や'…' の中なら、行の一部を **1** つのトークンとしてグループします。
- シェルは 1 行を以下のによってトークンに分割します。
  - 空白: *space tab newline*
  - メタ文字: `< > | ; & ( )`
- "…" や'…' の中でない場合、シェルは各トークンを予約語に対してチェックしその挙動を調整します。
  - 予約語: `if then elif else fi for in while unless do done case esac`
- "…" や'…' の中でない場合、シェルはエイリアスを展開します。

- "…" や'…' の中でない場合、シェルはティルダを展開します。
  - ”~” → 現ユーザーのホームディレクトリー
  - ”~*user*” → *user* のホームディレクトリー
- '…' の中でない場合、シェルはパラメーター” をその値に展開します。
  - パラメーター: ”$PARAMETER” or ”${PARAMETER}”
- '…' の中でない場合、シェルはコマンド置換を展開します。
  - ”$(command)” → ”command” の出力
  - ”` command `” → ”command” の出力
- "…" や'…' の中でない場合、シェルはパス名のグロブを展開します。
  - * → あらゆる文字
  - ? → 1 文字
  - […] → ”…” 中の 1 つ
- シェルはコマンドを次から検索して実行します。
  - 関数定義
  - ビルトインコマンド
  - ”$PATH” 中の実行ファイル
- シェルは次行に進みこのプロセスを一番上から順に反復します。

ダブルクォート中のシングルクォートに効果はありません。

シェル環境中で”set -x” を実行したり、シェルを”-x” オプションで起動すると、シェルは実行するコマンドを全てプリントするようになります。これはデバグをするのに非常に便利です。

12.1.7 シェルスクリプトのためのユーティリティープログラム

Debian システム上でできるだけポータブルなシェルプログラムとするには、ユーティリティープログラムを **essential** パッケージで提供されるプログラムだけに制約するのが賢明です。

- ”aptitude search ~E” は **essential** (必須) パッケージをリストします。
- ”dpkg -L パッケージ名 |grep '/man/man.*/'” はパッケージ名パッケージによって提供されるコマンドのマンページをリストします。

ティップ
moreutils は Debian の外では存在しないかも知れませんが、興味深い小さなプログラムを提供します。もっとも特記すべきは、オリジナルファイルを上書きしたいときに非常に有効な sponge(8) です。

例は項 1.6 を参照下さい。

パッケージ	ポプコン	サイズ	説明
dash	V:933, I:999	211	sh 用の小型で高速の POSIX 準拠シェル
coreutils	V:914, I:1000	19751	GNU コアユーティリティー
grep	V:800, I:1000	1297	GNU grep, egrep and fgrep
sed	V:789, I:1000	987	GNU sed
mawk	V:517, I:998	299	小さく高速な awk
debianutils	V:940, I:997	228	Debian 特有の雑多なユーティリティー
bsdutils	V:445, I:1000	338	4.4BSD-Lite 由来の基本ユーティリティー
bsdextrautils	V:771, I:877	382	4.4BSD-Lite 由来の追加のユーティリティー
moreutils	V:18, I:38	232	追加の Unix ユーティリティー

Table 12.7: シェルスクリプト用の小さなユーティリティープログラムを含むパッケージのリスト

パッケージ	ポプコン	サイズ	文書
dash	V:933, I:999	211	sh: sh 用の小型高速 POSIX 準拠シェル
bash	V:900, I:1000	7309	sh: bash-doc が提供する"info bash"
mawk	V:517, I:998	299	AWK: 小型高速 awk
gawk	V:246, I:288	3289	AWK: gawk-doc が提供する"info gawk"
perl	V:672, I:991	840	Perl: perl-doc and perl-doc-html が提供する perl(1) と html ページ
libterm-readline-gnu-perl	V:2, I:25	435	GNU ReadLine/History ライブラリーのための Perl 拡張: perlsh(1)
libreply-perl	I:0.08	171	Perl の REPL: reply(1)
libdevel-repl-perl	V:0.02, I:0.47	237	Perl の REPL: re.pl(1)
python3	V:748, I:977	81	Python: python-doc が提供する python3(1) と html ページ
tcl	V:24, I:171	20	Tcl: tcl-doc が提供する tcl(3) と詳細なマンページ
tk	V:18, I:165	20	Tk: tk-doc が提供する tk(3) と詳細なマンページ
ruby	V:61, I:150	32	Ruby: ruby(1), erb(1), irb(1), rdoc(1), ri(1)

Table 12.8: インタープリター関連のパッケージのリスト

12.2 インタープリター言語でのスクリプティング

Debian 上のタスクを自動化したい場合には、まず最初にインタープリター言語でタスクをスクリプト化すべきです。インタープリター言語選択のガイドラインは:

- もしタスクがシェルプログラムでできた CLI プログラムを組み合わせる簡単なタスクなら、`dash` を使います。
- もしタスクが簡単なタスクではなく何もないところから書くなら、`python3` を使います。
- もしタスクをするための加筆をする必要がある `perl`, `tcl`, `ruby`, ... で書かれたコードが Debian 上にすでに存在する場合には、その言語を使います。

もし出来上がったコードがおそすぎる場合には、実行速度にクリチカルな部分のみコンパイラー言語で書き直しインタープリター言語から呼びます。

12.2.1 インタープリター言語コードのデバグ

ほとんどのインタープリターは基本的文法チェックやコード追跡の機能を提供します。

- “**dash -n** *script.sh*”- シェルスクリプトの文法チェック
- “**dash -x** *script.sh*”- シェルスクリプトのトレース
- “**python3 -m py_compile** *script.py*”- Python スクリプトの文法チェック
- “**python3 -mtrace --trace** *script.py*”- Python スクリプトのトレース
- “**perl -I ../libpath -c** *script.pl*”- Perl スクリプトの文法チェック
- “**perl -d:Trace** *script.pl*”- Perl スクリプトのトレース

`dash` のコードチェックの際には、`bash` のようなインタラクティブ環境を用意する項 9.1.4 を試します。

`perl` のコードチェックの際には、Python のような REPL (=READ + EVAL + PRINT + LOOP) 環境を提供する Perl 用の環境を試しましょう。

12.2.2 シェルスクリプトを使った GUI プログラム

シェルスクリプトは魅力的な GUI プログラムを作るまで改善できます。`echo` や `read` コマンドを使う鈍いやりとりに代えて、いわゆるダイアログプログラムを使うのがこのトリックです。

簡単なシェルスクリプトだけでできるほど GUI プログラムがいかに簡単ということを示す GUI プログラムの例は以下です。

このスクリプトはファイルの選択 (デフォルトは`/etc/motd`) に `zenity` を使い、それを表示します。

このスクリプトの GUI ローンチャーは以下のようにして生成できます項 9.4.10.

```
#!/bin/sh -e
# Copyright (C) 2021 Osamu Aoki <osamu@debian.org>, Public Domain
# vim:set sw=2 sts=2 et:
DATA_FILE=$(zenity --file-selection --filename="/etc/motd" --title="Select a file to check ↩
    ") || \
  ( echo "E: File selection error" >&2 ; exit 1 )
# Check size of archive
if ( file -ib "$DATA_FILE" | grep -qe '^text/' ) ; then
  zenity --info --title="Check file: $DATA_FILE" --width 640  --height 400 \
    --text="$(head -n 20 "$DATA_FILE")"
else
  zenity --info --title="Check file: $DATA_FILE" --width 640  --height 400 \
    --text="The data is MIME=$(file -ib "$DATA_FILE")"
fi
```

パッケージ	ポプコン	サイズ	説明
x11-utils	V:233, I:553	651	xmessage(1): window 中にメッセージや質問を表示 (X)
whiptail	V:344, I:997	61	シェルスクリプトからユーザーフレンリーなダイアログボックスを表示 (newt)
dialog	V:8, I:75	535	シェルスクリプトからユーザーフレンリーなダイアログボックスを表示 (ncurses)
zenity	V:69, I:337	193	シェルスクリプトからグラフィカルなダイアログボックスを表示 (GTK)
ssft	V:0.01, I:0.14	80	シェルスクリプトフロントエンドツール (gettext を使った zenity や kdialog や dialog のラッパー)
gettext	V:49, I:206	20549	"/usr/bin/gettext.sh": メッセージ翻訳

Table 12.9: ダイアログプログラムのリスト

シェルスクリプトでの GUI プログラムへのこのようなアプローチは簡単な選択ケースでのみ有効です。何らかの複雑のあるプログラムを書く場合には、もっと能力あるプラットフォームで書くことを考えましょう。

12.2.3 GUI フィルター用のカスタム動作集

GUI ファイラープログラムは、追加の拡張パッケージを使って選択されたファイルに対していくつかの人気ある動作を実行するように拡張できます。また、GUI ファイラープログラムはあなたの特定のスクリプトを追加することで非常に特定のカスタム動作を実行するようにもできます。

- GNOME の場合、NautilusScriptsHowto を参照下さい。
- KDE の場合、Creating Dolphin Service Menus を参照下さい。
- Xfce の場合、Thunar - Custom Actions and https://help.ubuntu.com/community/ThunarCustomActions を参照下さい。
- LXDE の場合、Custom Actions を参照下さい。

12.2.4 究極の短い Perl スクリプト

データー処理をするためには、cut や grep や sed 等を実行するサブプロセスを起動する必要が sh ではあり、遅いです。一方、データーを処理する内部機能が perl ではあり、高速です。そのため、多くの Debian のシステムメンテナンススクリプトは perl を使います。

以下の AWK スクリプトとそれと等価 Perl スクリプトの断片を考えましょう。

```
awk '($2=="1957") { print $3 }' |
```

これは以下の数行のどれとも等価です。

```
perl -ne '@f=split; if ($f[1] eq "1957") { print "$f[2]\n"}' |
```

```
perl -ne 'if ((@f=split)[1] eq "1957") { print "$f[2]\n"}' |
```

```
perl -ne '@f=split; print $f[2] if ( $f[1]==1957 )' |
```

```
perl -lane 'print $F[2] if $F[1] eq "1957"' |
```

```
perl -lane 'print$F[2]if$F[1]eq+1957' |
```

最後のスクリプトは謎々状態です。Perl の以下の機能を利用しています。

- ホワイトスペースはオプション。
- 数字から文字列への自動変換が存在します。
- コマンドラインオプション経由の Perl 実行トリック集: perlrun(1)
- Perl の特別な変数集: perlvar(1)

このフレキシビリティーは Perl の強みです。同時に、これは我々に読解不能な絡まったコードを書くことを許容します。だから注意して下さい。

12.3 コンパイル言語でのコーディング

パッケージ	ポプコン	サイズ	説明
gcc	V:167, I:557	36	GNU C コンパイラ
libc6-dev	V:259, I:577	12863	GNU C ライブラリー: 開発用ライブラリーとヘッダーファイル集
g++	V:64, I:527	13	GNU C++ コンパイラー
libstdc++-16-dev	V:4, I:21	24416	GNU 標準 C++ ライブラリー v3 (開発用ファイル集)
cpp	V:346, I:717	18	GNU C プリプロセッサ
gettext	V:49, I:206	20549	GNU 国際化ユーティリティ
glade	V:0.2, I:2.5	1613	GTK ユーザーインターフェースビルダー
valac	V:0.2, I:2.9	533	GObject システム用の C# に似た言語
flex	V:7, I:66	1247	LEX 互換の高速字句解析ジェネレータ
bison	V:7, I:70	3122	YACC 互換のパーサジェネレータ
susv2	I:0.02	16	"The Single UNIX Specifications v2" を取得
susv3	I:0.04	16	"The Single UNIX Specifications v3" を取得
susv4	I:0.03	16	"The Single UNIX Specifications v4" を取得
golang	I:21	11	Go プログラミング言語コンパイラー
rustc	V:6, I:20	12745	Rust システムプログラム言語
gfortran	V:4, I:47	15	GNU Fortran 95 コンパイラー
fpc	I:2.3	101	Free Pascal

Table 12.10: コンパイラ関連のパッケージのリスト

ここで、項 12.3.3 と項 12.3.4 は、コンパイラーのようなプログラムがどのように高レベル記述を C 言語にすることで C 言語で書かれているかを示すために含めています。

12.3.1 C

C プログラム言語で書かれたプログラムをコンパイルする適正な環境を以下のようにして設定できます。

```
# apt-get install glibc-doc manpages-dev libc6-dev gcc build-essential
```

GNU C ライブラリーパッケージである libc6-dev パッケージは、C プログラム言語で使われるヘッダーファイルやライブラリールーチンの集合である C 標準ライブラリーを提供します。

C のリファレンスは以下を参照下さい。

- ”info libc” (C ライブラリー関数リファレンス)
- gcc(1) と”info gcc”
- each_C_library_function_name(3)
- Kernighan & Ritchie 著, ”The C Programming Language”, 第 2 版 (Prentice Hall)

12.3.2 単純な C プログラム (gcc)

簡単な例の”example.c” は”libm” ライブラリーを使って実行プログラム”run_example” に以下のようにしてコンパイル出来ます。

```
$ cat > example.c << EOF
#include <stdio.h>
#include <math.h>
#include <string.h>

int main(int argc, char **argv, char **envp){
        double x;
        char y[11];
        x=sqrt(argc+7.5);
        strncpy(y, argv[0], 10); /* prevent buffer overflow */
        y[10] = '\0'; /* fill to make sure string ends with '\0' */
        printf("%5i, %5.3f, %10s, %10s\n", argc, x, y, argv[1]);
        return 0;
}
EOF
$ gcc -Wall -g -o run_example example.c -lm
$ ./run_example
        1, 2.915, ./run_exam,     (null)
$ ./run_example 1234567890qwerty
        2, 3.082, ./run_exam, 1234567890qwerty
```

ここで、”-lm” は sqrt(3) のために libc6 パッケージで提供されるライブラリー”/usr/lib/libm.so” をリンクするのに必要です。実際のライブラリーは”/lib/” 中にあるファイル名”libm.so.6” で、それは”libm-2.7.so” にシムリンクされています。

出力テキスト中の最後のパラメーターを良く見ましょう。”%10s” が指定されているにもかかわらず 10 文字以上あります。

上記のオーバーラン効果を悪用するバッファーオーバーフロー攻撃を防止のために、sprintf(3) や strcpy(3) 等の境界チェック無しのポインターメモリー操作関数の使用は非推奨です。これに代えて snprintf(3) や strncpy(3) を使います。

12.3.3 Flex —改良版 Lex

Flex は Lex 互換の高速字句解析生成ソフトです。

flex(1) の入門書は”info flex” の中にあります。

シンプルな例が”/usr/share/doc/flex/examples/” の下にあります。[1]

12.3.4 Bison —改良版 Yacc

Yacc 互換の前方参照可能な LR パーサーとか LALR パーサー生成ソフトは、いくつかのパッケージによって Debian 上で提供されています。

[1] 現行のシステム下でこれらを動かすには少々微調整が必要かもしれません。

パッケージ	ポプコン	サイズ	説明
bison	V:7, I:70	3122	GNU LALR パーサー生成ソフト
byacc	V:0.1, I:2.6	263	Berkeley LALR パーサー生成ソフト
btyacc	V:0.01, I:0.04	247	byacc に基づいたバックトラッキング機能付きパーサー生成ソフト

Table 12.11: Yacc 互換の LALR パーサー生成ソフトのリスト

bison(1) の入門書は”`info bison`” の中にあります。

あなた自身の”`main()`” と”`yyerror()`” を供給する必要があります。”`main()`” は、Flex によって通常作成された”`yylex()`” を呼び出す”`yyparse()`” を呼び出します。

シンプルなターミナル上の計算機プログラムの作成例をここに示します。

`example.y` を作成しましょう:

```
/* calculator source for bison */
%{
#include <stdio.h>
extern int yylex(void);
extern int yyerror(char *);
%}

/* declare tokens */
%token NUMBER
%token OP_ADD OP_SUB OP_MUL OP_RGT OP_LFT OP_EQU

%%
calc:
 | calc exp OP_EQU    { printf("Y: RESULT = %d\n", $2); }
 ;

exp: factor
 | exp OP_ADD factor  { $$ = $1 + $3; }
 | exp OP_SUB factor  { $$ = $1 - $3; }
 ;

factor: term
 | factor OP_MUL term { $$ = $1 * $3; }
 ;

term: NUMBER
 | OP_LFT exp OP_RGT  { $$ = $2; }
  ;
%%

int main(int argc, char **argv)
{
  yyparse();
}

int yyerror(char *s)
{
  fprintf(stderr, "error: '%s'\n", s);
}
```

`example.l` を作成しましょう:

```
/* calculator source for flex */
%{
#include "example.tab.h"
%}

%%
[0-9]+ { printf("L: NUMBER = %s\n", yytext); yylval = atoi(yytext); return NUMBER; }
"+"    { printf("L: OP_ADD\n"); return OP_ADD; }
"-"    { printf("L: OP_SUB\n"); return OP_SUB; }
"*"    { printf("L: OP_MUL\n"); return OP_MUL; }
"("    { printf("L: OP_LFT\n"); return OP_LFT; }
")"    { printf("L: OP_RGT\n"); return OP_RGT; }
"="    { printf("L: OP_EQU\n"); return OP_EQU; }
"exit" { printf("L: exit\n");   return YYEOF; } /* YYEOF = 0 */
.      { /* ignore all other */ }
%%
```

そして、これを試すためにシェルプロンプトから以下を実行しましょう:

```
$ bison -d example.y
$ flex example.l
$ gcc -lfl example.tab.c lex.yy.c -o example
$ ./example
1 + 2 * ( 3 + 1 ) =
L: NUMBER = 1
L: OP_ADD
L: NUMBER = 2
L: OP_MUL
L: OP_LFT
L: NUMBER = 3
L: OP_ADD
L: NUMBER = 1
L: OP_RGT
L: OP_EQU
Y: RESULT = 9

exit
L: exit
```

12.4 静的コード分析ツール

静的解析 (lint) のようなツールは、自動静的コード分析に役立ちます。

Indent のようなツールは整合性をもってソースコードの再フォーマットすることで人間によるコードレビューを助けます。

Ctags のようなツールは、ソースコード中に見つかる名前のインデックス (とかタグ) を生成することで、人間がコードのレビューするのを助けます。

ティップ
あなたの好きなエディター (emacs か vim) を非同期の lint エンジンプラグインを使うように設定することはあなたがコードを書く際に役立ちます。このようなプラグインは Language Server Protocol を利用することで非常にパワフルになっています。プラグインは非常に変化の激しいので、Debian パッケージではなくアップストリームコードを使うのは良い方策かもしれません。

パッケージ	ポプコン	サイズ	説明
vim-ale	I:0.72	2833	Vim 8 や NeoVim 用の非同期静的解析エンジン
vim-syntastic	I:1.8	1379	Vim 用のシンタックスシェックのハック
elpa-flycheck	V:0.1, I:1.5	1130	Emacs 用の現代的な同時進行のシンタックスチェック
elpa-relint	I:0.05	150	Emacs Lisp の正規表現 (regexp) の間違い検出器
cppcheck-gui	V:0.09, I:0.93	7973	静的 C/C++ コード分析ツール (GUI)
shellcheck	V:4, I:17	22860	シェルスクリプトの静的解析 (lint) ツール
pyflakes3	V:2, I:14	20	Python 3 の受動チェッカー
pylint	V:4, I:19	2109	Python コード静的チェックソフト
perl	V:672, I:991	840	静的コードチェックソフト付きのインタープリタ: B::Lint(3perl)
rubocop	V:0.08, I:1.00	4708	Ruby 静的コード分析ツール
clang-tidy	V:2, I:11	20	Clang に基づく C++ の静的解析 (lint) ツール
splint	V:0.05, I:0.75	2325	C プログラムを静的にバグのチェックするためのツール
flawfinder	V:0.05, I:0.41	187	C/C++ ソースコードを検査してセキュリティーの脆弱性を探すツール
black	V:3, I:15	11130	非妥協的な Python コードフォーマッター
perltidy	V:0.4, I:2.9	3086	Perl スクリプトのインデントとリフォーマット
indent	V:0.3, I:4.7	438	C 言語ソースコードフォーマッタープログラム
astyle	V:0.2, I:2.4	770	C と C++ と Objective-C と C# と Java のソースコードインデンター
bcpp	V:0.02, I:0.21	114	C(++) の美化プログラム
xmlindent	V:0.06, I:0.70	52	XML ストリームリフォーマッタ
global	V:0.1, I:1.4	1923	ソースコードの検索と閲覧のツール
exuberant-ctags	V:1, I:12	341	ソースコード定義のタグファイルのインデックスの構築
universal-ctags	V:1, I:12	4238	ソースコード定義のタグファイルのインデックスの構築

Table 12.12: 静的コード分析ツールのリスト

12.5 デバグ

デバッグはプログラミング活動において重要です。プログラムのデバッグ法を知ることで、あなたも意味あるバグリポートを作成できるような良い Debian ユーザーになれます。

パッケージ	ポプコン	サイズ	文書
gdb	V:94, I:163	13472	gdb-doc が提供する”info gdb”
ddd	V:0.2, I:4.8	4519	ddd-doc が提供する”info ddd”

Table 12.13: デバッグパッケージのリスト

12.5.1 基本的な gdb 実行

Debian 上の第一義的デバッガは、実行中のプログラムを検査できるようにする gdb(1) です。

gdb と関連プログラムを以下のようにインストールしましょう。

```
# apt-get install gdb gdb-doc build-essential devscripts
```

gdb のよいチュートリアルについては以下を参照下さい:

- “info gdb”
- /usr/share/doc/gdb-doc/html/gdb/index.html 中の“Debugging with GDB”
- “ウェブ上のチュートリアル”

次は gdb(1) を”-g” を使ってデバッグ情報を付けてコンパイルされた”program” に使う簡単な例です。

```
$ gdb program
(gdb) b 1                # set break point at line 1
(gdb) run args           # run program with args
(gdb) next               # next line
...
(gdb) step               # step forward
...
(gdb) p parm             # print parm
...
(gdb) p parm=12          # set value to 12
...
(gdb) quit
```

ティップ
多くの gdb(1) コマンドは省略できます。タブ展開はシェル同様に機能します。

12.5.2 Debian パッケージのデバグ

全てのインストールされたバイナリーはデフォルトでは Debian システム上ではストリップされているべきなので、ほとんどのデバグシンボルは普通のパッケージからは除かれています。gdb(1) を使って Debian パッケージをデバグするためには、*-dbgsym パッケージ (例えば coreutils の場合は coreutils-dbgsym) をインストールする必要があります。ソースパッケージは、普通のバイナリーパッケージとともに *-dbgsym パッケージを自動生成

し、そうしたデバグパッケージは別にして debian-debug アーカイブ中に置かれます。詳細は Debian Wiki の記事を参照下さい。

デバッグしようとしているパッケージに *-dbgsym パッケージが無い場合は、以下のようにしてリビルドした後でインストールする必要があります。

```
$ mkdir /path/new ; cd /path/new
$ sudo apt-get update
$ sudo apt-get dist-upgrade
$ sudo apt-get install fakeroot devscripts build-essential
$ apt-get source package_name
$ cd package_name*
$ sudo apt-get build-dep ./
```

必要に応じてバグを修正します。

例えば以下のように、既存パッケージを再コンパイルする時は”+debug1” を後ろに付けたり、リリース前のパッケージをコンパイルする時は”~pre1” を後ろに付けたりと、正規の Debian バージョンとかち合わないようにパッケージバージョンを増やします。

```
$ dch -i
```

以下のようにしてデバグシンボル付きでパッケージをコンパイルしてインストールします。

```
$ export DEB_BUILD_OPTIONS="nostrip noopt"
$ debuild
$ cd ..
$ sudo debi package_name*.changes
```

パッケージのビルドスクリプトを確認して、バイナリーのコンパイルに確実に”CFLAGS=-g -Wall” が使われているようにします。

12.5.3 バックトレースの収集

プログラムがクラッシュするのに出会った場合に、バックトレース情報をバグレポートに切り貼りして報告するのは良い考えです。

バックトレースは gdb(1) によって以下のような段取りで得られます。

- GDB の中でクラッシュアプローチ:
  - GDB からプログラムを実行します。
  - プログラムがクラッシュします。
  - GDB プロンプトで”bt” と打ちます。
- 先にクラッシュアプローチ:
  - **“/etc/security/limits.conf”**ファイルを更新して以下を含めます:

    ```
    * soft core unlimited
    ```

  - シェルプロンプトで”ulimit -c unlimited” と打ちます。
  - このシェルプロンプトからプログラムを実行します。
  - プログラムがクラッシュしてコアダンプファイルができます。
  - ”gdb gdb ./program_binary core” として core dump ファイルを GDB にロードします。
  - GDB プロンプトで”bt” と打ちます。

無限ループやキーボードが凍結した場合、Ctrl-\ か Ctrl-C を押すか“kill -ABRT *PID*”を実行することでプログラムを強制終了できます。(項 9.4.12 を参照下さい)

ティップ
しばしば、一番上数行が”malloc()” か”g_malloc()” 中にあるバックトレースを見かけます。こういったことが起こる場合は、大体あまりあなたのバックトレースは役に立ちません。有用な情報を見つけるもっとも簡単な方法は環境変数”$MALLOC_CHECK_” の値を 2 と設定することです (malloc(3))。gdb を実行しながらこれを実行するには以下のようにします。

```
$ MALLOC_CHECK_=2 gdb hello
```

12.5.4 高度な gdb コマンド

コマンド	コマンド目的の説明
(gdb) thread apply all bt	マルチスレッドプログラムの全てのスレッドのバックトレースを取得
(gdb) bt full	関数コールのスタック上に来たパラメーターを取得
(gdb) thread apply all bt full	異常のオプションの組み合わせでバックトレースとパラメーターを取得
(gdb) thread apply all bt full 10	無関係の出力を切り最後の 10 のコールに関するバックトレースとパラメーターを取得
(gdb) set logging on	gdb アウトプットをファイルに書き出す (デフォールトは”gdb.txt”)

Table 12.14: 高度な gdb コマンドのリスト

12.5.5 ライブラリーへの依存の確認

以下のように ldd(1) を使ってプログラムのライブラリーへの依存関係をみつけだします。

```
$ ldd /usr/bin/ls
        librt.so.1 => /lib/librt.so.1 (0x4001e000)
        libc.so.6 => /lib/libc.so.6 (0x40030000)
        libpthread.so.0 => /lib/libpthread.so.0 (0x40153000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

`chroot` された環境下で ls(1) が機能するには、上記ライブラリーがあなたの `chroot` された環境内で利用可能である必要があります。

項 9.4.6 を参照下さい。

12.5.6 動的呼び出し追跡ツール

Debian 中に複数の動的呼び出し追跡ツールがあります。項 9.4 を参照下さい。

12.5.7 X エラーのデバグ

GNOME プログラム preview1 が X エラーを受けると、以下のようなメッセージが見つかります。

```
The program 'preview1' received an X Window System error.
```

このような場合には、プログラムを”--sync” 付きで実行して、バックトレースを得るために”gdk_x_error” 関数上で停止するようにしてみます。

12.5.8 メモリーリーク検出ツール

Debian にはメモリーリークを検出するプログラムがいくつか利用可能です。

パッケージ	ポプコン	サイズ	説明
libc6-dev	V:259, I:577	12863	mtrace(1): glibc 中の malloc デバッグ機能
valgrind	V:4, I:31	89318	メモリーデバッガとプロファイラ
electric-fence	V:0.2, I:2.2	69	malloc(3) デバッガ
libdmalloc5	I:0.54	373	メモリーアロケーションのデバグ用ライブラリー
duma	I:0.06	295	C および C++ プログラムのバッファオーバーランとアンダーランを検出するライブラリー

Table 12.15: メモリーリーク検出ツールのリスト

12.5.9 バイナリーのディスアッセンブリー

以下のように objdump(1) を使ってバイナリーコードをディスアッセンブルできます。

```
$  objdump -m i386 -b binary -D /usr/lib/grub/x86_64-pc/stage1
```

> 注意
> gdb(1) は対話的にコードをディスアッセンブルするのに使えます。

12.6 ビルドツール

パッケージ	ポプコン	サイズ	文書
make	V:155, I:558	1762	make-doc が提供する"info make"
autoconf	V:28, I:189	2244	autoconf-doc が提供する"info autoconf
automake	V:27, I:188	1959	automake が提供する"info automake"
libtool	V:23, I:172	1276	libtool-doc が提供する"info libtool"
cmake	V:23, I:124	63999	クロスプラットフォームでオープンソースの make システム cmake(1)
ninja-build	V:9, I:55	456	Make と似た精神の小さなビルドシステム ninja(1)
meson	V:6, I:29	4402	ninja の上に構築された高生産性のビルドシステム meson(1)
xutils-dev	V:0.6, I:6.6	1495	imake(1), xmkmf(1), 他

Table 12.16: ビルドツールパッケージのリスト

12.6.1 Make

Make はプログラムのグループを管理するためのユーティリティーです。make(1) を実行すると、make は"Makefile"というルールファイルを読み、ターゲットが最後に変更された後で変更された前提ファイルにターゲットが依存している場合やターゲットが存在しない場合にはターゲットを更新します。このような更新は同時並行的にされるかもしれません。

ルールファイルのシンタックスは以下の通りです。

```
target: [ prerequisites ... ]
 [TAB]  command1
 [TAB]  -command2 # ignore errors
 [TAB]  @command3 # suppress echoing
```

上記で、”[TAB]” は TAB コードです。各行は make による変数置換後シェルによって解釈されます。スクリプトを継続する行末には”\” を使います。シェルスクリプトの環境変数のための”$” を入力するためには”$$” を使います。

ターゲットや前提に関するインプリシット (暗黙) ルールは、例えば以下のように書けます。

```
%.o: %.c header.h
```

上記で、ターゲットは”%” という文字を (1 つだけ) 含んでいます。”%” は実際のターゲットファイル名の空でないいかなる部分文字列ともマッチします。前提もまた同様にそれらの名前が実際のターゲットファイル名にどう関連するかを示すために”%” を用いることができます。

自動変数	変数値
$@	ターゲット
$<	最初の前提条件
$?	全ての新規の前提条件
$^	全ての前提条件
$*	”%” はターゲットパターンの軸にマッチします

Table 12.17: make の自動変数のリスト

変数展開	説明
`foo1 := bar`	一回だけの展開
`foo2 = bar`	再帰的展開
`foo3 += bar`	後ろに追加

Table 12.18: make 変数の展開のリスト

”`make -p -f/dev/null`” を実行して自動的な内部ルールを確認下さい。

12.6.2 Autotools

Autotools は多くの Unix-like システムに移植可能なソースコードパッケージを作ることを援助するように設計された一連のプログラムツールです。

- Autoconf は”`configure.ac`” から”`configure`” を生成します。
  - その後、”`configure`” は”`Makefile.in`” から”`Makefile`” を生成します。
- Automake は”`Makefile.am`” から”`Makefile.in`” を生成します。
- Libtool は共有ライブラリーをソースコードからコンパイルする時のソフトウェアー移植性問題を解決するためのシェルプログラムです。

12.6.2.1 プログラムをコンパイルとインストール



警告
システムファイルをあなたがコンパイルしたプログラムでインストールする時に上書きしてはいけません。

Debian は”/usr/local/” とか”/opt” 中のファイルに触れません。プログラムをソースからコンパイルする場合、Debian とかち合わないようにそれを”/usr/local/” の中にインストールします。

```
$ cd src
$ ./configure --prefix=/usr/local
$ make # this compiles program
$ sudo make install # this installs the files in the system
```

12.6.2.2 プログラムのアンインストール

オリジナルのソースを保有し、それが autoconf(1)/automake(1) と使用しあなたがそれをどう設定したかを覚えているなら、以下のように実行してソフトウェアーをアンイストールします。

```
$ ./configure all-of-the-options-you-gave-it
$ sudo make uninstall
```

この代わりに、”/usr/local/” の下にだけインストールプロセスがファイルを置いたことが絶対に確実でそこに重要なものが無いなら、以下のようにしてその内容を消すことが出来ます。

```
# find /usr/local -type f -print0 | xargs -0 rm -f
```

どこにファイルがインストールされるか良く分からない場合には、checkinstall パッケージにある checkinstall(8) を使いアンインストールする場合クリーンなパスとなるようにすることを考えましょう。これは”-D” オプションを使うと Debian パッケージを作成できます。

12.6.3 Meson

ソフトウェアービルドシステムは進化してきています:

- Make の上に構築された Autotools は 1990 年代より移植可能なビルドインフラのデファクトスタンダードです。これは非常に遅いです。
- 2000 年に最初にリリースされた CMake は、スピードが大幅向上させたが、依然として本質的に遅い Make の上に構築されていました。(現在は、Ninja がバックエンドで使えます。)
- 2012 年に最初にリリースされた Ninja は、さらなるビルド速度の向上のために Make の置き換えを意図し、その入力ファイルはハイレベルビルドシステムが生成するように設計されています。
- 2013 年に最初にリリースされた Meson は、新しく人気ある Ninja をバックエンドに使うハイレベルビルドシステムです。

”The Meson Build system” や”The Ninja build system” にある文書を参照下さい。

12.7 ウェッブ

基本的な対話式動的ウェッブページは以下のようにして作られます。

- 質問 (クエリー) はブラウザーのユーザーに HTML フォームを使って提示されます。
- フォームのエントリーを埋めたりクリックすることによって以下の符号化されたパラメーター付きの URL 文字列をブラウザーからウェッブサーバーに送信します。
  - ”https://www.foo.dom/cgi-bin/program.pl?VAR1=VAL1&VAR2=VAL2&VAR3=VAL3”
  - ”https://www.foo.dom/cgi-bin/program.py?VAR1=VAL1&VAR2=VAL2&VAR3=VAL3”

  - "https://www.foo.dom/program.php?VAR1=VAL1&VAR2=VAL2&VAR3=VAL3"

- URL 中の"%nn" は 16 進数で nn の値の文字と置き換えられます。
- 環境変通は以下のように設定されます: "QUERY_STRING="VAR1=VAL1 VAR2=VAL2 VAR3=VAL3"".
- ウェッブサーバー上の CGI プログラム ("program.*" のいずれでも) が環境変数"$QUERY_STRING" の下で実行されます。
- CGI プログラムの STDOUT (標準出力) がウエブブラウザーに送られ対話式の動的なウェッブページとして表示されます。

セキュリティー上、CGI パラメーターを解釈する手作りの急ごしらえのプログラムは作らない方が賢明です。Perl や Python にはこのための確立したモジュールが存在します。PHP はこの様な機能が同梱されています。クライアントでのデーターのストレージの必要がある場合、HTTP クッキーが使われます。クライアントサイドのデーター処理が必要な場合、Javascript が良く使われます。

詳しくは、Common Gateway Interface や The Apache Software Foundation や JavaScript を参照下さい。

https://www.google.com/search?hl=en&ie=UTF-8&q=CGI+tutorial を URL として直接ブラウザーのアドレスに入れ Google で"CGI tutorial" を検索するとグーグルサーバー上の CGI スクリプトが動いているのを観察する良い例です。

12.8 ソースコード変換

ソースコード変換するプログラムがあります。

パッケージ	ポプコン	サイズ	キーワード	説明
f2c	V:0.1, I:1.9	443	FORTRAN → C	FORTRAN 77 から C/C++ へのソースコード変換ソフト: f2c(1)
intel2gas	V:0.01, I:0.17	178	NASM → GAS	NASM (Intel フォーマット) から GNU Assembler (GAS) への変換ソフト

Table 12.19: ソースコード変換ツールのリスト

12.9 Debian パッケージ作成

Debian パッケージを作りたい場合には、次を読みましょう。

- 基本的なパッケージシステムの理解には第 2 章
- 基本的なポーティングプロセスの理解のために、項 2.7.13
- 基本的な chroot 技術の理解のために、項 9.11.4
- debuild(1) と sbuild(1)
- リコンパイルとデバグは項 12.5.2
- Guide for Debian Maintainers (debmake-doc パッケージ)
- Debian Developer's Reference (developers-reference パッケージ)
- Debian ポリシーマニュアル (debian-policy パッケージ)

debmake や dh-make や dh-make-perl 等のパッケージングを補助するパッケージがあります。

Appendix A

補遺

以下が本文書の背景です。

A.1 Debian 迷路

Linux システムはネットワーク化されたコンピューターのための非常にパワフルなコンピュータープラットフォームです。しかし、Linux の全能力を利用する方法を学ぶことはたやすいことではありません。非 PostScript プリンタが接続された LPR プリンタの設定がこんなつまずく点の良い例でした。(最近のインストールでは CUPS システムが使われるのでもうこの様な問題はありません。)

” ソースコード” という完全かつ詳細なマップが存在します。これは非常に正確ですが理解することが難しいものです。また、HOWTO や mini-HOWTO と呼ばれるリファレンスもあります。これらは理解はしやすいのですが、詳細過ぎて全体像を失いがちです。ちょっとコマンドを実行する必要がある時に、長大な HOWTO の該当する章を探すのには骨が折れることが時々あります。

この”Debian リファレンス (第 2.153 版)” (2026-11-29 05:38:00 UTC) が Debian 迷路の真っ只中にいる皆様にとって解決の糸口となることを望みます。

A.2 著作権の経緯

Debian リファレンスは青木修 <osamu at debian dot org> が個人用システム管理メモとして書き始められました。多くの内容が debian-user メーリングリストや他の Debian のリソースから得られた知識由来です。

当時 Debian Documentation Project で非常にアクティブであった、Josip Rodin 氏の助言に従い、DDP 文書の一部として”Debian リファレンス (第 1 版、2001-2007)” を作りました。

6 年経った時点で、青木はオリジナルの”Debian リファレンス (第 1 版)” が時代遅れとなっている事に気づき多くの内容を書き換え始めました。新たな”Debian リファレンス (第 2 版)” が 2008 年にリリースされました。

著者は、新規トピックス (Systemd, Wayland, IMAP, PipeWire, Linux kernel 5.10) を取り扱い、旧式のトピックス (SysV init, CVS, Subversion, SSH protocol 1, Linux kernels before 2.5) を削除して、”Debian Reference (version 2)” を更新しました。Jessie 8 (2015-2020) リリースやそれ以前の状況は、ほぼ削除しました。

この”Debian Reference (version 2.153)” (2026-11-29 05:38:00 UTC) は、主に Trixie (=`stable`) と Forky (=`testing`) Debian リリースカバーします。

チュートリアルの内容はその内容とインスピレーションを次から得ました。

- ”Linux User’s Guide” Larry Greenfield 著 (1996 年 12 月)
  - ”Debian Tutorial” によって陳腐化

- ”Debian Tutorial” Havoc Pennington 著。(1998 年 12 年 11 日)
  - Oliver Elphick と Ole Tetlie と James Treacy と Craig Sawyer と Ivan E. Moore II による一部著作
  - ”Debian GNU/Linux: Guide to Installation and Usage” によって陳腐化
- ”Debian GNU/Linux: Guide to Installation and Usage” John Goerzen and Ossama Othman 著 (1999 年)
  - ”Debian リファレンス (第 1 版)” によって陳腐化

パッケージやアーカイブに関する記述はそのオリジンやインスピレーションの一部を次に遡ることができます。

- ”Debian FAQ” (Josip Rodin が維持していた 2002 年 3 月版)

他の内容はそのオリジンやインスピレーションを次に遡ることができます。

- ”Debian リファレンス (第 1 版)” 青木修著 (2001 年〜2007 年)
  - 2018 年のより新しい”Debian リファレンス (第 2 版)” によって陳腐化

以前の”Debian リファレンス (第 1 版)” は次によって作られました。

- ネットワーク設定に関する大部分の内容は Thomas Hood が寄稿
- X と VCS に関連するかなりの内容は Brian Nelson が寄稿
- ビルドスクリプトや多くの内容に関する訂正で Jens Seidel が寄与
- David Sewell による徹底的な校正
- 翻訳者やコントリビューターやバグ報告者達による多くの寄与

Debian システム上の多くのマニュアルページや info ページやアップストリームのウエッブページや Wikipedia の文書が本文書を書く上での第一義的参照情報として使われました。青木修が公正な使用と考える範囲内で、それらの多くの部分、特にコマンドの定義が、本文書の文体と目的に合うように注意深い編集をした後、断片的文言として使われました。

gdb デバッガーに関する記述は Arii Pollak と Loïc Minier と Dafydd Harries の了承のもと backtrace に関する Debian wiki の内容を拡張して使いました。

既に上記で触れた項目を除く現在の”Debian リファレンス (第 2.153 版)” (2026-11-29 05:38:00 UTC) の内容はほとんど私自身の仕事です。これらはコントリビュータ − によっても更新されています。

著者である青木修は本文書を世に送ることにご助力戴いた皆様に感謝いたします。

A.3 文書のフォーマット

英語の元文書のソースは現在 DocBook XML ファイルで書かれています。この Docbook XML ソースは HTML やプレーンテキストや PostScript や PDF に変換されます。(一部書式は頒布時にスキップされるかもしれません。)