



debian

Debian リファレンス

[FAMILY Given]

製作著作 © 2013-2021 Osamu Aoki (青木 修)

本 Debian リファレンス (第 2.85 版) (2021-09-17 09:11:56 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。

COLLABORATORS			
	TITLE : Debian リファレンス		
ACTION	NAME	DATE	SIGNATURE
WRITTEN BY	[FAMILY Given]	September 17, 2021	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

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	3
1.1.8	システムをシャットダウンする方法	4
1.1.9	まとも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	5
1.1.12	sudo の設定	5
1.1.13	お遊びの時間	6
1.2	Unix-like ファイルシステム	6
1.2.1	Unix ファイルの基礎	6
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	8
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	10
1.2.5	ユーザーのグループ (group) のパーミッション	11
1.2.6	タイムスタンプ	12
1.2.7	リンク	13
1.2.8	名前付きパイプ (FIFO)	14
1.2.9	ソケット	14
1.2.10	デバイスファイル	15
1.2.11	特別なデバイスファイル	15
1.2.12	procfs と sysfs	16
1.2.13	tmpfs	16
1.3	ミッドナイトコマンドー (MC)	17

1.3.1	MC のカスタム化	17
1.3.2	MC の始動	17
1.3.3	MC のファイルマネージャー	17
1.3.4	MC のコマンドライントリック	18
1.3.5	MC の内部エディター	18
1.3.6	MC の内部ビューワー	19
1.3.7	MC の自動起動機能	19
1.3.8	MC の FTP 仮想ファイルシステム	19
1.4	基本の Unix 的作業環境	19
1.4.1	login シェル	19
1.4.2	Bash のカスタム化	20
1.4.3	特別のキーストローク	21
1.4.4	マウス操作	21
1.4.5	ページャー	22
1.4.6	テキストエディター	22
1.4.7	デフォルトのテキストエディターの設定	22
1.4.8	Using vim	23
1.4.9	シェル活動の記録	24
1.4.10	基本 Unix コマンド	24
1.5	シェルプロンプト	26
1.5.1	コマンド実行と環境変数	26
1.5.2	”\$LANG” 変数	26
1.5.3	”\$PATH” 変数	28
1.5.4	”\$HOME” 変数	28
1.5.5	コマンドラインオプション	28
1.5.6	シェルグロブ	28
1.5.7	コマンドの戻り値	29
1.5.8	典型的なコマンドシーケンスとシェルリディ렉션	30
1.5.9	コマンドエイリアス	31
1.6	Unix 的テキスト処理	32
1.6.1	Unix テキストツール	32
1.6.2	正規表現	33
1.6.3	置換式	33
1.6.4	正規表現を使ったグローバル置換	34
1.6.5	テキストファイルからのデータ抽出	35
1.6.6	コマンドをパイプするためのスクリプト断片	36

2 Debian パッケージ管理	38
2.1 Debian パッケージ管理の前提条件	38
2.1.1 パッケージ設定	38
2.1.2 基本的な注意事項	39
2.1.3 永遠のアップグレード人生	40
2.1.4 Debian アーカイブの基本	41
2.1.5 Debian は 100% フリーソフトウェアです	44
2.1.6 パッケージ依存関係	45
2.1.7 パッケージ管理のイベントの流れ	46
2.1.8 パッケージ管理のトラブルへの応急対処法	47
2.2 基本的パッケージ管理操作	48
2.2.1 apt と apt-get/apt-cache と aptitude の比較	48
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	49
2.2.3 aptitude のインタラクティブな使用	51
2.2.4 aptitude のキーバインディング	51
2.2.5 aptitude の下でのパッケージの表示	52
2.2.6 aptitude を使った探索方法	53
2.2.7 aptitude の regex 式	53
2.2.8 aptitude による依存関係の解決	55
2.2.9 パッケージアクティビティログ	55
2.3 aptitude 操作例	55
2.3.1 regex にマッチするパッケージ名のパッケージをリスト	55
2.3.2 regex マッチをしての閲覧	56
2.3.3 パッケージの完全削除	56
2.3.4 自動 / 手動インストール状態の整理	56
2.3.5 システム全体のアップグレード	57
2.4 高度なパッケージ管理操作	58
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	58
2.4.2 インストールされたパッケージファイルの検証	60
2.4.3 パッケージ問題からの防御	60
2.4.4 パッケージメタデータの検索	60
2.5 Debian パッケージ管理の内部	60
2.5.1 アーカイブのメタデータ	60
2.5.2 トップレベルの”Release” ファイルと信憑性	61
2.5.3 アーカイブレベルの”Release” ファイル	62
2.5.4 パッケージメタデータの取得	63
2.5.5 APT に関するパッケージ状態	63
2.5.6 aptitude に関するパッケージ状態	63
2.5.7 取得したパッケージのローカルコピー	63

2.5.8	Debian パッケージファイル名	64
2.5.9	dpkg コマンド	64
2.5.10	update-alternative コマンド	65
2.5.11	dpkg-statoverride コマンド	66
2.5.12	dpkg-divert コマンド	66
2.6	壊れたシステムからの復元	66
2.6.1	古いユーザーの設定との非互換性	67
2.6.2	重複するファイルを持つ相異なるパッケージ	67
2.6.3	壊れたパッケージスクリプトの修正	67
2.6.4	dpkg コマンドを使つての救済	68
2.6.5	パッケージセレクションの復元	68
2.7	パッケージ管理のヒント	69
2.7.1	Debian パッケージの選択方法	69
2.7.2	混合したアーカイブソースからのパッケージ	69
2.7.3	候補バージョンの調整	71
2.7.4	Updates と Backports	72
2.7.5	”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	73
2.7.6	unstable からのパッケージと共に、testing を追いかける	73
2.7.7	experimental からのパッケージと共に、unstable を追いかける	75
2.7.8	パッケージの自動ダウンロードとアップグレード	75
2.7.9	APT のよるダウンロードバンド幅の制限	75
2.7.10	緊急ダウングレード	76
2.7.11	誰がパッケージをアップロードしたのか?	76
2.7.12	equivs パッケージ	77
2.7.13	安定版システムへのパッケージ移植	77
2.7.14	APT のためのプロキシサーバー	78
2.7.15	パッケージ管理の追加参考文書	78
3	システムの初期化	79
3.1	ブートストラッププロセスの概要	79
3.1.1	1 段目: UEFI	80
3.1.2	2 段目: ブートローダー	80
3.1.3	3 段目: ミニ Debian システム	81
3.1.4	4 段目: 通常の Debian システム	82
3.2	Systemd init	82
3.2.1	ホスト名	84
3.2.2	ファイルシステム	84
3.2.3	ネットワークインターフェースの初期化	84
3.3	カーネルメッセージ	84

3.4	システムメッセージ	85
3.5	System management	85
3.6	Other system monitors	87
3.7	systemd のカスタム化	87
3.7.1	ソケットの起動	87
3.8	udev システム	88
3.8.1	カーネルモジュール初期化	88
4	認証とアクセスの制御	90
4.1	通常の Unix 認証	90
4.2	アカウントとパスワードの情報管理	92
4.3	良好なパスワード	92
4.4	暗号化されたパスワード作成	93
4.5	PAM と NSS	93
4.5.1	PAM と NSS によってアクセスされる設定ファイル	94
4.5.2	最新の集中システム管理	94
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」	95
4.5.4	パスワード規則強化	96
4.6	認証のセキュリティ	96
4.6.1	インターネット上でセキュアなパスワード	96
4.6.2	セキュアシェル	97
4.6.3	インターネットのためのセキュリティ強化策	97
4.6.4	root パスワードのセキュリティ確保	97
4.7	他のアクセスコントロール	98
4.7.1	sudo	98
4.7.2	PolicyKit	98
4.7.3	サーバーのサービスへのアクセスの制限	99
4.7.4	Linux のセキュリティ機能	99
5	ネットワークの設定	101
5.1	基本的ネットワークインフラ	101
5.1.1	ホスト名の解決	101
5.1.2	ネットワークインターフェース名	103
5.1.3	LAN のためのネットワークアドレス範囲	103
5.1.4	ネットワークデバイスサポート	104
5.2	デスクトップのためのモダンネットワーク設定	104
5.2.1	GUI のネットワーク設定ツール	105
5.3	GUI 無しのモダンネットワーク設定	105
5.4	低水準ネットワーク設定	106

5.4.1	Iproute2 コマンド	106
5.4.2	安全な低レベルネットワーク操作	106
5.5	ネットワークの最適化	107
5.5.1	最適 MTU の発見	107
5.5.2	WAN TCP の最適化	108
5.6	Netfilter インフラ	108
6	ネットワークアプリケーション	110
6.1	ウェブブラウザ	110
6.1.1	ブラウザ設定	110
6.2	メールシステム	111
6.2.1	Eメールの基本	112
6.2.2	現代的メールサービスの限界	112
6.2.3	Historic mail service expectation	113
6.2.4	メール転送エージェント (MTA)	113
6.2.4.1	exim4 設定	114
6.2.4.2	SASL を使う postfix の設定	115
6.2.4.3	メールアドレス設定	116
6.2.4.4	基本的な MTA の操作	117
6.3	リモートアクセスサーバーとユーティリティー (SSH)	117
6.3.1	SSH の基本	119
6.3.2	リモートホストでのユーザ名	119
6.3.3	リモートパスワード無しでの接続	120
6.3.4	外部 SSH クライアントへの対処法	120
6.3.5	ssh-agent の設定	120
6.3.6	リモートホストからメールを送信する	121
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	121
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	121
6.3.9	SSH のトラブルシュート	121
6.4	プリントサーバーとユーティリティー	122
6.5	他のネットワークアプリケーションサーバー	122
6.6	他のネットワークアプリケーションクライアント	123
6.7	システムデーモンの診断	123
7	GUI システム	125
7.1	GUI desktop environment	125
7.2	GUI 通信プロトコル	126
7.3	GUI インフラストラクチャー	127
7.4	GUI アプリケーション	127

7.5	フォント	129
7.5.1	基本的なフォント	129
7.5.2	フォントのラスタ化	129
7.6	サンドボックス	131
7.7	リモートデスクトップ	132
7.8	X サーバ接続	132
7.8.1	X サーバローカル接続	132
7.8.2	X サーバリモート接続	133
7.8.3	X サーバ chroot 接続	133
7.9	クリップボード	133
8	I18N と L10N	135
8.1	ロケール	135
8.1.1	UTF-8 ロケールを使う根拠	135
8.1.2	ロケールの再設定	136
8.1.3	ファイル名の符号化方式	137
8.1.4	地域化されたメッセージと翻訳された文書	137
8.1.5	ロケールの効果	137
8.2	キーボード入力	138
8.2.1	The keyboard input for Linux console and X Window	138
8.2.2	Wayland 向けのキーボード入力	138
8.2.3	IBus を使うインプットメソッドのサポート	138
8.2.4	日本語の例	138
8.3	ディスプレイ出力	139
8.4	東アジア不明瞭文字幅文字	140
9	システムに関するティップ	141
9.1	The console tips	141
9.1.1	シェルの活動を綺麗に記録	141
9.1.2	screen プログラム	142
9.1.3	Navigating around directories	143
9.1.4	Readline wrapper	143
9.2	Vim のカスタム化	143
9.2.1	Customizing vim with internal features	144
9.2.2	Customizing vim with external packages	144
9.3	データーの記録と表現	145
9.3.1	ログデーモン	145
9.3.2	ログアナライザー	145
9.3.3	テキストデーターのカスタム化表示	145

9.3.4	時間と日付のカスタム化表示	146
9.3.5	着色化されたシェル出力	147
9.3.6	着色化されたコマンド	147
9.3.7	複雑な反復のためにエディターでの活動を記録	147
9.3.8	X アプリケーションの画像イメージの記録	148
9.3.9	設定ファイルの変更記録	148
9.4	プログラム活動の監視と制御と起動	148
9.4.1	プロセスの時間計測	150
9.4.2	スケジューリングの優先度	150
9.4.3	ps コマンド	150
9.4.4	top コマンド	150
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	151
9.4.6	プログラム活動の追跡	151
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	151
9.4.8	一定間隔でコマンドを反復実行	151
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	152
9.4.10	GUI からプログラムをスタート	152
9.4.11	スタートするプログラムのカスタム化	153
9.4.12	プロセスの停止	154
9.4.13	タスク 1 回実行のスケジュール	154
9.4.14	タスク定期実行のスケジュール	154
9.4.15	Alt-SysRq キー	156
9.5	システム管理ティップ	157
9.5.1	だれがシステムを利用している?	157
9.5.2	全員への警告	157
9.5.3	ハードウェアの識別	157
9.5.4	ハードウェア設定	159
9.5.5	システムとハードウェアの時間	159
9.5.6	ターミナルの設定	159
9.5.7	音のインフラ	160
9.5.8	スクリーンセーバーの無効化	160
9.5.9	ブザー音の無効化	161
9.5.10	メモリー使用状況	161
9.5.11	システムのセキュリティーと整合性のチェック	162
9.6	データ保存のティップ	162
9.6.1	ディスク空間の利用状況	162
9.6.2	ディスクパーティション設定	163
9.6.3	UUID を使ってパーティションをアクセス	163
9.6.4	LVM2	164

9.6.5	ファイルシステム設定	164
9.6.6	ファイルシステムの生成と整合性チェック	165
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	165
9.6.8	スーパブロックによるファイルシステムの最適化	166
9.6.9	ハードディスクの最適化	166
9.6.10	ソリッドステートドライブの最適化	166
9.6.11	SMART を用いたハードディスクの破壊の予測	167
9.6.12	\$TMPDIR 経由で一時保存ディレクトリーを指定	167
9.6.13	LVM を使う使用可能なストレージ空間の拡張	167
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	167
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	168
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	168
9.6.17	シmlinkを使う使用可能なストレージ空間の拡張	168
9.7	ディスクイメージ	168
9.7.1	ディスクイメージの作成	169
9.7.2	ディスクに直接書込み	169
9.7.3	ディスクイメージファイルをマウント	169
9.7.4	ディスクイメージのクリーニング	170
9.7.5	空のディスクイメージ作成	171
9.7.6	ISO9660 イメージファイル作成	171
9.7.7	CD/DVD-R/RW に直接書込み	172
9.7.8	ISO9660 イメージファイルをマウント	172
9.8	バイナリーデーター	173
9.8.1	バイナリーデーターの閲覧と編集	173
9.8.2	ディスクをマウントせずに操作	173
9.8.3	データーの冗長性	173
9.8.4	データーファイルの復元と事故の証拠解析	174
9.8.5	大きなファイルを小さなファイルに分割	174
9.8.6	ファイル内容の消去	175
9.8.7	ダミーファイル	175
9.8.8	ハードディスクの全消去	175
9.8.9	ハードディスク未使用部分の全消去	176
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	176
9.8.11	全てのハードリンクを検索	177
9.8.12	見えないディスクスペースの消費	177
9.9	データー暗号化ティップ	177
9.9.1	dm-crypt/LUKS を使ったリムーバブルディスクの暗号化	178
9.9.2	dm-crypt を使って swap パーティションを暗号化	178
9.9.3	dm-crypt/LUKS で暗号化されたディスクのマウント	179

9.10	カーネル	179
9.10.1	カーネル変数	179
9.10.2	カーネルヘッダー	179
9.10.3	カーネルと関連モジュールのコンパイル	180
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	180
9.10.5	ハードウェアドライバとファームウェア	181
9.11	仮想化システム	182
9.11.1	Virtualization and emulation tools	182
9.11.2	仮想化の業務フロー	184
9.11.3	仮想ディスクイメージファイルをマウント。	184
9.11.4	Chroot システム	185
9.11.5	複数のデスクトップシステム	185
10	データ管理	187
10.1	共有とコピーとアーカイブ	187
10.1.1	アーカイブと圧縮ツール	188
10.1.2	コピーと同期ツール	189
10.1.3	アーカイブの慣用句	189
10.1.4	コピーの慣用句	190
10.1.5	ファイル選択の慣用句	191
10.1.6	アーカイブメディア	192
10.1.7	リムーバブルストレージデバイス	193
10.1.8	データ共有用のファイルシステム選択	194
10.1.9	ネットワーク経由でのデータ共有	195
10.2	バックアップと復元	196
10.2.1	バックアップユーティリティのスイート	197
10.2.2	Personal backup	198
10.3	データセキュリティのインフラ	198
10.3.1	Gnupg のためのキー管理	198
10.3.2	GnuPG をファイルに使用	200
10.3.3	Mutt で GnuPG を使用	200
10.3.4	Vim で GnuPG を使用	201
10.3.5	MD5 和	201
10.4	ソースコードマージツール	201
10.4.1	ソースファイル間の相違の抽出	201
10.4.2	ソースファイルに更新をマージ	201
10.4.3	Interactive merge	203
10.5	Git	203
10.5.1	Git クライアントの設定	203
10.5.2	Basic Git commands	204
10.5.3	Git tips	204
10.5.4	Git リファレンス	205
10.5.5	Other version control systems	205

11 データー変換	208
11.1 テキストデーター変換ツール	208
11.1.1 テキストファイルを iconv を使って変換	208
11.1.2 ファイルが UTF-8 であると iconv を使い確認	210
11.1.3 iconv を使ってファイル名変換	210
11.1.4 行末変換	210
11.1.5 タブ変換	211
11.1.6 自動変換付きエディター	211
11.1.7 プレーンテキスト抽出	212
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	212
11.2 XML データー	212
11.2.1 XML に関する基本ヒント	213
11.2.2 XML 処理	214
11.2.3 XML データー抽出	214
11.2.4 The XML data lint	216
11.3 タイプセッティング	216
11.3.1 roff タイプセッティング	216
11.3.2 TeX/LaTeX	217
11.3.3 マニュアルページを綺麗に印刷	217
11.3.4 マニュアルページの作成	218
11.4 印刷可能データー	218
11.4.1 Ghostscript	218
11.4.2 2 つの PS や PDF ファイルをマージ	218
11.4.3 印刷可能データーユーティリティ	219
11.4.4 CUPS を使って印刷	219
11.5 メールデーター変換	220
11.5.1 メールデーターの基本	220
11.6 グラフィクスデーターツール	221
11.7 その他のデーター変換	221
12 プログラミング	224
12.1 シェルスクリプト	224
12.1.1 POSIX シェル互換性	225
12.1.2 シェル変数	225
12.1.3 シェル条件式	226
12.1.4 シェルループ	227
12.1.5 シェルコマンドライン処理シーケンス	228
12.1.6 シェルスクリプトのためのユーティリティプログラム	229
12.2 Scripting in interpreted languages	229

12.2.1	Debugging interpreted language codes	229
12.2.2	GUI program with the shell script	230
12.2.3	究極の短い Perl スクリプト	231
12.3	Coding in compiled languages	232
12.3.1	C	232
12.3.2	単純な C プログラム (gcc)	232
12.3.3	Flex —改良版 Lex	233
12.3.4	Bison —改良版 Yacc	233
12.4	静的コード分析ツール	234
12.5	デバグ	234
12.5.1	基本的な gdb 実行	234
12.5.2	Debian パッケージのデバグ	236
12.5.3	バックトレースの収集	237
12.5.4	上級 gdb コマンド	237
12.5.5	ライブラリーへの依存の確認	238
12.5.6	Dynamic call tracing tools	238
12.5.7	X エラーのデバグ	238
12.5.8	メモリーリーク検出ツール	238
12.5.9	バイナリーのディスアセンブリー	238
12.6	ビルドツール	239
12.6.1	Make	239
12.6.2	Autotools	239
12.6.2.1	プログラムをコンパイルとインストール	240
12.6.2.2	プログラムのアンインストール	240
12.6.3	Meson	240
12.7	ウェブ	241
12.8	ソースコード変換	241
12.9	Debian パッケージ作成	241
A	補遺	243
A.1	Debian 迷路	243
A.2	著作権の経緯	243
A.3	文書のフォーマット	244

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	4
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	7
1.4	"ls -l" の出力の最初の文字のリスト	9
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	10
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	11
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	12
1.9	タイムスタンプのタイプのリスト	12
1.10	スペシャルなデバイスファイルのリスト	16
1.11	MC のキーバインディング	18
1.12	enter キー入力への MC の反応	19
1.13	シェルプログラムのリスト	20
1.14	Bash のキーバインディングのリスト	21
1.15	List of mouse operations and related key actions on Debian	22
1.16	List of basic Vim key strokes	23
1.17	基本の Unix コマンドのリスト	25
1.18	ロケールの値の 3 つの部分	27
1.19	推奨ロケールのリスト	27
1.20	"\$HOME" の値のリスト	28
1.21	シェルグロブパターン	29
1.22	コマンドの終了コード	29
1.23	シェルコマンドの慣用句	30
1.24	事前定義されたファイルデスクリプタ	31
1.25	BRE と ERE のメタ文字	33
1.26	置換式	34
1.27	コマンドをパイプするためのスクリプト断片	37
2.1	Debian のパッケージ管理ツールのリスト	39
2.2	Debian アーカイブサイトのリスト	42

2.3	Debian アーカイブエリアのリスト	43
2.4	スイーツとコード名の関係	43
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ブサイトのリスト	48
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	50
2.7	aptitude(8) に関する特記すべきコマンドオプション	50
2.8	aptitude のキーバインディングのリスト	51
2.9	aptitude の表示のリスト	52
2.10	標準パッケージ画面の分類	53
2.11	aptitude の regex 式のリスト	54
2.12	パッケージアクティビティのログファイル	55
2.13	高度なパッケージ管理操作	59
2.14	Debian アーカイブのメタデータの内容	61
2.15	Debian パッケージの名前の構造	64
2.16	Debian パッケージ名の各部分に使用可能な文字	64
2.17	dpkg が作成する特記すべきファイル	65
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。	72
2.19	Debian アーカイブ専用のプロキシツールのリスト	78
3.1	ブートローダーのリスト	80
3.2	The meaning of the menu entry of the above part of /boot/grub/grub.cfg	81
3.3	Debian システムのブートユーティリティのリスト	83
3.4	カーネルエラーレベルのリスト	85
3.5	List of typical journalctl command snippets	85
3.6	List of typical systemctl command snippets	86
3.7	List of other monitoring command snippets under systemd	87
4.1	3 つの pam_unix(8) に関する重要な設定ファイル	90
4.2	"/etc/passwd" の 2 番目のエントリーの内容	91
4.3	アカウント情報を管理するコマンドのリスト	92
4.4	パスワード生成ツールのリスト	93
4.5	特記すべき PAM と NSS システムのリスト	94
4.6	PAM NSS によりアクセスされる設定ファイルのリスト	95
4.7	インセキュアとセキュアのサービスとポートのリスト	96
4.8	追加セキュリティ策を提供するツールのリスト	97
5.1	GUI のネットワーク設定ツール	102
5.2	ネットワークアドレス範囲のリスト	104
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	106
5.4	低レベルネットワークコマンドのリスト	106

5.5	ネットワーク最適化ツールのリスト	107
5.6	最適 MTU 値の基本的なガイドライン	108
5.7	ファイアーウォールツールのリスト	109
6.1	ウェブブラウザのリスト	111
6.2	ブラウザプラグインのリスト	111
6.3	メールユーザーエージェント (MUA) のリスト	112
6.4	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	114
6.5	重要 postfix マニュアルページのリスト	116
6.6	メールアドレス関連のファイルのリスト	116
6.7	基本的 MTA 操作のリスト	118
6.8	リモートアクセスサーバーとユーティリティのリスト	118
6.9	SSH 設定ファイルのリスト	119
6.10	SSH クライアント起動例のリスト	120
6.11	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	120
6.12	プリントサーバーとユーティリティのリスト	122
6.13	他のネットワークアプリケーションサーバー	123
6.14	他のネットワークアプリケーションクライアント	124
6.15	よく使われる RFC のリスト	124
7.1	List of desktop environment	125
7.2	List of notable GUI infrastructure packages	127
7.3	特筆すべき GUI アプリケーションの一覧	128
7.4	List of notable TrueType and OpenType fonts	130
7.5	有用フォント環境と関連パッケージのリスト	130
7.6	List of notable sandbox environment and related packages	131
7.7	List of notable remote access server	132
7.8	X サーバーへの接続方法のリスト	132
7.9	List of programs related to manipulating character clipboard	134
8.1	IBus とエンジンパッケージのリスト	139
9.1	List of programs to support console activities	141
9.2	screen キーバインディングのリスト	143
9.3	Information on the initialization of vim	145
9.4	システムログアナライザーのリスト	146
9.5	Display examples of time and date for the "ls -l" command with the time style value	146
9.6	画像の操作ツールのリスト	148
9.7	設定の履歴を記録するパッケージのリスト	148
9.8	プログラム活動の監視と制御のツールのリスト	149

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	150
9.10 ps コマンドのスタイルのリスト	150
9.11 kill コマンドが良く使うシグナルのリスト	155
9.12 List of notable SAK command keys	156
9.13 ハードウェア識別ツールのリスト	158
9.14 ハードウェア設定ツールのリスト	158
9.15 サウンドパッケージのリスト	160
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	161
9.17 報告されるメモリーサイズのリスト	161
9.18 システムセキュリティや整合性確認のためのツールリスト	162
9.19 ディスクパーティション管理パッケージのリスト	163
9.20 ファイルシステム管理用パッケージのリスト	165
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	173
9.22 ディスクをマウントせずに操作するパッケージのリスト	173
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	174
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	174
9.25 データー暗号化ユーティリティのリスト	177
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	180
9.27 仮想化ツールのリスト	183
10.1 アーカイブと圧縮ツールのリスト	188
10.2 コピーと同期ツールのリスト	189
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	194
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	195
10.5 バックアップスイートのユーティリティのリスト	197
10.6 データーセキュリティインフラツールのリスト	199
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	199
10.8 トラストコードの意味のリスト	199
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	200
10.10 ソースコードマージツールのリスト	202
10.11 git 関連のパッケージとコマンドのリスト	203
10.12 Main Git commands	205
10.13 Git tips	206
10.14 List of other version control system tools	207
11.1 テキストデーター変換ツールのリスト	208
11.2 符号化方式値とその使い方リスト	209
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	210
11.4 bsdmainutils と coreutils パッケージ中のタブ変換コマンドのリスト	211

11.5 プレーンテキストデーター抽出ツールのリスト	212
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	213
11.7 XML で事前定義されているエントリーのリスト	214
11.8 XML ツールのリスト	215
11.9 DSSL ツールのリスト	215
11.10 テキストデーター変換ツールのリスト	215
11.11 XML 整形印刷ツールのリスト	216
11.12 タイプ設定ツールのリスト	216
11.13 マンページ作成を補助するパッケージのリスト	218
11.14 Ghostscript PostScript インタープリタのリスト	218
11.15 プリントできるデーターのユーティリティのリスト	219
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	220
11.17 画像データーツールのリスト	222
11.18 その他のデーター変換ツールのリスト	223
12.1 典型的 bashisms のリスト	225
12.2 シェル変数のリスト	225
12.3 シェル変数展開のリスト	226
12.4 重要なシェル変数置換のリスト	226
12.5 条件式中のファイル比較演算子	227
12.6 条件式中での文字列比較演算子のリスト	227
12.7 シェルスクリプト用の小さなユーティリティプログラムを含むパッケージのリスト	229
12.8 List of interpreter related packages	230
12.9 ダイアログプログラムのリスト	230
12.10 コンパイラ関連のパッケージのリスト	232
12.11 Yacc 互換の LALR パーサー生成ソフトのリスト	234
12.12 静的コード分析ツールのリスト	235
12.13 デバッグパッケージのリスト	235
12.14 上級 gdb コマンドのリスト	237
12.15 メモリーリーク検出ツールのリスト	238
12.16 ビルドツールパッケージのリスト	239
12.17 make の自動変数のリスト	239
12.18 make 変数の展開のリスト	240
12.19 ソースコード変換ツールのリスト	242

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 2.85 版) (2021-09-17 09:11:56 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、以下を参照ください:

- [現行安定システム用 Debian GNU/Linux インストールガイド](#)
- [現行テスト \(testing\) 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が、このため最新状況を反映した正確な記述は困難です。現行の不安定版の 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作ろ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次の特徴があります。

- 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パッケージ
- セキュリティーアップデートへの平易なアクセス提供による、安定性とセキュリティの重視
- 不安定版 unstable やテスト版 testing アーカイブによる、最新のソフトウェア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ェアアーキテクチャー

Debian 中のフリーソフトウェア構成要素は、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 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次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。(ポップコン等の利用)

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てしました。

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と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。

- [The Debian Administrator's Handbook](#)
- 一般的情報は <https://www.debian.org> にある Debian サイト
- `"/usr/share/doc/package_name"` ディレクトリ下にある文書
- Unix スタイルのマニュアルページ: `"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: http://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
- <http://tldp.org/> にある Linux 文書プロジェクト (TLDP) の HOWTO 文書
- <http://www.unix.org/> にある 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

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次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ムではほとんど見かけません。

All command examples are run under the English locale "LANG=en_US.UTF8". Please don't expect the placeholder strings such as *command-in-root-account* and *command-in-user-account* to be translated in command examples. This is an intentional choice to keep all translated examples to be up-to-date.

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。(対応部分を文切り型の名詞句的表現に訳しています。)

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ー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はダブルクォーテー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。読者は次の様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。読者は次の様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ー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。読者は次の様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ー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。読者は次の様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ー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。読者は次の様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。読者は次の様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ー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/ba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。読者は次の様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ー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。読者は次の様にタイプして情報を得るように心がけて下さい。

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

ポップコン

ポッ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ています。それがダウンロードされた日付は 2021-08-14 17:04:37 UTC で、183707 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 196036 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 63210 つのパッケージしか含みません。ポッ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:" が前についたポップコンの数は"1000 * (PC で最近実行されたパッケージに関するポップコン提出)/(全ポップコン提出)" として計算される。

"installs" を意味する"I:" が前についたポップコンの数は"1000 * (PC にインストールされているパッケージに関するポップコン提出)/(全ポップコン提出)" 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はありません。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"/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しょう
- パスワードとセキュリティキーを保護する
- [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しょう
 - 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
- 常に代替ブートメディア (USB メモリースティック、CD、…) を確保しましょう

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>

ウィキペディアの["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。円滑なスタートができるように、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

Upon starting the system, you are presented with the character based login screen if you did not install any GUI environment such as GNOME or KDE desktop system. Suppose your hostname is foo, the login prompt looks as follows.

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 11 foo tty1

foo login: penguin
Password:
Linux foo 5.10.0-6-amd64 #1 SMP Debian 5.10.28-1 (2021-04-09) 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: Thu May 13 08:47:13 JST 2021 on tty1
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。

You can gain the shell prompt under GUI environment by starting a `x-terminal-emulator` program such as `gnome-terminal(1)`, `rxvt(1)` or `xterm(1)`. Under the GNOME Desktop environment, press SUPER-key (Windows-key) and typing in "terminal" to the search prompt does the trick.

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次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・書込み・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 のパスワードを他人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 等のハードウェアデバイスも含む) 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

When your desktop menu does not start GUI system administration tools automatically with the appropriate privilege, you can start them from the root shell prompt of the terminal emulator, such as `gnome-terminal(1)`, `rxvt(1)`, or `xterm(1)`. See [項1.1.4](#) and [項7.8](#).



警告

gdm3(1) 等のディスプレイ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。クリティカルな情報が表示されている際には、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リモートの信頼できない 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。通常仮想コンソール 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アクティビティを終了できます。文字コンソールの場合は、こうすると login プロンプトに戻ります。これらのコントロール文字は通常 "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フォーマンス向上のためにメモリーへの**データのキャッシュ**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。)

通常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 まとも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) を使って次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
...
```

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。

パッケージ	ポップコン	サイズ	説明
mc	V:55, I:234	1492	テキストモードの全画面ファイルマネージャー
sudo	V:597, I:809	4589	ユーザーに限定的な root 権限を与えるプログラム
vim	V:102, I:404	3286	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:57, I:969	1574	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:4, I:19	18364	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:25, I:252	2367	テキストモード WWW ブラウザー
gpm	V:11, I:16	540	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

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次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	ポップコン	サイ ズ	説明
doc-debian	1:850	166	Debian プロジェクトの文書、(Debian FAQ) 他
debian-policy	1:29	4306	Debian ポリシーマニュアルと関連文書
developers-reference	1:6	1916	Debian 開発者のためのガイドラインと情報
debmake-doc	1:0	10989	Debian メンテナ向けガイド
debian-history	1:1	4285	Debian プロジェクトの歴史
debian-faq	1:846	817	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

次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成できます。root シェルプロンプトで次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は次のようなすれば削除できます。

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

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次のような単純な 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、次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ください。



警告
システムセキュリティ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へ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を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。

何故なら、たとえデフォルトのインストール後でも、Debian システムには適正なファイルパーミッションが設定されていて、非特権ユーザーがシステムに損害を与えられないようになっているからです。もちろん悪用可能な穴が残っているかもしれませんが、こんな問題まで心配する人はこのセクションを読んでいる場合ではなく、[Securing Debian Manual](#) を読むべきです。

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、共有・非共有・従属・バインド不可としてファイルシステムを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ァイルシステムの根 (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・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"` と読めます) 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、次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、次の完全に記述したファイル名の例をご覧ください: `"/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/init.d/"` 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、通常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](#)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) という言葉は `"root ユーザー"` という意味でも `"ルートディレクトリー"` とい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

パスとい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/filesystem.hierarchy.7"` や `hier(7)`) に記述されています。手始めとして次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
<code>/</code>	ルートディレクトリー
<code>/etc/</code>	システム全体の設定ファイル
<code>/var/log/</code>	システムのログファイル
<code>/home/</code>	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**は次の3つの影響されるユーザーのカテゴリ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” と呼ばれる、全他ユーザー (**other**) (**o**)

ファイルでは、それぞれに対応するパーミッションは次のようになります。

- 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次のようになります。

- 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を実行すると、次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 (次の9文字。ユーザーとグループと他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

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

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次の通り。

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

例えば次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**)
- スティッキビット (他ユーザーの **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**ファイル生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、同じ名前でも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
```

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。

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、読み出し (read) (=4) と書き込み (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	ユーザーに関して、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

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、次を試してみてください:

```
$ 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 builtins(7) をご覧ください。

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

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。)

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

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

特定のユーザーにグループ許可を適用するには、/etc/group に関しては "sudo vigr" と /etc/gshadow に関しては "sudo vigr -s" を用いて、そのユーザーをグループのメンバーにする必要があります。新規のグループ設定を有効にするにはログアウト後ログイン (もしくは "exec newgrp" を実行) する必要があります。

注意

もし "auth optional pam_group.so" 行が "/etc/pam.d/common-auth" に書き加えられ、"/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章参照下さい。)

ハードウェアデバイスは Debian システム上では一種のファイルでしかありません。CD-ROM や USB メモリースティック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 によって作成されていると、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	パスワード無しに sudo を実行
lpadmin	プリンターのデータベースからプリンターを追加・変更・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/base-passwd” 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通常使用状況下でのファイルシステムの読み出しパ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。

- [ハードリンク](#)
 - 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次の例をご覧下さい。

```
$ 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 "`rw-rwxrwx`" 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付きません。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”|”というシンタックスが提供する機能のように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、次を試してみてください:

```
$ 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。ソケットは名前付きパイプ (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であ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

”netstat -an” 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、ディスプレイ、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: ハードディスク等

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

通常のプリンターへのアクセスは lp(1) を使います。

次のように 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。

最近の 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。

以上はシェルのリディ렉션とともによく使われます。(項1.5.8参照下さい)。

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	暗号的にセキュア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

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い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起動 / 設定ファイル”/etc/sysctl.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” ディレクトリー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/filesystems” 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 パッケージを次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 システムを探索するために 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索します。

- ”/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
- ”/sbin” と”/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 は次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

In some consoles such as `gnome-terminal(1)`, key strokes of function-keys may be stolen by the console program. You can disable these features in "Preferences" → "General" and "Shortcuts" menu for `gnome-terminal`.

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。

それでも MC の表示の問題が解消しない際には、[項9.5.6](#)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 で通常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 のコマンドライントリック

- 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、カーソルの位置に挿入されます。F2 を押すとファイルをセーブします。F10 を押すと選択範囲はなくなります。ほとんどのカーソルキーは直感的に働きます。

このエディターは次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られます。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” に設定するのの一計です。

```
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。このビューワーは次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
```

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进行处理します (項9.4.11参照下さい)。これは非常に便利な MC の機能です。

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	ウェブブラウザに内容をパイプ
”*.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

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 FTP 仮想ファイルシステム

MC を Internet 越しでの FTP を用いた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、”p” を押して FTP 仮想ファイルシステムを有効にします。”username:passwd@hostname.domainname”の形式で URL を入力すると、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。

”[deb.debian.org/debian]” を URL として Debian アーカイブを閲覧します。

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実行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

ログインシェルは chsh(1) を使えば選択できます。

ティップ

POSIX-ライクなシェルは基本シンタックスはにっていますが、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ください。

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。

パッケージ	ポプコン	サイズ	POSIX シェル	説明
bash	V:796, I:999	6470	はい	Bash : GNU Bourne Again SHell (デファクトスタンダード)
bash-completion	V:31, I:922	1523	N/A	programmable completion for the bash shell
dash	V:913, I:993	221	はい	Debian の Almquist シェル 、シェルスクリプトに好適
zsh	V:36, I:74	2467	はい	Z shell : 多くの拡張された標準シェル
tcsh	V:8, I:28	1316	いいえ	TENEX C Shell : 拡張バージョンの Berkeley csh
mksh	V:7, I:12	1478	はい	Korn シェル の 1バージョン
csh	V:2, I:8	343	いいえ	OpenBSD の C シェル、 Berkeley csh の派生
sash	V:0, I:6	1090	はい	組み込みコマンド付きの 独立シェル (標準の”/bin/sh” には不向き)
ksh	V:2, I:15	3284	はい	真の AT&T バージョンの Korn シェル
rc	V:0, I:1	169	いいえ	AT&T Plan 9 の rc シェル の実装
posh	V:0, I:0	190	はい	ポリシー準拠の通常シェル (pdksh の派生)

Table 1.13: シェルプログラムのリスト

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、次を試してみてください。

```
# 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。通常の 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に配慮下さい。次に特記すべき暗記す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	カーソルの前の 1 行を消去
Ctrl-H	カーソルの前の 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

Mouse operations for text on Debian system mix 2 styles with some twists:

- 伝統的な Unix 流のマウス操作:
 - use 3 buttons (click)
 - use PRIMARY
 - used by X applications such as xterm and text applications in Linux console
- 現代的な GUI 流のマウス操作:
 - use 2 buttons (drag + click)
 - use PRIMARY and CLIPBOARD
 - used in Modern GUI applications such as gnome-terminal

Here, the PRIMARY selection is the highlighted text range. Within the terminal program, Shift-Ctrl-C is used instead to avoid terminating a running program.

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。

In order to use a mouse in Linux character consoles, you need to have gpm(8) running as daemon.

アクション	反応
マウスの左クリックアンドドラッグ	select range as PRIMARY selection
左クリック	select the start of range for PRIMARY selection
右クリック (伝統的)	select the end of range for PRIMARY selection
右クリック (現代的)	context dependent menu (cut/copy/paste)
ミドルクリックまたは Shift-Ins	insert PRIMARY selection at the cursor
Ctrl-X	cut PRIMARY selection to CLIPBOARD
Ctrl-C (ターミナルで Shift-Ctrl-C)	copy PRIMARY selection to CLIPBOARD
Ctrl-V	paste CLIPBOARD at the cursor

Table 1.15: List of mouse operations and related key actions on Debian

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” を押すみましょう。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。

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エーション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、過剰な機能があります。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奨します。

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トリングプログラムが普通付いてきます。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。カーソルを “|tutor|” に移動し Ctrl-J 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 や他のエグ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奨します。

Debian ではシステムのデフォ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” を初心者におすすめ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 Using vim

The recent vim(1) starts itself in the sane ”`nocompatible`” option and enters into the `NORMAL` mode.¹

mode	key strokes	アクション
NORMAL	:help only	display the help file
NORMAL	:e filename.ext	open new buffer to edit filename.ext
NORMAL	:w	overwrite current buffer to the original file
NORMAL	:w filename.ext	write current buffer to filename.ext
NORMAL	:q	quit vim
NORMAL	:q!	force to quit vim
NORMAL	:only	close all other split open windows
NORMAL	:set nocompatible?	check if vim is in the sane nocompatible mode
NORMAL	:set nocompatible	set vim to the sane nocompatible mode
NORMAL	i	enter the INSERT mode
NORMAL	R	enter the REPLACE mode
NORMAL	v	enter the VISUAL mode
NORMAL	V	enter the linewise VISUAL mode
NORMAL	Ctrl-V	enter the blockwise VISUAL mode
except TERMINAL - JOB	ESC-key	enter the NORMAL mode
NORMAL	:term	enter the TERMINAL - JOB mode
TERMINAL - NORMAL	i	enter the TERMINAL - JOB mode
TERMINAL - JOB	Ctrl-W N (or Ctrl-\ Ctrl-N)	enter the TERMINAL - NORMAL mode
TERMINAL - JOB	Ctrl-W :	enter the Ex-mode in TERMINAL - NORMAL mode

Table 1.16: List of basic Vim key strokes

Please use the ”`vimtutor`” program to learn vim through an interactive tutorial course.

The vim program changes its behavior to typed key strokes based on **mode**. Typing in key strokes to the buffer is mostly done in `INSERT`-mode and `REPLACE`-mode. Moving cursor is mostly done in `NORMAL`-mode. Interactive selection is done in `VISUAL`-mode. Typing ”:” in `NORMAL`-mode changes its **mode** to `Ex`-mode. `Ex`-mode accepts commands.

ティップ

The Vim comes with the **Netrw** package. Netrw supports reading files, writing files, browsing directories over a network, and local browsing! Try Netrw with ”vim .” (a period as the argument) and read its manual at ”:help netrw”.

For the advanced configuration of vim, see 項9.2.

¹Even the older vim can starts in the sane ”`nocompatible`” mode by starting it with the ”-N” option.

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いかもしれません。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

The new Vim (version>=8.2) can be used to record the shell activities cleanly using TERMINAL - JOB-mode. See [項1.4.8](#).

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、次を試してみてください:

```
$ 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い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次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 `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名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、次を試してみてください:

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

コマンド	説明
<code>pwd</code>	カレント / ワーキングディレクトリーの名前を表示
<code>whoami</code>	現在のユーザー名を表示
<code>id</code>	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
<code>file foo</code>	"foo" ファイルのファイルタイプを表示
<code>type -p commandname</code>	"commandname" コマンドのファイルの位置を表示
<code>which commandname</code>	,,
<code>type commandname</code>	"commandname" コマンドに関する情報を表示
<code>apropos key-word</code>	"key-word" に関連したコマンドを発見
<code>man -k key-word</code>	,,
<code>whatis commandname</code>	"commandname" コマンドに関する 1 行の説明を表示
<code>man -a commandname</code>	"commandname" コマンドに関する説明を表示 (Unix スタイル)
<code>info commandname</code>	"commandname" コマンドに関する比較的長い説明を表示 (GNU スタイル)
<code>ls</code>	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)
<code>ls -a</code>	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)
<code>ls -A</code>	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)
<code>ls -la</code>	ディレクトリーの内容を詳細情報とともにリスト
<code>ls -lai</code>	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
<code>ls -d</code>	現ディレクトリーの中の全ディレクトリーをリスト
<code>tree</code>	ファイルツリーの内容を表示
<code>lsuf foo</code>	"foo" ファイルのオープンの状態をリスト
<code>lsuf -p pid</code>	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
<code>mkdir foo</code>	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
<code>rmdir foo</code>	現ディレクトリー中の "foo" というディレクトリーを削除
<code>cd foo</code>	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
<code>cd /</code>	ディレクトリーをルートディレクトリーに変更
<code>cd</code>	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
<code>cd /foo</code>	絶対ディレクトリーパス "/foo" にディレクトリーを変更
<code>cd ..</code>	親ディレクトリーにディレクトリーを変更
<code>cd ~foo</code>	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
<code>cd -</code>	一つ前のディレクトリーにディレクトリーを変更
<code></etc/motd pager</code>	"/etc/motd" の内容をデフォルトのページャーで表示
<code>touch junkfile</code>	空ファイル "junkfile" を作成
<code>cp foo bar</code>	既存のファイル "foo" を新規ファイル "bar" にコピー
<code>rm junkfile</code>	ファイル "junkfile" を削除
<code>mv foo bar</code>	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)
<code>mv foo bar</code>	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)
<code>mv foo bar/baz</code>	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)
<code>chmod 600 foo</code>	既存のファイル "foo" を他人から読出し不可かつ書込み不可 (全員実行不可)
<code>chmod 644 foo</code>	既存のファイル "foo" を他人からは読出し可だが書込み不可 (全員実行不可)
<code>chmod 755 foo</code>	既存のファイル "foo" を他人からは読出し可だが書込み不可 (全員実行可能)
<code>find . -name pattern</code>	シェルで "pattern" にマッチするファイル名を探索 (比較的遅い)
<code>locate -d . pattern</code>	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ベースを使い比較的早い)
<code>grep -e "pattern" *.html</code>	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

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が。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 的なコマンドは次のように (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します。

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

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、次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。

環境変数のデフォルト値は PAM システムが初期設定されます。その後次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。

- The PAM system such as `pam_env` may set environment variables by `/etc/pam.conf`, `/etc/environment` and `/etc/default/locale`.
- `gdm3` のようなディスプレイマネージャーは GUI セッション向けの環境変数を再設定します。
- The user specific program initialization may reset environment variables by `~/.profile`, `~/.bash_profile` and `~/.bashrc`.

1.5.2 "\$LANG" 変数

The default locale is defined in the `"$LANG"` environment variable and is configured as `"LANG=xx_YY.UTF-8"` by the installer or by the subsequent GUI configuration, e.g., "Settings" → "Region & Language" → "Language" / "Formats" for GNOME.

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勧めします。

`"$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"`。

典型的なコマンドの実行は次のようなシェルの行シーケンスを用います。

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

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頭に環境変数定義をつけないのが普通です。上記の例の代わりに次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"/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"/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"/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	通常の root シェルから実行されるプログラム
/home/normal_user	通常のユーザーシェルから実行されるプログラム
/home/normal_user	通常のユーザーの 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。

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。

例えば、次を試してみてください:

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 (部分) 名
.*	". " で始まるファイル (部分) 名
?	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

```
$ mkdir junk; cd junk; touch 1.txt 2.txt 3.c 4.h .5.txt ..6.txt
$ echo *.txt
1.txt 2.txt
$ echo *
1.txt 2.txt 3.c 4.h
$ echo *.hc]
3.c 4.h
$ echo .*
. .5.txt ..6.txt
$ echo .*[^.]*
.5.txt ..6.txt
$ echo [^1-3]*
4.h
$ cd ../; rm -rf junk
```

glob(7) を参照下さい。

注意
通常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、次を試してみてください。

```
$ [ 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に注意して下さい。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렉션

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
<code>command &</code>	<code>command</code> をサブシェル中でバックグラウンド実行
<code>command1 command2</code>	<code>command1</code> の標準出力を <code>command2</code> の標準入力にパイプ (同時並行で実行)
<code>command1 2>&1 command2</code>	<code>command1</code> の標準出力と標準エラー出力を <code>command2</code> の標準入力にパイプ (同時進行で実行)
<code>command1 ; command2</code>	<code>command1</code> を実行し、後に続いて <code>command2</code> を実行
<code>command1 && command2</code>	<code>command1</code> を実行; もし成功したら、後に続いて <code>command2</code> を実行 (<code>command1</code> と <code>command2</code> の両方が成功したら、正常終了を返す)
<code>command1 command2</code>	<code>command1</code> を実行; もし成功しなかったら、後に続いて <code>command2</code> を実行 (<code>command1</code> か <code>command2</code> のどちらかが成功したら、正常終了を返す)
<code>command > foo</code>	<code>command</code> の標準出力を <code>foo</code> ファイルにリダイレクト (上書き)
<code>command 2> foo</code>	<code>command</code> の標準エラー出力をファイル <code>foo</code> にリダイレクト (上書き)
<code>command >> foo</code>	<code>command</code> の標準出力をファイル <code>foo</code> にリダイレクト (追記)
<code>command 2>> foo</code>	<code>command</code> の標準エラー出力を <code>foo</code> ファイルにリダイレクト (追記)
<code>command > foo 2>&1</code>	<code>command</code> の標準出力と標準エラー出力を <code>foo</code> ファイルにリダイレクト
<code>command < foo</code>	<code>command</code> の標準入力を <code>foo</code> ファイルからリダイレクト
<code>command << delimiter</code>	<code>command</code> の標準入力を "delimiter" に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)
<code>command <<- delimiter</code>	<code>command</code> の標準入力を "delimiter" に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) マニュアル中の "SIGNALS" と "JOB CONTROL" セクションや `builtins(1)` を参照下さい。

例えば、次を試してみてください:

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

ファイルディスクリプタの 0-2 は事前定義されています。

デバイス	説明	ファイルディスクリプタ
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ィスクリプタ

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、次を試してみてください:

```
$ 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、次を試してみてください:

```
$ 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)`。
 - `pcregrep` パッケージの `pcregrep(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 `python(1)` は考え得る全てのテキスト処理をします。”`/usr/share/doc/python/` 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** がついた通常のロケール (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。

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。

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

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いますが、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、次を試してみてください:

```
$ 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。

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、次を試してみてください:

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

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

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) コマンドは次のようにすると”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) コマンドは次のようにすると”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) コマンドを使い次のようにすると”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' '+w' '+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/ge|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、次を試してみてください:

```
$ 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、次を試してみてください:

```
$ 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釈できます。

```
$ 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

次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](#)参照下さい)。

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

Debian は、フリーソフトウェア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。

The current Debian package management system which can utilize all these resources is [Advanced Packaging Tool \(APT\)](#).

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きます。現在、amd64 アーキテクチャーでは 63210 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・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。
- インストールされた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 Debian パッケージ管理の前提条件

2.1.1 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。

- 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、パッケージの初期インストールプロセスを助けるための 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し、それぞれ毎の設定スクリプトとともに提供されます。
-

パッケージ	ポップコン	サイズ	説明
dpkg	V:932, I:999	6874	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:887, I:999	4337	APT front-end to manage packages with CLI: apt / apt-get / apt-cache
aptitude	V:72, I:398	4256	APT front-end to interactively manage packages with full screen console: aptitude (8)
tasksel	V:33, I:975	395	APT front-end to install selected tasks: tasksel (8)
unattended-upgrades	V:326, I:447	326	セキュリティアップデート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:112, I:203	6559	Software Center for GNOME (GUI APT front-end)
synaptic	V:39, I:308	7873	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:379, I:997	1164	APT ユーティリティプログラム: apt-extracttemplates (1) と apt-ftparchive (1) と apt-sortpkgs (1)
apt-listchanges	V:356, I:849	423	パッケージ変更履歴の通知ツール
apt-listbugs	V:8, I:12	465	APT による各インストール前にクリティカルバグをリストする
apt-file	V:17, I:79	90	APT パッケージ探索ユーティリティ -- コマンドラインインターフェース
apt-rdepends	V:0, I:6	40	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

- 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の全機能が利用可能です。ただしセキュリティリスクのある機能はデフォ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。

2.1.2 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つと破壊されます。

[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アップデート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、用心として次の有効なアクションですら避けておくべきと考えます。次は留意点です。

- ”/etc/apt/sources.list”の中にテスト版 **testing** とか不安定版 **unstable** とかを含めません。
- ”/etc/apt/sources.list”の中に標準の Debian と Debian 以外の Ubuntu のようなアーカイブを混在させません。

- `"/etc/apt/preferences"` を作成しません。
- パッケージ管理ツールのデフォルトに影響を理解せずに変更しません。
- ランダムなパッケージを `"dpkg -i random_package"` でインストールしません。
- 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ません。
- `"/var/lib/dpkg/"` の中のファイルを消去や改変しません。
- ソースから直接コンパイルしたソフトウェア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ません。
 - 必要な場合は `"/usr/local/"` か `"/opt/"` 中にインストールします。

上記のアクションで起きる Debian パッケージシステムへのコンパチブルでない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アップデート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。

2.1.3 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、セキュリティアップデートをした安定版 `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。

Despite my warnings above, I know many readers of this document may wish to run the newer `testing` or `unstable` suites.

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。

This list is targeted for the **self-administered** Desktop environment.

- Use the `testing` suite since it is practically the rolling release automatically managed by the Debian archive QA infrastructure such as the [Debian continuous integration](#), the [source only upload practices](#), and the [library transition tracking](#). The packages in the `testing` suite are updated frequently enough to offer all the latest features.
- Set the codename corresponding to the `testing` suite (currently `"bookworm"`) in the `"/etc/apt/sources.list"`.
- Manually update this codename in the `"/etc/apt/sources.list"` to the new one only after assessing situation by yourself for about a month after the major suite release. The Debian user and developer mailing list are good sources of information for this, too.

The use of the `unstable` suite isn't recommended. The `unstable` suite is **good for debugging packages** as a developer but tends to expose you to unnecessary risks for the normal Desktop usage. Even though the `unstable` suite of the Debian system looks very stable for most of the times, there have been some package problems and a few of them were not so trivial to resolve.

Here are some basic precautionary measure ideas to ensure quick and easy recovery from bugs in Debian packages.

- 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

- レスキューブートのためのインストール用 CD を手元に確保
- `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
- Install a corresponding sandboxed upstream binary package in case of trouble (see [項7.6](#))
- `chroot` が類似の環境を作り事前に最新のシステムを実行 ([項9.11](#)参照下さい)

**注意**

If you can not do any one of these precautionary actions, you are probably not ready for the testing and unstable suites.

2.1.4 Debian アーカイブの基本

[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。

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。

典型的な HTTP アクセスの場合、現在の安定版 `stable=bullseye` システムを例にとると、次の様に `/etc/apt/sources` ファイルの中にアーカイブは規定されています。

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

deb http://security.debian.org/debian-security bullseye-security main contrib
deb-src http://security.debian.org/debian-security bullseye-security main contrib
```

上記で、次の安定版 `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"` でなくコード名の `"bullseye"` を使います。

`"/etc/apt/sources.list"` の意味は `sources.list(5)` に記載されていて、要点は以下です。

- `"deb"` 行がバイナリーパッケージのための定義です。
- `"deb-src"` 行がソースパッケージのための定義です。
- 一番目の引数は、Debian アーカイブの root URL です。
- 二番目の引数は、スイーツ名かコード名のどちらかで与えられるディストリビューション名です。
- 三番目次の引数は、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です。

ソース関連のメタデータにアクセスしない `aptitude` のためだけなら `"deb-src"` 行は安全に省略 (もしくは `"#"` を行頭に挿入してコメントアウト) 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の更新速度が向上します。URL は `"http://"` や `"ftp://"` や `"file://"` 等々の何れも可能です。

ティップ

もし上記の例で `"bullseye"` でなく `"sid"` が使われる場合には、セキュリティアップデートのための `"deb: http://security.debian.org/ ..."` 行は不要です。安定版 `stable` とテスト版 `testing` (即ち `bullseye` と `bookworm`) にのみセキュリティアップデートがあります。 `"sid"` (不安定版 `unstable`) のためにはセキュリティアップデートのアーカイブが存在しません。

アーカイブの URL	スイート名 (コード名)	目的
http://deb.debian.org/debian/	stable (bullseye)	安定版 (bullseye) のリリース
http://deb.debian.org/debian/	testing (bookworm)	テスト版 (bookworm) のリリース
http://deb.debian.org/debian/	unstable (sid)	不安定版 (sid) のリリース
http://deb.debian.org/debian/	experimental	実験的プリリリース (任意、開発者専用)
http://deb.debian.org/debian/	stable-proposed-updates	次回安定版ポイントリリース用のアップデート (任意)
http://security.debian.org/debian-security/	bullseye-security	安定版用のセキュリティアップデート (重要)
http://security.debian.org/debian-security/	testing-security	テスト版用のセキュリティアップデート (重要)
http://deb.debian.org/debian/	bullseye-updates	bullseye のためのスパムフィルターや IM クライアント他用のコンパチブルなアップデート
http://deb.debian.org/debian/	bullseye-backports	bullseye のための新しくバックポートされたパッケージ (任意)

Table 2.2: Debian アーカイブサイトのリスト

次は設定ファイル内に用いられる 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。

注意



セキュリティアップデート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** リリースを走らせることは、純粋な **unstable** リリースを走らせ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、[bullseye-updates](#) や <http://backports.debian.org> (項2.7.4参照下さい) 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2.7.3) のみ複数のリストに意味があります。

ティップ

stable や testing スイーツの Debian システムでは、上記の例のようにセキュリティアップデートを有効とするように "/etc/apt/sources.list" の中に "<http://security.debian.org/>" の行を含めることはいいことです。

注意

stable アーカイブのセキュリティバグは Debian のセキュリティ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バグは Debian の testing セキュリティ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修正を使うことで通常比較的良好な状態です。Debian がセキュリティ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	62089	DFSG に完全準拠し、non-free のパッケージに非依存 (main = 主要)
contrib	351	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	770	非 DFSG 準拠

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2.1.5参照)。

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い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次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
bullseye リリース後	コード名 = bullseye	コード名 = bookworm	コード名 = sid
bookworm リリース後	コード名 = bookworm	コード名 = trixie	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コンパチブルであるようにする義務が 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 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コンパチブルであるようにするように努めます。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

たとえいつも非安定版 unstable やテスト版 testing アーカイブを使っていようと、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アップデートされた安定版 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サポートはありません ([Debian testing-security-announce 2008-12](#))。1ヶ月ほど経てば、非安定版 unstable アーカイブなら注意を払えば使えるかもしれません。

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](#)”

2.1.5 Debian は 100% フリーソフトウェアです

Debian は以下の理由で 100% フリーソフトウェアです:

- 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のみを提供します。
- Debian は main からのフリーソフトウェア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 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はありません。

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であり続けます」。 [Debian 社会契約](#)の第一項)
- Debian サーバーは 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 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コミュニティ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「contrib」と「non-free」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。

ユーザーは non-free 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をサポートできません。)
- あなたの 100% フリーソフトウェアの Debian システムへの汚染

[Debian フリーソフトウェアガイドライン](#) は [Debian](#) のフリーソフトウェア基準です。Debian は「ソフトウェア」に関して、パッケージ中の文書、ファームウェア、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基準は非常に厳格なものとなります。

典型的な non-free 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 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ェアパッケージ。(主に non-free/kernel セクション中にある)
- 商用使用やコンテンツ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 contrib パッケージの数は main パッケージの数の 2% 以下です。non-free や contrib エリアへのアクセスを有効にしてもパッケージ起源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。

2.1.6 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

正常な設定として"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あります。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。

2.1.7 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次のようになります。

- 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が選ばれる候補バージョンを選択 (例外については項2.7.3参照下さい)
 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。

2.1.8 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/package_name/`” 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、次のようにタイプして下さい。

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

ウェブサイト	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	<code>sensible-browser "http://bugs.debian.org/"</code>
既知のパッケージに関するバグレポート	<code>sensible-browser "http://bugs.debian.org/package_name"</code>
既知のバグ番号に関するバグレポート	<code>sensible-browser "http://bugs.debian.org/bug_number"</code>

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ブサイトのリスト

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。

“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。

2.2 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用できます。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奨されません。
 - それには、“`apt full-upgrade`” か “`apt-get dist-upgrade`” を使うことが推奨されます。[Bug #411280](#) 参照ください。
- `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。パニックしないでください。
 - 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・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` はハードウェ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` コマンドは最も多様な 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` はインストールされた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` はハードウェ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` and `aptitude` can be mixed without major troubles.

`"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。

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フィーチャーとともに提供されますが、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` コマンドを使ってください。

`"aptitude"` のすぐ後ろにコマンドオプションをつけられます。

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

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
N/A	aptitude search '~i!~M'	apt-mark showmanual	手動インストールされた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

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

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次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。

キー	キーバインディング
F10 もしくは 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

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イ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アップグレードされます。

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次の例のように表示されます。

```
idA      libsmclient      -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が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2.7.3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。

表示	ビューの説明
パッケージ画面	表 2.10参照 (デフォルト)
推奨を監査	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
平坦なパッケージリスト	パッケージを分類せずにリスト (regex とともに使用)
Debtags 表示	パッケージの debtags のエントリーにより分類したパッケージをリスト
ソースパッケージ画面	ソースパッケージごとに分けてパッケージの一覧を表示

Table 2.9: aptitude の表示のリスト

注意
パッケージの debtags によるタグ付け状況を改善するのにご協力下さい！

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ろ取りするのに使えます。

分類	ビューの説明
更新可能なパッケージ	section → area → package と整理してパッケージをリスト
新規パッケージ	,,
インストール済みのパッケージ	,,
インストールされていないパッケージ	,,
廃止された、またはローカルで作成されたパッケージ	,,
仮想パッケージ	同一機能のパッケージをリスト
タスク	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。
- 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。
- デフォルトの依存関係は `"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。

拡張マッチ規則の説明	regex 式
パッケージ名とのマッチ	<code>~n</code> 名前の <i>regex</i>
記述とのマッチ	<code>~d</code> 記述の <i>regex</i>
タスク名とのマッチ	<code>~t</code> タスクの <i>regex</i>
debtags とのマッチ	<code>~Gdebtags</code> の <i>regex</i>
メンテナとのマッチ	<code>~mmaintainer</code> の <i>regex</i>
パッケージセクションとのマッチ	<code>~s</code> セクションの <i>regex</i>
パッケージバージョンとのマッチ	<code>~v</code> バージョンの <i>regex</i>
アーカイブ (archive) とのマッチ	<code>~A{bullseye,bookworm,sid}</code>
オリジン (origin) とのマッチ	<code>~O{debian,...}</code>
優先度 (priority) とのマッチ	<code>~p{extra,important,optional,required,standard}</code>
必須 (essential) パッケージとのマッチ	<code>~E</code>
仮想パッケージとのマッチ	<code>~v</code>
新規パッケージとのマッチ	<code>~N</code>
次のアクションとのマッチ	<code>~a{install,upgrade,downgrade,remove,purge,hold,keep}</code>
インストール済みパッケージとのマッチ	<code>~i</code>
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)	<code>~M</code>
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)	<code>~i!~M</code>
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	<code>~U</code>
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	<code>~C</code>
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	<code>~g</code>
壊れた依存関係宣言をしたパッケージとマッチ	<code>~b</code>
<i>type</i> の壊れた依存関係を宣言しているパッケージとマッチ	<code>~Btype</code>
<i>type</i> の壊れた依存関係を宣言している <i>pattern</i> パッケージとマッチ	<code>~D[type:]pattern</code>
<i>type</i> の壊れた依存関係を宣言している <i>pattern</i> パッケージとマッチ	<code>~DB[type:]pattern</code>
<i>pattern</i> マッチするパッケージが <i>type</i> の依存関係を宣言しているパッケージとマッチ	<code>~R[type:]pattern</code>
<i>pattern</i> マッチするパッケージが <i>type</i> の壊れた依存関係を宣言しているパッケージとマッチ	<code>~RB[type:]pattern</code>
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	<code>~R~i</code>
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	<code>!~R~i</code>
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	<code>~R~i ~Rrecommends:~i</code>
フィルターされたバージョンの <i>pattern</i> とマッチ	<code>~S filter pattern</code>
常に全てのパッケージにマッチ (真)	<code>~T</code>
どのパッケージにもマッチしない (偽)	<code>~F</code>

Table 2.11: aptitude の regex 式のリスト

- `"~Pterm" == "~Dprovides:term"`
- `"~Cterm" == "~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なく”Recommends:” リストに規定されたパッケージも引き込みます。このような自動的にインストールされた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 パッケージアクティビティログ

パッケージアクティビティの履歴はログファイルで確認できます。

ファイル	内容
<code>/var/log/dpkg.log</code>	全パッケージアクティビティの dpkg レベルのアクティビティのログ
<code>/var/log/apt/term.log</code>	APT アクティビティのログ
<code>/var/log/aptitude</code>	aptitude コマンドアクティビティのログ

Table 2.12: パッケージアクティビティのログファイル

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 regex にマッチするパッケージ名のパッケージをリスト

次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'
p libnss-ldap - NSS module for using LDAP as a naming service
p libpam-ldap - Pluggable Authentication Module allowing LDAP interfaces
```

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。

2.3.2 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。

2.3.3 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次のようにして完全削除できます。

次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、次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。その見出しの下で 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。

2.3.4 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、自動インストールとなるよう”M” と”インストール済みのパッケージ”の上でタイプします。
4. パッケージ表示制限を”~prequired|~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”インストール済みのパッケージ”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、”インストール済みのパッケージ”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。

2.3.5 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10.2参照下さい) を事前にしておくべきです。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勧めします。

`/etc/apt/sources.list` ファイルの内容を新規リリースへと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` 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2.1.4にある `/etc/apt/sources.list` の `"bullseye"` を `"bookworm"` 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、インストールされた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. `/etc/apt/sources.list` ファイル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。

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。

**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。

以下に注意下さい。

- 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 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”/etc/apt/sources.list”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。

コマンド	アクション
<code>COLUMNS=120 dpkg -l パッケージ名パターン</code>	バグレポートのためにインストールされたパッケージの状態をリスト
<code>dpkg -L パッケージ名</code>	インストールされたパッケージの内容をリスト
<code>dpkg -L パッケージ名 egrep '/usr/share/man/man.*/.+'</code>	インストールされたパッケージのマニュアルページをリスト
<code>dpkg -S ファイル名パターン</code>	マッチするファイル名があるインストールされたパッケージをリスト
<code>apt-file search ファイル名パターン</code>	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
<code>apt-file list パッケージ名パターン</code>	アーカイブ中のマッチするパッケージをリスト
<code>dpkg-reconfigure パッケージ名</code>	特定パッケージを再設定
<code>dpkg-reconfigure -plow パッケージ名</code>	もっとも詳細な質問で特定パッケージを再設定
<code>configure-debian</code>	フルスクリーンメニューからパッケージを再設定
<code>dpkg --audit</code>	部分的にインストールされたパッケージに関してシステムを監査
<code>dpkg --configure -a</code>	全ての部分的にインストールされたパッケージを設定
<code>apt-cache policy バイナリーパッケージ名</code>	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
<code>apt-cache madison パッケージ名</code>	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
<code>apt-cache showsrc バイナリーパッケージ名</code>	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
<code>apt-get build-dep パッケージ名</code>	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
<code>aptitude build-dep package_name</code>	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
<code>apt-get source パッケージ名</code>	(標準アーカイブから) ソースをダウンロード
<code>dget URL for dsc file</code>	(他のアーカイブから) ソースをダウンロード
<code>dpkg-source -x パッケージ名 _ バージョン-debian のレビジョン.dsc</code>	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
<code>debuild binary</code>	ローカルのソースツリーからパッケージをビルド
<code>make-kpkg kernel_image</code>	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
<code>make-kpkg --initrd kernel_image</code>	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
<code>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</code>	ローカルパッケージをシステムにインストール
<code>apt install /path/to/package_filename.deb</code>	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
<code>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</code>	ローカルパッケージ (複数) をシステムにインストール
<code>dpkg --get-selections '*'>selection.txt</code>	dpkg レベルのパッケージ選択状態情報を保存
<code>dpkg --set-selections <selection.txt</code>	dpkg レベルのパッケージ選択状態情報を設定
<code>echo package_name hold dpkg --set-selections</code>	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

2.4.2 インストールされた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されたパッケージファイルを検証できます。MD5sum がどのような仕組みかは項10.3.5参照下さい。

注意

侵入者によって MD5sum のデータ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の検索

最近では 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えます。

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 でインストールされた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/` ディレクトリで `grep -e regex パターン *` コマンドを実行しないとけません。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

各ディストリビューションのメタデータ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ブブラウザで閲覧できます。6 つのタイプの重要メタデータがあります。

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は保存されています。

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

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/Release>"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2.1.4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 [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”/etc/apt/trusted.gpg”に取り込むには次のようにします。
 - 最新の base-files パッケージを用いてキーリングをインストールすることで自動で取り込む。
 - ftp-master.debian.org に掲示された最新の公開アーカイブキーを gpg か apt-key ツールを用いて手動で取り込む。
- セキュアー APT システムはこの”Release.gpg”ファイルと”/etc/apt/trusted.gpg”中の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フォーマンスを保ったまま良好なセキュリティが確保できます (項10.3参照下さい)。

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.org/debian/dists/testing/main/binary-amd64/Release”等の”/etc/apt/sources.list”中の”deb”行で特定され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:”スタンプには、[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”等)が使われます。

experimental や bullseye-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/Release”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”となっていない通常のアーカイブではデフォルトの 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ルトの Pin-Priority 値は 1 です (apt_preferences(5) と項2.7.3参照下さい)。

2.5.4 パッケージメタデータの取得

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のローカルコピーを更新する必要があります。このようなローカルのコピーは”/etc/apt/sources.list”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) の名前に対応する次のファイル名です (項2.1.4参照下さい)。

- ”/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 _source_Sources”
- ”/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.gpg”
- ”/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”/etc/apt/sources.list”中に”deb”行があれば”Packages”メタデータが更新されます。もし”/etc/apt/sources.list”中に”deb-src”行があれば”Sources”メタデータが更新されます。

”Packages”や”Sources”メタデータ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:”スタンプを含んでいます。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/”ディレクトリツリーの下に置かれています。

”Packages”メタデータ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”の下で”Clean package cache”か”Clean obsolete files”メニューによって強制実行できる。

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

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。

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

Table 2.16: Debian パッケージ名の各部分に使用可能な文字

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、通常の 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。

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。

ファイル	内容の説明
/var/lib/dpkg/info/パッケージ名.conf files	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.conf files	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 コマンドが利用する代替情報
/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(8) が設定する迂回情報
/var/lib/dpkg/statoverride	dpkg(1) が利用し、dpkg-statoverride(8) 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

"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。

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/” 中のシmlinkとして保持します。選択プロセスには”/var/lib/dpkg/alternatives/” 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(8) 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のパッケージアップグレードがリセットします。

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(8) 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お薦めできません。

2.6 壊れたシステムからの復元

非安定 (unstable) システムを動かす時には、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。後日設定情報を回復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 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(1) 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2.1.6参照下さい)。

パッケージメ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2.7.2参照下さい) 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(1) 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ンス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
```

2.6.3 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、次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次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する
- 行末に”|| true” を挿入し成功を強制的に返さす

全ての部分的にインストールされたパッケージを次のコマンドで設定します。

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


2.6.4 dpkg コマンドを使っの救済

dpkg は非常に低レベルのパッケージツールなのでネットワーク接続もないブート不能な非常に劣悪な状況下でも機能します。foo パッケージが壊れていて置き換える必要があると仮定します。

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、次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
```

ティップ

システムがそれほど壊れていないなら、項2.7.10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。

ハードディスクからブートできない場合は、他の方法でのブート方法を考えるべきです。

1. Debian インストーラー (debian-installer) の CD を使ってレスキューモードでブートします。
2. ブートできないハードディスク上のシステムを `/target` にマウントします。
3. 古いバージョンの foo パッケージを次のようにしてインストールします。

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

この例は、たとえハードディスク上の dpkg コマンドが壊れていても機能します。

ティップ

ハードディスク上の別のシステムであれ、GNU/Linux のライブ CD であれ、ブート可能な USB キードライブであれ、ネットブートであれ、どのように起動された GNU/Linux システムでも同様にして壊れたシステムを救済するのに使えます。

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。

2.6.5 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勧めします。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

2.7.1 Debian パッケージの選択方法

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 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-10)
- ポブコン: 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アップデート
- 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。

2.7.2 混合した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 [bullseye-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。

このよう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する必要があります。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

壊れたパッケージを短期的に避ける場合以外では、公式にサポートされていない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一般的には賢明ではありません。たとえ apt-pinning (項 [2.7.3](#) 参照下さい) を使った場合にもこれは当てはまります。chroot や類似のテクニック (項 [9.11](#) 参照下さい) 使って、他のアーカイブからのプログラムを実行するよう検討するべきです。

2.7.3 候補バージョンの調整

**警告**

初心者のユーザーによる apt-pinning の利用は大トラブル発生を間違いなく起こします。本当に必要な時以外は apt-pinning の利用は避けなければいけません。

"/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。これが通常状態で 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2.7.2参照下さい) 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。

**注意**

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 は全く任意の操作で、著者が使うようにと勧めているわけではありません。

**注意**

アーカイブレベルの Release ファイル (項2.5.3参照下さい) 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。

**注意**

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2.6.4と項2.7.2参照下さい) 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2.7.10参照下さい)。

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。

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、そのデフォルト値が使われます。

ターゲットのリリースアーカイブは次のようにして設定できます。

- ”APT::Default-Release ”stable” ;” 行を使う”/etc/apt/apt.conf” ファイル
- ”apt-get install -t testing some-package” 等の”-t” オプションの引数

アーカイブ中のアーカイブレベルの Release ファイル (項2.5.3参照下さい) に”NotAutomatic: yes” や”ButAutomaticUpgrades: yes” が含まれると **NotAutomatic** かつ **ButAutomaticUpgrades** アーカイブと設定されます。

複数アーカイブソースの *package* に関する Pin-Priority 値は”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/ bullseye-backports/main Packages” 等としてリストされます。

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、[bullseye-updates](#) と [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/ bullseye main contrib non-free
deb http://security.debian.org/ bullseye-security main contrib
deb http://deb.debian.org/debian/ bullseye-updates main contrib non-free
deb http://deb.debian.org/debian/ bullseye-backports main 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 bullseye-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。
- bullseye-backports からインストールしされた古いパッケージのみが bullseye-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 bullseye-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次のコマンドを使います。

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

2.7.5 ”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

初心者のユーザーによる apt-pinning の利用は大トラブル発生を間違いなく起こします。本当に必要な時以外は apt-pinning の利用は避けなければいけません。

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次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
```

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



警告

初心者のユーザーによる apt-pinning の利用は大トラブル発生を間違いなく起こします。本当に必要な時以外は apt-pinning の利用は避けなければいけません。

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” ファイル中に次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/ testing-security main contrib
```

”/etc/apt/preferences” を次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) 次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アップデートのプロセスが遅くなることをさけられます。

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次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”ファイルに次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。

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

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` ファイルに次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ルトの Pin-Priority 値は、**NotAutomatic** アーカイブ (項2.5.3参照下さい) なので、常に 1 (<100) です。experimental アーカイブにある特定パッケージを次回アップグレード時に自動的に追跡しようとししない限り、`/etc/apt/preferences` ファイル中で experimental アーカイブを使うために Pin-Priority 値を明示的に設定する必要はありません。

2.7.8 パッケージの自動ダウンロードとアップグレード



警告

初心者のユーザーによる apt-pinning の利用は大トラブル発生を間違いなく起こします。本当に必要な時以外は apt-pinning の利用は避けなければいけません。

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、セキュリティアップグレードがすでに閉じたセキュリティホールからの侵入者によりシステムが壊される危険性より小さいなら、パラメーターを次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";
```

unstable システムを使っている場合には、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れるので、それはしたくないでしょう。そんな 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";
```

2.7.9 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 のパラメーターを次のように設定します。

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


2.7.10 緊急ダウングレード



警告

初心者のユーザーによる apt-pinning の利用は大トラブル発生を間違いなく起こします。本当に必要な時以外は apt-pinning の利用は避けなければいけません。



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回復処置の後に全ての重要データをバックアップし、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2.7.3参照下さい)。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2.6.4参照下さい)。

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/ bookworm main contrib non-free
```

“/etc/apt/preferences” を次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。

2.7.11 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。

2.7.12 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

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" ファイルに次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アップデートします。

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
次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きい無駄遣いですので、[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 のようなウェブ (http) プロキシサーバー (項[6.5](#)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
approx	V:0, I:0	6317	Debian アーカイブファイルのキャッシュプロキシサーバー (コンパイルされた OCaml プログラム)
apt-cacher	V:0, I:0	289	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:5, I:5	1562	ソフトウェアパッケージの頒布の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、汎用のウェブ (http) プロキシは比較的堅牢ですしそのような変化に合うのも簡単です。

2.7.15 パッケージ管理の追加参考文献

パッケージ管理に関しては次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-key(8)` と `sources.list` と `apt.conf(5)` と `apt_preferences(5)`;
 - `"/usr/share/doc/apt-doc/guide.html/index.html"` と `"/usr/share/doc/apt-doc/offline.html/index.html"` from the `apt-doc` package;
 - `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 システムが以下に起動され設定されるかの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された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。

Here is a rough overview of the key points of the Debian system initialization. Since the Debian system is a moving target, you should refer to the latest documentation.

- [Debian Linux Kernel Handbook](#) is the primary source of information on the Debian kernel.
- 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 段階のようです。各段階は次の段階の制御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しないで下さい。

3.1.1 1 段目: UEFI

The [Unified Extensible Firmware Interface \(UEFI\)](#) defines a boot manager as part of the UEFI specification. When a computer is powered on, the boot manager is the 1st stage of the boot process which checks the boot configuration and based on its settings, then executes the specified OS boot loader or operating system kernel (usually boot loader). The boot configuration is defined by variables stored in NVRAM, including variables that indicate the file system paths to OS loaders or OS kernels. An [EFI system partition \(ESP\)](#) is a data storage device partition that is used in computers adhering to the UEFI specification. Accessed by the UEFI firmware when a computer is powered up, it stores UEFI applications and the files these applications need to run, including operating system boot loaders. (On the legacy PC system, [BIOS](#) stored in the [MBR](#) may be used instead.)

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。

The Debian system normally uses the Linux kernel as the default system kernel. The [initrd](#) image for the current 5.x Linux kernel is technically the [initramfs](#) (initial RAM filesystem) image.

多くのブートローダー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:214	158	サポート	GRUB UEFI	This is smart enough to understand disk partitions and filesystems such as vfat, ext4, ... (UEFI)
grub-pc	V:26, I:757	533	サポート	GRUB 2	This is smart enough to understand disk partitions and filesystems such as vfat, ext4, ... (BIOS)
grub-rescue-pc	V:0, I:1	6380	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)
lilo	V:0, I:2	697	サポート	Lilo	ハードディスク上のセクター位置に依存します。(旧式)
syslinux	V:3, I:46	344	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:46	344	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, I:1	90	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, I:6	50	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



警告

[grub-rescue-pc](#) パッケージのイメージから作ったブート可能なレスキューメディア (CD かフロッピー) 無しにブートローダーを試してはいけません。これさえあると、ハードディスク上に機能するブートローダーが無くともシステムの起動ができます。

For GRUB 2, the menu configuration file is located at `"/boot/grub/grub.cfg"` and its key part of menu entry looks like:

```
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
}
```

For this part of `/boot/grub/grub.cfg`, this menu entry means the following.

設定	変数値
GRUB2 modules loaded	gzio, part_gpt, ext2
root file system partition used	partition identified by UUID=fe3e1db5-6454-46d6-a14c-071208ebe4b1
kernel image path in the root file system	/boot/vmlinuz-5.10.0-6-amd64
kernel boot parameter used	"root=UUID=fe3e1db5-6454-46d6-a14c-071208ebe4b1 ro quiet"
initrd image path in the root file system	/boot/initrd.img-5.10.0-6-amd64

Table 3.2: The meaning of the menu entry of the above part of `/boot/grub/grub.cfg`

ティップ

You can customize GRUB splash image by setting `GRUB_BACKGROUND` variable in `/etc/default/grub` pointing to the image file or placing the image file itself in `/boot/grub/`.

”info grub” と `grub-install(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起動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 インストローラー](#) 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次の段階にコントロールを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ェア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**

読み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 段階目: 通常の 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 段階目です。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ルトのファイルパスは `/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 `/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。

3.2 Systemd init

このセクションは PID=1 (詰まり、`init` プロセス) の `systemd(1)` プログラムがどのようにシステムを起動するのかを説明します。

The `systemd` `init` process spawns processes in parallel based on the unit configuration files (see `systemd.unit(5)`) which are written in declarative style instead of SysV-like procedural style.

The spawned processes are placed in individual [Linux control groups](#) named after the unit which they belong to in the private `systemd` hierarchy (see [cgroups](#) and [項4.7.4](#)).

The unit configuration files are loaded from a set of paths (see `systemd-system.conf(5)`) as follows:

- `/lib/systemd/system`: OS のデフォルトの設定ファイル
- `/etc/systemd/system`: OS デフォルト設定ファイルをオーバーライドするシステム管理者設定ファイル
- `/run/systemd/system`: OS デフォルト設定ファイルをオーバーライドする実行時生成される設定ファイル

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` 参照) によっても定義される。

unit 設定ファイルのサuffixにそのタイプを折込みます:

パッケージ	ポップコン	サイズ	説明
systemd	V:826, I:928	16015	並行処理のためのイベント依存の init(8) デーモン (sysvinit 代替)
systemd-sysv	V:818, I:925	139	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:687, I:940	131	sysvinit と systemd 間を切り替える補助ツール
initscripts	V:77, I:283	176	システムの始動と停止のためのスクリプト
sysvinit-core	V:7, I:9	279	System-V 的な init(8) ユーティリティ
sysv-rc	V:154, I:296	81	System-V 的なランレベル変更メカニズム
sysvinit-utils	V:459, I:999	79	System-V 的なユーティリティ (startpar(8)、bootlogd(8)、…)
lsb-base	V:890, I:999	49	Linux Standard Base 3.2 の init スクリプト機能
insserv	V:204, I:292	150	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
uswsusp	V:3, I:7	714	Linux が提供するユーザースペースソフトウェアによるサスペンドを使うためのツール
kexec-tools	V:1, I:9	288	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, I:1	128	ブートプロセスのパフォーマンスアナライザー
bootchart2	V:0, I:0	94	ブートプロセスのパフォーマンスアナライザー
pybootchartgui	V:0, I:0	177	ブートプロセスのアナライザー (可視化)
mingetty	V:0, I:3	38	コンソール専用 getty(8)
mgetty	V:0, I:1	315	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のリスト

- ***.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) を参照ください。
- ***.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) を参照ください。
- ***.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) を参照ください。
- ***.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) を参照ください。
- ***.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) を参照ください。
- ***.path** は systemd がパス基準で起動するために監視するパスを記述します。systemd.path(5) を参照ください。
- ***.socket** は systemd がソケット基準で起動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) を参照ください。
- ***.timer** は systemd がタイマー基準で起動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) を参照ください。
- ***.slice** は cgroups(7) でリソースを管理します。systemd.slice(5) を参照ください。
- ***.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) を参照ください。
- ***.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) を参照ください。

システムの始動 (init) されると systemd プロセスは (通常 "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 有効化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。

3.2.1 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。

3.2.2 ファイルシステム

通常のディスクやネットワークのファイルシス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) を参照ください。

mdadm(8) を用いるソフトウェア RAID は `/etc/mdadm/mdadm.conf` で設定されます。mdadm.conf(5) を参照ください。

**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。

3.2.3 ネットワークインターフェースの初期化

最近の systemd 下の 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"` で通常初期化されます。

どのように設定するのは第 5 章を参照下さい。

3.3 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、その閾値で設定できる。

```
# dmesg -n3
```


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	クリティ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	デバッ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 システムメッセージ

Under `systemd`, both kernel and system messages are logged by the journal service `systemd-journald.service` (a.k.a `journald`) either into a persistent binary data below `/var/log/journal` or into a volatile binary data below `/run/log/journal`. These binary log data are accessed by the `journalctl(1)` command. For example, you can display log from the last boot as:

```
$ journalctl -b
```

操作	コマンド断片
View log for system services and kernel from the last boot	<code>"journalctl -b --system"</code>
View log for services of the current user from the last boot	<code>"journalctl -b --user"</code>
View job log of <code>"\$unit"</code> from the last boot	<code>"journalctl -b -u \$unit"</code>
View job log of <code>"\$unit"</code> (<code>"tail -f"</code> style) from the last boot	<code>"journalctl -b -u \$unit -f"</code>

Table 3.5: List of typical `journalctl` command snippets

Under `systemd`, the system logging utility `rsyslogd(8)` may be uninstalled. If it is installed, it changes its behavior to read the volatile binary log data (instead of pre-`systemd` default `/dev/log`) and to create traditional permanent ASCII system log data. This can be customized by `/etc/default/rsyslog` and `/etc/rsyslog.conf` for both the log file and on-screen display. See `rsyslogd(8)` and `rsyslog.conf(5)`. See also [項9.3.2](#).

3.5 System management

The `systemd` offers not only init system but also generic system management operations with the `systemctl(1)` command.

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"`を通常前置する。

`"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される。

- 白い"`●`" は" 活動停止" や" 停止済み" の状態を示す。
- 赤い"`●`" は" 失敗発生" や" エラー発生" の状態を示す。
- 緑の"`●`" は" 活動中" や" 再起動中" や" 起動中" の状態を示す。

操作	コマンド断片
全ターゲットユニット設定をリスト	"systemctl list-units --type=target"
全サービスユニット設定をリスト	"systemctl list-units --type=service"
全ユニット設定タイプをリスト	"systemctl list-units --type=help"
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"
デフォルトのターゲット設定を取得	"systemctl get-default"
"graphical" にデフォ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再起動	"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再起動	"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: List of typical systemctl command snippets

3.6 Other system monitors

Here are a list of other monitoring command snippets under `systemd`. Please read the pertinent manpages including `cgroups(7)`.

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する	"systemd-analyze time"
初期化にかかった時間を全ての unit に関してリストする	"systemd-analyze blame"
読み込み"\$unit" ファイル中のエラーを検出する。	"systemd-analyze verify \$unit"
Show terse runtime status information of the user of the caller's session	"loginctl user-status"
Show terse runtime status information of the caller's session	"loginctl session-status"
<code>cgroups</code> を用いてブートプロセスを追跡する	"systemd-cgls"
<code>cgroups</code> を用いてブートプロセスを追跡する	"ps xawf -eo pid,user,cgroup,args"
<code>cgroups</code> を用いてブートプロセスを追跡する	"/sys/fs/cgroup/systemd/" の下の <code>sysfs</code> を読む

Table 3.7: List of other monitoring command snippets under `systemd`

3.7 systemd のカスタム化

3.7.1 ソケットの起動

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` の後に起動される。`"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"` をかえましょう。

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 `inetd` (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有効化していました。`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
```

3.8 udev システム

udev システムはハードウェア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3.8.1参照下さい) 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。

3.8.1 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3.8参照下さい) 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" ファイル中にリストしてプリロードする必要のある (modules(5) 参照下さい) 次に記すような非ハードウェアや特殊ハードウェアのドライバモジュールがあります。

- 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5.6) 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**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/bl" 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ェアを特定します。項[9.5.3](#)を参照下さい。

ブート時に期待されるハードウェア機能を有効とな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 を手元に置る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 通常の Unix 認証

通常の 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” ファイルは次の内容です。

```
...
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次の意味です。

- 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。
	このアカウントへのログイン不可
!	このアカウントへのログイン不可

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$IffyQwBe9b8tiyIkt2P4F/: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次の意味です。

- 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次の意味です。

- 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"authoptionalpam_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
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 <i>user_name</i>	" <i>user_name</i> " のアカウント情報の閲覧
getent shadow <i>user_name</i>	" <i>user_name</i> " のシャドーされたアカウント情報の閲覧
getent group <i>group_name</i>	" <i>group_name</i> " 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	アカウント開設のための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:32, I:371	364	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:814, I:994	1465	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 システムのような最新の [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次のようになります。

- PAM は、アプリケーションソフトウェアが使う柔軟な認証メカニズムを提供し、パスワードデータの交換に関与します。
- NSS は、`ls(1)` and `id(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。

- `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` 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)` を参照ください。

パッケージ	ポップコン	サイズ	説明
libpam-modules	V:813, I:999	1048	差し替え可能な認証モジュール (基本サービス)
libpam-ldap	I:12	249	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-cracklib	I:15	117	cracklib のサポートを可能にする差し替え可能な認証モジュール
libpam-systemd	V:484, I:869	574	logind のために登録ユーザーセッションを登録するプラグブルオーセンティケーション (PAM)
libpam-doc	I:1	1046	差し替え可能な認証モジュール (html と text の文書)
libc6	V:946, I:999	12772	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:11	3161	GNU C ライブラリー: マンページ
glibc-doc-reference	I:4	12740	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	I:508	150	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-ldap	I:11	265	LDAP をネームサービスとして使う NSS モジュール
libnss-ldapd	I:14	153	LDAP をネームサービスとして使う NSS モジュール (libnss-ldap の新たなフォーク)

Table 4.5: 特記すべき PAM と NSS システムのリスト

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。

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 最新の集中システム管理

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 的システムを最新の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](#) です。

LDAP サーバーは、`libpam-ldap` と `libnss-ldap` 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、次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)` 等のプログラムを走らせることで集中化された LDAP サーバーを設置します。
- デフォルトの "`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。
 - Debian では、"`/etc/pam_ldap.conf`" を `libpam-ldap` の設定ファイル、"`/etc/pam_ldap.secret`" を root のパスワードを保存するファイルとして使っています。
- デフォルト ("`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。

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_limits(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

- Debian では、"/etc/libnss-ldap.conf" を libnss-ldap の設定ファイルとして使っています。
- パスワードのセキュリティー確保のために libpam-ldap 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-ldap 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。

libpam-doc パッケージで提供される pam_ldap.conf(5) や "/usr/share/doc/libpam-doc/html/" や glibc-doc パッケージで提供される "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\(YP\)/NIS/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 を使っているのだから "/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-cracklib 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。

On a typical GNOME system which automatically installs libpam-gnome-keyring, "/etc/pam.d/common-password" looks like:

```
# here are the per-package modules (the "Primary" block)
password requisite pam_cracklib.so retry=3 minlen=8 difok=3
password [success=1 default=ignore] pam_unix.so obscure use_authok 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

注意

ここに書かれている情報はあなたのセキュリティ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なパスワード

多くのトランスポート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はない考えです。これらに関しては、"[トランスポートレイヤーセキュリティ](#)"(TLS) もしくはその前身の"セキュアソケットレイヤー"(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にしてサービスができます。

インセキュアなサービス名	ポート	セキュアなサービス名	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: インセキュアとセキュア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ができます。(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。)

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、公開鍵認証、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うとします。公開鍵認証を利用すると、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。次のようなセキュリティ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5.6](#)参照下さい) を使うのはセキュリティ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, I:3	102	小さなポートノックのデーモン <code>knockd(1)</code> とクライアント <code>konck</code>
fail2ban	V:116, I:129	2092	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
libpam-shield	V:0, I:0	115	パスワード推測によるリモートからの攻撃者を締め出す

Table 4.8: 追加セキュリティ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確保

あなたの機器に他人が root 権限を持ってアクセスするのを阻止するには、次のアクションが必要です。

- ハードディスクへの物理的アクセスを阻止
- UEFI/BIOS をロックして、リムーバブルメディアからの起動を阻止
- 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、パスワードをリセットすることは次の手順を使うと比較的簡単です。

1. ハードディスクを CD から起動可能な UEFI/BIOS のついた PC に移動します。
2. レスキューメディア (Debian ブートディスク、Knoppix CD、GRUB CD、...) でシステムを起動します。
3. ルートパーティションを読み出し / 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` の起動時に GRUB のメニュー項目を編集可能 ([項3.1.2](#)参照下さい) なら、次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/hda6 rw init=/bin/sh"` のような感じに変更してシステムを起動します。

2. `/etc/passwd` を編集し、`root` アカountの2番目の項目を空にします。
3. システム再起動します。

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9.5.11参照下さい) 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。

このような懸念を回避できる唯一の合理的なソフトウェア的解決法は、[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起動毎に必要なになってしまいます。

4.7 他のアクセスコントロール

There are access controls to the system other than the password based authentication and file permissions.

注意

カーネルの[セキユアーアテンションキー \(SAK\)](#) 機能の制限は項9.4.15を参照下さい。

4.7.1 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` はユーザーの通常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/examples` 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ount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。

4.7.2 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。

4.7.3 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有効に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ウォール

項3.5 と項4.5.1と項5.6 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リスクに注意を払わなければいけません。)

4.7.4 Linux のセキュリティ機能

Linux kernel has evolved and supports security features not found in traditional UNIX implementations.

Linux supports [extended attributes](#) which extend the traditional UNIX attributes (see [xattr\(7\)](#)).

Linux divides the privileges traditionally associated with superuser into distinct units, known as [capabilities\(7\)](#), which can be independently enabled and disabled. Capabilities are a per-thread attribute since kernel version 2.2.

The [Linux Security Module \(LSM\) framework](#) provides a [mechanism for various security checks](#) to be hooked by new kernel extensions. For example:

- AppArmor
 - Security-Enhanced Linux (SELinux)
 - Smack (Simplified Mandatory Access Control Kernel)
 - Tomoyo Linux
-

Since these extensions may tighten privilege model tighter than the ordinary Unix-like security model policies, even the root power may be restricted. You are advised to read the [Linux Security Module \(LSM\) framework document at kernel.org](#).

Linux [namespaces](#) wrap a global system resource in an abstraction that makes it appear to the processes within the namespace that they have their own isolated instance of the global resource. Changes to the global resource are visible to other processes that are members of the namespace, but are invisible to other processes. Since kernel version 5.6, there are 8 kinds of namespaces (see [namespaces\(7\)](#), [unshare\(1\)](#), [nsenter\(1\)](#)).

As of Debian 11 Bullseye (2021), Debian uses unified cgroup hierarchy (a.k.a. [cgroups-v2](#)).

Usage examples of [namespaces](#) with [cgroups](#) to isolate their processes and to allow resource control are:

- [Systemd](#). See [項3.2](#).
- [サンドボックス環境](#)。 [項7.6](#) を参照。
- [Linux containers](#) such as [Docker](#), [LXC](#). See [項9.11](#).

These functionalities can't be realized by [項4.1](#). These advanced topics are mostly out-of-scope for this introductory document.

Chapter 5

ネットワークの設定

ティップ

最近の 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)` を参照ください。

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 メソッドが最初に発動されます。ホスト名が `/etc/hosts` ファイルに見つかり、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。

例えば、`/etc/hosts` は以下の内容です。

```
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	ポップコン	サイズ	タイプ	説明
network-manager	V:360, I:426	15210	設定::NM	NetworkManager (デーモン): ネットワークを自動管理
network-manager-gnome	V:124, I:357	5351	設定::NM	NetworkManager (GNOME フロントエンド)
ifupdown	V:612, I:989	217	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)
isc-dhcp-client	V:222, I:980	686	設定:: 低レベル	DHCP クライアント
pppoeconf	V:0, I:8	192	設定:: ヘルパー	PPPoE コネクションの設定ヘルパー
wpasupplicant	V:335, I:491	3439	, ,	WPA と WPA2 (IEEE 802.11i) のためのクライアントサポート
wpaui	V:0, I:2	780	, ,	wpa_supplicant の Qt GUI クライアント
wireless-tools	V:180, I:239	297	, ,	Linux のワイヤレス拡張を操作するツール
iw	V:320, I:467	293	, ,	Linux のワイヤレスデバイスを設定するツール
iproute2	V:695, I:937	2867	設定::iproute2	iproute2 、IPv6 や他の上級ネットワーク設定: ip(8) や tc(8) 等
iptables	V:314, I:990	2521	設定::Netfilter	パケットフィルタと NAT のための管理ツール (Netfilter)
iputils-ping	V:224, I:997	113	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 (旧来、GNU)
iputils-arping	V:7, I:98	55	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:4, I:53	72	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:266	597	テスト	Ethernet デバイス設定の表示と変更
mtr-tiny	V:6, I:55	161	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:46	214	, ,	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:1, I:38	2105	, ,	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:246	4509	, ,	ネットワークマッパー / ポートスキャナー (Nmap 、コンソール)
zenmap	V:1, I:8	2939	, ,	ネットワークマッパー / ポートスキャナー (GTK)
tcpdump	V:19, I:208	1329	, ,	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	I:54	65	, ,	ネットワークトラフィックアナライザー (Wireshark 、GTK)
tshark	V:3, I:33	408	, ,	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, I:3	401	, ,	tcpdump の出力から接続状況のまとめを作成
snort	V:0, I:1	2206	, ,	柔軟なネットワーク侵入検知システム (Snort)
ntopng	V:1, I:2	969	, ,	ネットワークの使用状況をウェブブラウザで表示
dnsutils	V:59, I:490	261	, ,	BIND によって提供されるネットワーククライアント: nslookup(8) と nsupdate(8) と dig(8)
dlint	V:0, I:6	53	, ,	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, I:1	61	, ,	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が、[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 (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。

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、それは次の内容です。

```
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, [Zeroconf](#)) 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 mdns4` のようなスタンプがあるべきです。
- ホスト名が `".local"` で終わる [擬似-top-level domain](#) (TLD) が解決されます。
- mDNS の IPv4 リンク-ローカルのマルチキャストアドレス "224.0.0.251" とか IPv6 でそれに相当する `"FF02::FB"` が `".local"` で終わる名前の DNS クエリーに用いられます。

非推奨である [NETBios over TCP/IP](#) を使うホスト名解決は、`winbind` パッケージをインストールすると提供できます。このような機能を有効にするには、`/etc/nsswitch.conf` ファイル中に `hosts: files mdns4_minimal [NOTFOUND=return] dns mdns4 wins` のようなスタンプが必要です。(最近の Windows システムは通常 `dns` メソッドをホスト名の解決に使います。)

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。

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。

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	of subnets
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
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。

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ェア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](#) のファームウェアが必要です。項[9.10.5](#)を参照下さい。

5.2 デスクトップのためのモダンネットワーク設定

最近の systemd 下の 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”で通常初期化されます。

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。

- それらには洒落た [GUI](#) やコマンドラインのユーザーインターフェースとともに提供されます。
- それらのバックエンドシステムとして、自前の[デーモン](#)とともに提供されます。
- 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バイ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. 次のようにして、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一つの方法です。)

```
$ sudo adduser foo netdev
```

2. `/etc/network/interfaces` の設定を次のようにできるだけ簡単にします。

```
auto lo
iface lo inet loopback
```

3. 次のようにして NM を再起動します。

```
$ sudo systemctl restart network-manager
```

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` にリストされてい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`、`gnome-bluetooth` 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 ください。

5.3 GUI 無しのモダンネットワーク設定

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)` を参照ください。

これにより GUI 無しのモダン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 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。

5.4.1 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 [IPROUTE2 Utility Suite Howto](#) を参照下さい。

5.4.2 安全な低レベルネットワーク操作

次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	有効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
tracpath 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
netstat -a	オープンポートの発見
netstat -l --inet	聴取中のポートの発見
netstat -ln --tcp	聴取中の TCP ポートの発見 (数字)
dlint example.com	"example.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"/sbin/" 中にあります。"/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"/sbin" を追加する必要があるかもしれません。

5.5 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	ポップコン	サイズ	説明
iftop	V:8, I:118	97	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:4, I:55	305	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, I:8	60	インターフェース統計モニター
bmon	V:1, I:18	146	可搬型バンド幅モニター兼速度推定機
ethstatus	V:0, I:4	40	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
bing	V:0, I:1	80	経験則的確率バンド幅試験ソフト
bwm-ng	V:2, I:18	90	簡単軽量のコンソール式のバンド幅モニター
ethstats	V:0, I:0	23	コンソール式のイーサネット統計モニター
ipfm	V:0, I:0	78	帯域分析ツール

Table 5.5: ネットワーク最適化ツールのリスト

5.5.1 最適 MTU の発見

[最大送信単位 \(MTU\)](#) 値は、ping(8) を"-M do" オプションとともに使って ICMP パケットをデーターサイズ 1500 (IP+ICMP ヘッダー分の 28 バイトを加えて) から始めて IP フラグメンテーションしない最大サイズを見つけることで実験的に決定できます。

例えば、次を試してみてください:

```
$ ping -c 1 -s $((1500-28)) -M do www.debian.org
PING www.debian.org (194.109.137.218) 1472(1500) bytes of data.
From 192.168.11.2 icmp_seq=1 Frag needed and DF set (mtu = 1454)

--- www.debian.org ping statistics ---
0 packets transmitted, 0 received, +1 errors
```

MTU=1500 ではなく MTU=1452 を試す

MTU=1454 で ping(8) が成功するのを確認します。

MTU が 1500 でない場合は NM で MTU を設定したほうがよいでしょう。

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。

ティップ
PMTU 値が 1454 となる上記例は [Asynchronous Transfer Mode](#) (ATM) をバックボーンネットワークとして使い顧客を [PPPoE](#) でサービスしていた FTTP プロバイダーの場合でした。実際の PMTU 値はあなたの環境に依存します。例えば私の新しい FTTP プロバイダーの場合は 1500 です。

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。

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
イーサネット接続 (IP: DHCP または固定)	1500	標準かつデフォルト
イーサネット接続 (IP: PPPoE)	1492 (=1500-8)	PPP ヘッダーに 2 バイト、PPPoE ヘッダーに 6 バイト、
イーサネット接続 (ISP のバックボーン: ATM、IP: DHCP または固定)	1462 (=48*31-18-8)	著者推定: イーサネットヘッダーに 18、SAR 末尾に 8
イーサネット接続 (ISP のバックボーン: ATM、IP: PPPoE)	1454 (=48*31-8-18-8)	author's speculation: 18 bytes for Ethernet header, 8 bytes for SAR trailer, 2 bytes for PPP header and 6 bytes for PPPoE header

Table 5.6: 最適 MTU 値の基本的なガイドライン

- 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5.6参照下さい) 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効です。iptables(8) 中の "TCP MSS" を参照下さい。

5.5.2 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 バッファサイズパラメーターを "TCP Tuning Guide" や "TCP tuning" に書かれてい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。

5.6 Netfilter インフラ

Netfilter はLinux カーネルのモジュール (項3.8.1参照下さい) を利用するステートフルファイア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。

netfilter のユーザー空間の主プログラムは iptables(8) です。シェルから対話形式で手動で netfilter を設定し、その状態を iptables-save(8) で保存し、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。

<http://www.netfilter.org/documentation/> (または "/usr/share/doc/iptables/html/" 中) の文書を参照下さい。

- [Linux Networking-concepts HOWTO](#)
- [Linux 2.4 Packet Filtering HOWTO](#)

パッケージ	ポプコン	サイ ズ	説明
iptables	V:314, I:990	2521	netfilter の管理ツール (IPv4 用の iptables(8) 、IPv6 用の ip6tables(8))
arptables	V:0, I:2	96	netfilter の管理ツール (ARP 用の arptables(8))
ebtables	V:16, I:37	265	netfilter の管理ツール (Ethernet ブリッジング用の ebtables(8))
iptstate	V:0, I:3	116	netfilter の状態を常時モニター (top(1) と類似)
shorewall-init	V:0, I:0	68	Shoreline ファイアーウォール初期化
shorewall	V:4, I:12	2458	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, I:0	65	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:1, I:2	779	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, I:0	64	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 NAT HOWTO](#)

ティップ

これらは Linux **2.4** のために書かれたとはいえ、[iptables\(8\)](#) コマンドも [netfilter](#) カーネル機能も現在の Linux **2.6** や **3.x** カーネルシリーズにもあてはまります。

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

最近の 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](#) を参照ください。

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 ウェブブラウザ

多くの[ウェブブラウザ](#)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ツにアクセスするために存在します。

6.1.1 ブラウザー設定

次に示す特別の URL 文字列を使うと一部のブラウザでその設定値を確認する事ができます。

- `”about:”`
- `”about:config”`
- `”about:plugins”`

Debian は、[Java \(ソフトウェアプラットフォーム\)](#) や [Flash](#) のみならず、[MPEG](#) や [MPEG2](#) や [MPEG4](#) や [DivX](#) や [Windows Media Video \(.wmv\)](#) や [QuickTime \(.mov\)](#) や [MP3 \(.mp3\)](#) や [Ogg/Vorbis](#) ファイルや DVDs や VCDs 等を取り扱うブラウザのプラグインコンポーネントを提供します。Debian では contrib や non-free アーカイブエリアに non-free のブラウザプラグインパッケージを提供しています。

ティップ

上記の Debian パッケージを使うのが遥に簡単であるとはいえ、今でもブラウザのプラグインは`”*.so”` をプラグインディレクトリー (例えば`”/usr/lib/iceweasel/plugins/”`) 等にインストールしブラウザを再起動することで手動で有効にすることができます。

パッケージ	ポプコン	サイ ズ	タイプ	ウェブブラウザの説明
chromium	V:54, I:134	192549	X	Chromium 、(Google からのオープンソースブラウザー)
firefox	V:13, I:20	209827	,,	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:199, I:421	200011	,,	Firefox ESR , (Firefox 延長サポートリリース)
epiphany-browser	V:4, I:23	3730	,,	GNOME 、 HIG 準拠 、 Epiphany
konqueror	V:17, I:95	20763	,,	KDE 、 Konqueror
dillo	V:1, I:7	1536	,,	Dillo , (軽量ブラウザー, FLTK 準拠)
w3m	V:25, I:252	2367	テキスト	w3m
lynx	V:11, I:97	1948	,,	Lynx
elinks	V:5, I:27	1767	,,	ELinks
links	V:5, I:38	2250	,,	Links (テキストのみ)
links2	V:1, I:14	5418	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: ウェブブラウザのリスト


パッケージ	ポプコン	サイ ズ	エリア	説明
pepperflashplugin-nonfree	V:0, I:15	26	contrib	Pepper FlashPlayer - ブラウザプラグイン

Table 6.2: ブラウザプラグインのリスト

ウェブサイトによっては使っているブラウザのユーザーエージェント文字列によって接続を拒否します。こういう状況はユーザーエージェント文字列を偽装することで回避できます。例えば、これは次の内容を”~/.gnome2/epiphany/mozilla/epiphany/user.js”か”~/.mozilla/firefox/*.default/user.js”といったユーザー設定ファイル追加すればできます。

```
user_pref{"general.useragent.override","Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0)"};
```

こうする代わりに、URL に”about:config”を入力して表示画面内容を右クリックしてこの変数を追加や再設定することでもできます。




注意

偽装されたユーザーエージェント文字列は [Java](#) に対して良からぬ副次効果を引き起こすかもしれません。

6.2 メールシステム

This section focuses on typical mobile workstations on consumer grade Internet connections.



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。(通常これらはメッセージエンベロープの情報と共通ですが、必ずしもそうとは限りません。)
- The email message format covering header and body data is extended by [Multipurpose Internet Mail Extensions \(MIME\)](#) from the plain ASCII text to other character encodings, as well as attachments of audio, video, images, and application programs.

Full featured GUI based [email clients](#) offer all the following functions using the GUI based intuitive configuration.

- It creates and interprets the message header and body data using [Multipurpose Internet Mail Extensions \(MIME\)](#) to deal the content data type and encoding.
- It authenticates itself to the ISP’s SMTP and IMAP servers using the legacy [basic access authentication](#) or modern [OAuth 2.0](#). (For [OAuth 2.0](#), set it via Desktop environment settings. E.g., ”Settings” -> ”Online Accounts”).
- It sends the message to the ISP’s smarthost SMTP server listening to the message submission port (587).
- It receives the stored message on the ISP’s server from the TLS/IMAP4 port (993).
- 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	ポップコン	サイズ	タイプ
evolution	V:29, I:219	475	X GUI プログラム (GNOME3、グループウェアスイート)
thunderbird	V:57, I:134	168690	X GUI プログラム (GTK、 Mozilla Thunderbird)
kmail	V:33, I:85	18011	X GUI プログラム (KDE)
mutt	V:33, I:278	7071	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, I:0	2334	(x)emacs の下でキャラクターターミナルプログラム

Table 6.3: メールユーザーエージェント (MUA) のリスト

6.2.2 現代的メールサービスの限界

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限があります。

- 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。
- A mail may be rejected by any host en route to the destination quietly unless it appears as authentic as possible.
- 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:

- The SMTP port (25) connections from hosts serviced by the consumer grade network to the Internet are blocked.
- The SMTP port (25) connections to hosts serviced by the consumer grade network from the Internet are blocked.
- The outgoing messages from hosts serviced by the consumer grade network to the Internet can only be sent via the message submission port (587).
- [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 Historic mail service expectation

Some programs on Debian expect to access the `/usr/sbin/sendmail` command to send emails as their default or customized setting since the mail service on a UNIX system functioned historically as:

- An email is created as a text file.
- The email is handed to the `/usr/sbin/sendmail` command.
- For the destination address on the same host, the `/usr/sbin/sendmail` command makes local delivery of the email by appending it to the `/var/mail/$username` file.
 - Commands expecting this feature: `apt-listchanges`, `cron`, `at`, ...
- For the destination address on the remote host, the `/usr/sbin/sendmail` command makes remote transfer of the email to the destination host found by the DNS MX record using SMTP.
 - Commands expecting this feature: `popcon`, `reportbug`, `bts`, ...

6.2.4 メール転送エージェント (MTA)

In principle, mobile workstations should function without the `/usr/sbin/sendmail` command provided by the [mail transfer agent \(MTA\)](#) program.

The Debian system usually installs MTA to cope with [項6.2.2](#) and [項6.2.3](#) even if mobile workstations installed full featured GUI based [email clients](#).

For mobile workstations, the typical choice of MTA is either `exim4-daemon-light` or `postfix` with its installation option such as "Mail sent by smarthost; received via SMTP or fetchmail" selected. These are light weight MTAs that respect `/etc/aliases`.

ティップ

Configuring `exim4` to send the Internet mail via multiple corresponding smarthosts for multiple source email addresses is non-trivial. If you need such capability for some programs, set them up to use `msmtp` which is easy to set up for multiple source email addresses. Then leave main MTA only for a single email address.

パッケージ	ポップコン	サイズ	説明
exim4-daemon-light	V:320, I:346	1517	Exim4 メール転送エージェント (MTA: Debian のデフォルト)
exim4-daemon-heavy	V:7, I:8	1668	Exim4 メール転送エージェント (MTA: 柔軟な代替候補)
exim4-base	V:328, I:355	1720	Exim4 文書 (text) と共通ファイル
exim4-doc-html	I:1	3667	Exim4 文書 (html)
exim4-doc-info	I:1	625	Exim4 文書 (info)
postfix	V:147, I:162	4183	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:9	4444	Postfix 文書 (html+text)
saslm2-bin	V:5, I:19	428	Cyrus SASL API の実装 (SMTP AUTH について postfix を補完)
cyrus-saslm2-doc	I:1	575	Cyrus SASL - 文書
msmtp	V:6, I:12	547	軽量 MTA
msmtp-mta	V:4, I:5	86	軽量 MTA (msmtp の sendmail 互換性拡張)
esmtp	V:0, I:0	128	軽量 MTA
esmtp-run	V:0, I:0	32	軽量 MTA (esmtp の sendmail 互換性拡張)
nullmailer	V:8, I:10	479	Strip down MTA, no local mail
ssmtp	V:6, I:10	2	Strip down MTA, no local mail
sendmail-bin	V:14, I:15	1859	高機能 MTA (既に慣れている場合)
courier-mta	V:0, I:0	2442	超高機能 MTA (ウェブインターフェースなど)

Table 6.4: 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

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-*` パッケージを次のように (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。

”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-addresses”を使用します。)

”DNS クエリを最小限に留めますか (ダイヤルオンデマンド)?” に次の内のひとつの返答をします。

- 起動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。
- 起動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を作成します。

```
$ sudo vim /etc/exim4/passwd.client
...
$ cat /etc/exim4/passwd.client
^smtp.*\.hostname\.dom:username@hostname.dom:password
```

Configure `exim4(8)` with `"QUEUERUNNER='queueonly'"`, `"QUEUERUNNER='nodaemon'"`, etc. in `"/etc/default/exim4"` to minimize system resource usages. (optional)

次のようにして `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。真のホスト名は次の様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。

次のようにすれば `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.template"`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`



警告

For all practical consideration, use **SMTP** with **STARTTLS** on port 587 or **SMTPS** SSL (SMTPS) on port 465, instead of plain SMTP on port 25.

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。

`postfix` と `sasl2-bin` パッケージを次のように (再) 設定します。

```
$ sudo systemctl stop postfix
$ sudo dpkg-reconfigure postfix
```

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 エイリアステーダーベースのメンテナンス

Table 6.5: 重要 postfix マニュアルページのリスト

”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 hash:/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_RE”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) コマンド実行後有効。
/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) コマンド実行後有効。

Table 6.6: 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" をエクスポートしてその値をオーバーライドします。

ティップ

通常 popularity-contest パッケージは root アカウトからメールを 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次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 は exiqgrep(8) や exipick(8) のようないくつかのユーティリティプログラムとともに供給されます。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。

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)

セキュアシェル (SSH) はインターネット経由で接続するセキュアな方法です。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。

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 6.7: 基本的 MTA 操作のリスト

パッケージ	ポプコン	サイズ	ツール	説明
openssh-client	V:809, I:997	4298	ssh(1)	セキュアシェルクライアント
openssh-server	V:701, I:844	1567	sshd(8)	セキュアシェルサーバー
ssh-askpass	V:2, I:32	106	ssh-askpass(1)	ユーザーに ssh-add 用のパスフレーズを尋ねる (ブレーン X)
ssh-askpass-gnome	V:0, I:5	294	ssh-askpass-gnome(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, I:0	42	ssh-askpass-fullscreen(1)	asks user for a pass phrase for ssh-add (GNOME) with extra eye candy
shellinabox	V:0, I:2	507	shellinaboxd(1)	web server for browser accessible VT100 terminal emulator

Table 6.8: リモートアクセスサーバーとユーティリティのリスト

ssh(1) はユーザーにとってより賢明でよりセキュアな 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 (初期デフォルト CTRL-]) に会うことで中断される事はありません。

Although shellinabox is not a SSH program, it is listed here as an interesting alternative for the remote terminal access.
See also 項7.8 for connecting to remote X client programs.



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。
Please read "/usr/share/doc/openssh-client/README.Debian.gz", ssh(1), sshd(8), ssh-agent(1), and ssh-keygen(1).
ssh-add(1) and ssh-agent(1).



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。
Don't enable rhost based authentication (HostbasedAuthentication in /etc/ssh/sshd_config).

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ルト、ssh_config(5) 参照下さい
/etc/ssh/sshd_config	SSH サーバーのデフォルト、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 鍵 (ecdsa, ed25519 など)

Table 6.9: SSH 設定ファイルのリスト

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。

```
Host salsa.debian.org people.debian.org
User foo-guest
```

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 6.10: SSH クライアント起動例のリスト

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、ローカルで認証鍵を生成しリモートシステム上に公開鍵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。

環境	フリーの SSH プログラム
Windows	puTTY (http://www.chiark.greenend.org.uk/~sgtatham/putty/) (GPL)
Windows (cygwin)	cygwin 中の SSH (http://www.cygwin.com/) (GPL)
古典的 Macintosh	macSSH (http://www.macssh.com/) (GPL)
Mac OS X	OpenSSH; ターミナルアプリケーションの ssh を使用しましょう (GPL)

Table 6.11: 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

6.3.5 ssh-agent の設定

SSH の認証鍵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。

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 鍵 (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)
```

次に示すように、今後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、通常の 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次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9.4.13参照下さい) を使って次のようにして 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しょう; 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

In the old Unix-like system, the BSD [Line printer daemon \(lpr\)](#) was the standard and the standard print out format of the classic free software was [PostScript \(PS\)](#). Some filter system was used along with [Ghostscript](#) to enable printing to the non-PostScript printer. See [項11.4.1](#).

In the modern Debian system, the [Common UNIX Printing System \(CUPS\)](#) is the de facto standard and the standard print out format of the modern free software is [Portable Document Format \(PDF\)](#).

The CUPS uses [Internet Printing Protocol \(IPP\)](#). The IPP is now supported by other OSs such as Windows XP and Mac OS X and has become new cross-platform de facto standard for remote printing with bi-directional communication capability.

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:3, I:4	362	printer (515)	BSD <code>lpr/lpd</code> (ラインプリンターデーモン)
lprng	V:0, I:1	3064	, ,	, , (拡張)
cups	V:129, I:376	1134	IPP (631)	インターネット印刷 CUPS サーバー
cups-client	V:139, I:434	493	, ,	CUPS 用 System V プリンターコマンド : <code>lp(1)</code> と <code>lpstat(1)</code> と <code>lpoptions(1)</code> と <code>cancel(1)</code> と <code>lpmove(8)</code> と <code>lpinfo(8)</code> と <code>lpadmin(8)</code> 等
cups-bsd	V:35, I:361	122	, ,	CUPS 用 BSD プリンターコマンド : <code>lpr(1)</code> と <code>lpq(1)</code> と <code>lprm(1)</code> と <code>lpc(8)</code> 等
printer-driver-gutenprint	V:55, I:343	941	非該当	CUPS 用のプリンタードライバ

Table 6.12: プリントサーバーとユーティリティのリスト

ティップ

CUPS システムはウェブブラウザを”<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。

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 `isc-dhcp-server` 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。

パッケージ	ポート	サイズ	プロトコル	説明
telnetd	V:1, I:2	111	TELNET	TELNET サーバー
telnetd-ssl	V:0, I:0	169	,,	,, (SSL サポート)
nfs-kernel-server	V:42, I:78	342	NFS	Unix 式ファイル共有
samba	V:105, I:155	16630	SMB	Windows のファイルとプリンター共有
netatalk	V:2, I:3	2077	ATP	Apple/Mac のファイルとプリンター共有 (AppleTalk)
proftpd-basic	V:23, I:31	488	FTP	汎用ファイルダウンロード
apache2	V:244, I:310	614	HTTP	汎用ウェブサーバー
squid	V:12, I:14	8381	,,	汎用ウェブ プロキシサーバー
squid3	V:3, I:9	240	,,	,,
bind9	V:52, I:63	1070	DNS	他のホストの IP アドレス
isc-dhcp-server	V:18, I:52	1471	DHCP	クライアント自身の IP アドレス

Table 6.13: 他のネットワークアプリケーションサーバー

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、次のようにします。

```
$ 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。

```
$ telnet -z ssl pop.gmail.com 995
```

```
$ openssl s_client -connect pop.gmail.com:995
```

次の [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。
”/etc/services” の中にポートの使用され方が記載されています。

パッケージ	ポプコン	サイズ	プロトコル	説明
netcat	I:43	16	TCP/IP	TCP/IP 用万能ツール (スイス陸軍ナイフ)
openssl	V:814, I:994	1465	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:6, I:16	520	,,	万能 SSL ラッパー
telnet	V:61, I:901	163	TELNET	TELNET クライアント
telnet-ssl	V:0, I:3	209	,,	,, (SSL サポート)
nfs-common	V:182, I:323	768	NFS	Unix 式ファイル共有
smbclient	V:16, I:174	2017	SMB	MS Windows のファイルとプリンター共有
cifs-utils	V:29, I:123	307	,,	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
ftp	V:16, I:251	137	FTP	FTP クライアント
lftp	V:6, I:40	2255	,,	,,
ncftp	V:3, I:21	1339	,,	フルスクリーンの FTP クライアント
wget	V:255, I:987	3477	HTTP と FTP	ウェブダウンローダー
curl	V:160, I:575	427	,,	,,
axel	V:0, I:4	216	,,	加速ダウンローダー
aria2	V:2, I:19	1854	,,	BitTorrent と Metalink サポート付き、加速ダウンローダー
bind9-host	V:176, I:946	370	DNS	bind9 由来の host(1) コマンド、"Priority: standard"
dnsutils	V:59, I:490	261	,,	bind 由来の dig(1) コマンド、"Priority: standard"
isc-dhcp-client	V:222, I:980	686	DHCP	IP アドレスの獲得
ldap-utils	V:12, I:74	719	LDAP	LDAP サーバーからデータ獲得

Table 6.14: 他のネットワークアプリケーションクライアント

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 6.15: よく使われる RFC のリスト

Chapter 7

GUI システム

7.1 GUI desktop environment

There are several choices for the full featured [GUI](#) desktop environment on the Debian system.

タスクパッケージ	ポプコン	サイ ズ	説明
task-gnome-desktop	I:166	9	GNOME デスクトップ環境
task-xfce-desktop	I:102	9	Xfce デスクトップ環境
task-kde-desktop	I:65	6	KDE Plasma デスクトップ環境
task-mate-desktop	I:47	9	MATE デスクトップ環境
task-cinnamon-desktop	I:36	9	Cinnamon デスクトップ環境
task-lxde-desktop	I:33	9	LXDE デスクトップ環境
task-lxqt-desktop	I:13	9	LXQt デスクトップ環境
task-gnome-flashback-desktop	I:0	6	GNOME Flashback デスクトップ環境

Table 7.1: List of desktop environment

ティップ

Dependency packages selected by a task metapackage may be out of sync with the latest package transition state under the Debian unstable/testing environment. For task-gnome-desktop, you may need to adjust package selections as follows:

- Start aptitude(8) as `sudo aptitude -u`.
 - カーサを"Tasks" に移動し"Enter" を押す。
 - カーサを"End-user" に移動し"Enter" を押す。
 - カーサを"GNOME" に移動し"Enter" を押す。
 - カーサを task-gnome-desktop に移動し"Enter" を押す。
 - カーサを"Depends" に移動し"m" (手動で manually 選択) を押す。
 - カーサを"Recommends" に移動し"m" (手動で manually 選択) を押す。
 - カーサを"task-gnome-desktop に移動し"- を押す。(drop)
 - Adjust selected packages while dropping problematic ones causing package conflicts.
 - "g" を押してインストールを開始する。
-

This chapter will focus mostly on the default desktop environment of Debian: task-gnome-desktop offering [GNOME](#) on [wayland](#).

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 System コアプロトコル](#) (xwayland 経由)

Please check [freedesktop.org site for how Wayland architecture is different from X Window architecture](#).

From user's perspective, differences can be colloquially summarized as:

- Wayland is a same-host GUI communication protocol: new, simpler, faster, no setuid root binary
- X Window is a network-capable GUI communication protocol: traditional, complex, slower, setuid root binary

For applications using Wayland protocol, the access to their display contents from a remote host is supported by the [VNC](#) or [RDP](#). See [項7.7](#)

Modern X servers have [the MIT Shared Memory Extension](#) and communicate with their local X clients using the local shared memory. This bypasses the network transparent [Xlib](#) interprocess communication channel and gains performance. This situation was the [background](#) of creating Wayland as a local-only GUI communication protocol.

Using the `xeyes` program started from the GNOME terminal, you can check GUI communication protocol used by each GUI application.

```
$ xeyes
```

- If the mouse cursor is on an application such as "GNOME terminal" which uses Wayland display server protocol, eyes don't move with the mouse cursor.
-

- If the mouse cursor is on an application such as "xterm" which uses X Window System core protocol, eyes move with the mouse cursor exposing not-so-isolated nature of X Window architecture.

As of April 2021, many popular GUI applications such as GNOME and [LibreOffice \(LO\)](#) applications have been migrated to the Wayland display server protocol. I see xterm, gitk, chromium, firefox, gimp, dia, and KDE applications still use X Window System core protocol.

注意

For both the xwayland on Wayland or the native X Window System, the old X server configuration file `/etc/X11/xorg.conf` shouldn't exist on the system. The graphics and input devices are now configured by the kernel with [DRM](#), [KMS](#), and [udev](#). The native X server has been rewritten to use them. See "[modeb default video mode support](#)" in the Linux kernel documentation.

7.3 GUI インフラストラクチャー

Here are notable GUI infrastructure packages for the GNOME on Wayland environment.

パッケージ	ポプコン	パッケージサイズ	説明
mutter	V:7, I:204	208	GNOME の mutter ウィンドウマネージャ [自動]
xwayland	V:137, I:227	4572	An X server running on top of wayland [auto]
gnome-remote-desktop	V:14, I:23	207	Remote desktop daemon for GNOME using PipeWire [auto]
gnome-tweaks	V:14, I:169	1264	Advanced configuration settings for GNOME

Table 7.2: List of notable GUI infrastructure packages

Here, "**[auto]**" means that these packages are automatically installed when `task-gnome-desktop` is installed.

ティップ

`gnome-tweaks` is the indispensable configuration utility. For example:

- You can force "Over-Amplification" of sound volume from "General".
- You can force "Caps" to become "Esc" from "Keyboard & Mouse" -> "Keyboard" -> "Additional Layout Option".

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。

パッケージ	ポプコン	パッケージサイズ	タイプ	説明
evolution	V:29, I:219	475	GNOME	個人情報管理 (グループウェアと電子メール)
thunderbird	V:57, I:134	168690	GTK	E メールクライアント (Mozilla Thunderbird)
kontact	V:1, I:14	2152	KDE	個人情報管理 (グループウェアと電子メール)
libreoffice-writer	V:167, I:423	39325	LO	ワードプロセッサ
abiword	V:1, I:11	5133	GNOME	ワードプロセッサ
calligrawords	V:0, I:7	5717	KDE	ワードプロセッサ
scribus	V:1, I:22	30370	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, I:4	1326	GNOME	ラベルエディター
libreoffice-calc	V:166, I:418	32969	LO	スプレッドシート
gnumeric	V:5, I:20	9933	GNOME	スプレッドシート
calligrasheets	V:0, I:5	10890	KDE	スプレッドシート
libreoffice-impress	V:149, I:415	9930	LO	プレゼンテーション
calligrastage	V:0, I:5	5102	KDE	プレゼンテーション
libreoffice-base	V:116, I:289	7465	LO	データベース管理
kexi	V:0, I:2	7576	KDE	データベース管理
libreoffice-draw	V:150, I:416	14596	LO	ベクトル画像エディター (ドロー)
inkscape	V:42, I:188	84892	GNOME	ベクトル画像エディター (ドロー)
karbon	V:0, I:6	3473	KDE	ベクトル画像エディター (ドロー)
dia	V:4, I:30	3727	GTK	フローチャートやダイアグラムエディター
gimp	V:58, I:315	19706	GTK	ビットマップ画像エディター (ペイント)
shotwell	V:16, I:217	6451	GTK	デジタル写真オーガナイザー
digikam	V:2, I:12	2646	KDE	デジタル写真オーガナイザー
darktable	V:6, I:17	22902	GTK	lighttable and darkroom for photographers
planner	V:0, I:4	1146	GNOME	プロジェクト管理
calligraplan	V:0, I:2	18517	KDE	プロジェクト管理
gnucash	V:3, I:11	32304	GNOME	個人会計
homebank	V:0, I:2	1114	GTK	個人会計
lilypond	V:0, I:8	7337	-	music typesetter
kmymoney	V:0, I:2	12036	KDE	個人会計
librecad	V:3, I:18	7893	Qt アプリ	computer-aided design (CAD) system (2D)
freecad	I:19	53	Qt アプリ	computer-aided design (CAD) system (3D)
kicad	V:2, I:15	98707	GTK	electronic schematic and PCB design software
xsane	V:14, I:165	2346	GTK	スキャナーのフロントエンド
libreoffice-math	V:144, I:419	1963	LO	数式エディター
calibre	V:8, I:35	55048	KDE	e-book コンバーターとライブラリーの管理
fbreader	V:1, I:14	3074	GTK	e-book リーダー
evince	V:105, I:313	956	GNOME	文書 (pdf) ビューワー
okular	V:43, I:115	14646	KDE	文書 (pdf) ビューワー
x11-apps	V:27, I:454	2437	純粋な X アプリ	xeyes(1) など。
x11-utils	V:171, I:581	712	純粋な X アプリ	xev(1), xwininfo(1) など。

Table 7.3: 特筆すべき GUI アプリケーションの一覧

7.5 フォント

Many useful scalable fonts are available for users on Debian. User's concern is how to avoid redundancy and how to configure parts of installed fonts to be disabled. Otherwise, useless font choices may clutter your GUI application menus.

Debian system uses [FreeType](#) 2.0 library to rasterise many scalable font formats for screen and print:

- [Type 1 \(PostScript\) フォント](#)、3 次 [ベジエ曲線](#) を使用 (ほぼ廃れた形式)
- [TrueType フォント](#)、2 次 [ベジエ曲線](#) を使用 (よい選択肢)
- [OpenType フォント](#)、3 次 [ベジエ曲線](#) を使用 (最良の選択肢)

7.5.1 基本的なフォント

The following table is compiled in the hope to help users to chose appropriate scalable fonts with clear understanding of the metric compatibility and the glyph coverage. Most fonts cover all Latin fonts, Greek, and Cyril character glyphs. The final choice of activated fonts can also be affected by your aesthetics. These fonts can be used for the screen display or for the paper printing.

Here:

- "MCM" stands for "metric compatible with fonts provided by Microsoft"
- "MCMATC" stands for "metric compatible with fonts provided by Microsoft: [Arial](#), [Times New Roman](#), [Courier New](#)"
- "MCAHTC" stands for "metric compatible with fonts provided by [Adobe](#): Helvetica, Times, Courier"
- Numbers in font type columns stands for the rough relative "M" width for the same point size font.
- "P" in mono font type columns stands for its usability for programming having clearly distinguishable "0"/"O" and "1"/"l"/"I".
- The `ttf-mscorefonts-installer` package downloads Microsoft's "[Core fonts for the Web](#)" and installs [Arial](#), [Times New Roman](#), [Courier New](#), [Verdana](#), These installed font data are non-free data.

Many free Latin fonts have their lineage traced to [URW Nimbus](#) family or [Bitstream Vera](#).

ティップ

If your locale needs fonts not covered well by the above fonts, please use aptitude to check under task packages listed under "Tasks" -> "Localization". The font packages listed as "Depends:" or "Recommends:" in the localization task packages are the primary candidates.

7.5.2 フォントのラスタ化

Debian uses [FreeType](#) to rasterize fonts. Its font choice infrastructure is provided by the [Fontconfig](#) font configuration library.

ティップ

Some font packages such as `fonts-noto*` install too many fonts. You may also want to keep some font packages installed but disabled under the normal use situation. The multiple [glyphs](#) are expected for some [Unicode](#) code points due to [Han unification](#) and unwanted glyphs may be chosen by the unconfigured Fontconfig library. One of the most annoying case is "U+3001 IDEOGRAPHIC COMMA" and "U+3002 IDEOGRAPHIC FULL STOP" among CJK countries. You can avoid this problematic situation easily by configuring font availability using Font Manager GUI ([font-manager](#)).

You can list font configuration state from the command line, too.

- `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)`

You can configure font configuration state from the text editor but this is non-trivial. See `fonts.conf(5)`.

パッケージ	ポップコン	サイズ	サンセリフ	セリフ	等幅	フォントについての注釈
fonts-cantarell	V:37, I:273	836	59	-	-	Cantarell (GNOME 3, display)
fonts-noto	I:133	35	61	63	40	Noto fonts (Google, multi-lingual with CJK)
fonts-dejavu	I:462	39	58	68	40	DejaVu (GNOME 2, MCM: Verdana , extended Bitstream Vera)
fonts-liberation2	V:48, I:333	4261	56	60	40	Liberation fonts for LibreOffice (Red Hat, MCMATC)
fonts-croscore	V:10, I:47	5278	56	60	40	Chrome OS: Arimo, Tinos and Cousine (Google, MCMATC)
fonts-crosextra-carlito	I:276	2732	57	-	-	Chrome OS: Carlito (Google, MCM: Calibri)
fonts-crosextra-caladea	I:275	258	-	55	-	Chrome OS: Caladea (Google, MCM: Cambria) (Latin only)
fonts-freefont-ttf	V:55, I:257	6656	57	59	40	GNU FreeFont (extended URW Nimbus)
fonts-quicksand	I:358	392	56	-	-	Debian task-desktop, Quicksand (display, Latin only)
fonts-hack	V:9, I:89	2508	-	-	40 P	A typeface designed for source code Hack (Facebook)
fonts-sil-gentiumplus	I:39	5756	-	54	-	Gentium SIL
fonts-sil-charis	V:2, I:12	8900	-	59	-	Charis SIL
fonts-urw-base35	V:46, I:94	11095	56	60	40	URW Nimbus (Sans, Roman No. 9 L , Mono L , MCAHTC)
fonts-ubuntu	V:1, I:5	4103	58	-	33 P	Ubuntu fonts (display)
fonts-terminus	V:0, I:1	450	-	-	33	Cool retro terminal fonts
ttf-mscorefonts-installer	V:1, I:61	92	56?	60	40	Downloader of Microsoft non-free fonts (see below)

Table 7.4: List of notable [TrueType](#) and [OpenType](#) fonts

パッケージ	ポップコン	サイズ	説明
libfreetype6	V:432, I:995	896	FreeType フォントラスタライブラリー
libfontconfig1	V:432, I:860	526	Fontconfig 、フォント設定ライブラリー
fontconfig	V:353, I:757	583	fc - *: Fontconfig の CLI コマンド
font-manager	V:1, I:9	1015	Font Manager: Fontconfig の GUI コマンド
nautilus-font-manager	V:0, I:0	37	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

7.6 サンドボックス

Many mostly GUI applications on Linux are available in binary formats from non-Debian sources.

- [AppImage -- Linux apps that run anywhere](#)
- [FLATHUB -- Apps for Linux, right here](#)
- [snapcraft -- The app store for Linux](#)



警告

Binaries from these sites may include proprietary non-free software packages.

There is some *raison d'être* for these binary format distributions for Free Software aficionados using Debian since these can accommodate clean set of libraries used for each application by the respective upstream developer independent of the ones provided by Debian.

The inherent risk of running external binaries can be reduced by using the [sandbox environment](#) which leverages modern Linux security features (see [項4.7.4](#)).

- For binaries from AppImage and some upstream sites, run them in [firejail](#) with [manual configuration](#).
- For binaries from FLATHUB, run them in [Flatpak](#) . (No manual configuration required.)
- For binaries from snapcraft, run them in [Snap](#) . (No manual configuration required. Compatible with daemon programs.)

The `xdg-desktop-portal` package provides a standardized API to common desktop features. See [xdg-desktop-portal \(flatpak\)](#) and [xdg-desktop-portal \(snap\)](#)

パッケージ	ポプコン	サイズ	説明
flatpak	V:31, I:35	6562	Flatpak application deployment framework for desktop apps
gnome-software-plugin-flatpak	V:4, I:14	335	Flatpak support for GNOME Software
snapd	V:53, I:58	55035	Daemon and tooling that enable snap packages
gnome-software-plugin-snap	V:0, I:2	237	Snap support for GNOME Software
xdg-desktop-portal	V:125, I:209	1425	desktop integration portal for Flatpak and Snap
xdg-desktop-portal-gtk	V:71, I:205	921	xdg-desktop-portal backend for gtk (GNOME)
xdg-desktop-portal-kde	V:1, I:4	842	xdg-desktop-portal backend for Qt (KDE)
xdg-desktop-portal-wlr	V:0, I:0	83	xdg-desktop-portal backend for wlroots (Wayland)
firejail	V:1, I:5	1370	a SUID security sandbox program firejail for use with AppImage

Table 7.6: List of notable sandbox environment and related packages

This sandbox environment technology is very much like apps on smart phone OS where apps are executed under controlled resource accesses.

Some large GUI applications such as web browsers on Debian also use sandbox environment technology internally to make them more secure.

7.7 リモートデスクトップ

パッケージ	ポプコン	サイズ	プロトコル	説明
gnome-remote-desktop	V:14, I:23	207	RDP, RFB (VNC)	GNOME Remote Desktop サーバ
vinagre	V:9, I:222	4249	RDP, RFB (VNC), SPICE, SSH	Vinagre: GNOME remote desktop client
remmina	V:12, I:47	550	RDP, RFB (VNC), SPICE, SSH, ...	Remmina: GTK remote desktop client
krdc	V:2, I:21	3538	RDP, RFB (VNC)	KRDC: KDE remote desktop client
guacd	V:0, I:0	83	RDP, RFB (VNC), SSH / HTML5	Apache Guacamole: clientless remote desktop gateway (HTML5)
virt-viewer	V:4, I:51	1545	RFB (VNC), SPICE	Virtual Machine Manager's GUI display client of guest OS

Table 7.7: List of notable remote access server

7.8 X サーバ接続

There are several ways to connect from an application on a remote host to the X server including xwayland on the local host.

パッケージ	ポプコン	サイズ	コマンド	説明
openssh-server	V:701, I:844	1567	sshd with option X11-forwarding	SSH サーバ (セキュア)
openssh-client	V:809, I:997	4298	ssh -X	SSH クライアント (セキュア)
xauth	V:155, I:951	86	xauth	X 認証ファイルユーティリティ
x11-xserver-utils	V:272, I:516	511	xhost	X のサーバアクセス制御

Table 7.8: X サーバへの接続方法のリスト

7.8.1 X サーバローカル接続

Access to the local X server by the local applications which use X core protocol can be locally connected through a local UNIX domain socket. This can be authorized by the authority file holding [access cookie](#). The authority file location is identified by the "\$XAUTHORITY" environment variable and X display is identified by the "\$DISPLAY" environment variable. Since these are normally set automatically, no special action is needed, e.g. "gitk" as the following.

```
username $ gitk
```

注意

For xwayland, XAUTHORITY holds value like "/run/user/1000/.mutter-Xwaylandauth.YVSU30".

7.8.2 X サーバリモート接続

Access to the local X server display from the remote applications which use X core protocol is supported by using the X11 forwarding feature.

- ローカルホスト上で `gnome-terminal` を開きます。
- Run `ssh(1)` with `-X` option to establish a connection with the remote site as the following.

```
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次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。

See [項6.3](#) for SSH/SSHD.



警告

A remote [TCP/IP](#) to the X server is disabled by default on the Debian system for security reasons. Don't enable them by simply setting “`xhost +`” nor by enabling [XDMCP connection](#), if you can avoid it.

7.8.3 X サーバ `chroot` 接続

Access to the X server by the applications which use X core protocol and run on the same host but in an environment such as `chroot` where the authority file is not accessible, can be authorized securely with `xhost` by using the [User-based access](#), e.g. “`gitk`” as the following.

```
username $ xhost + si:localuser:root ; sudo chroot /path/to
# cd /src
# gitk
# exit
username $ xhost -
```

7.9 クリップボード

For clipping text to clipboard, see [項1.4.4](#).

For clipping graphics to clipboard, see [項11.6](#).

Some CLI commands can manipulate character clipboard (PRIMARY and CLIPBOARD), too.

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:9, I:43	59	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:10, I:51	64	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:0, I:1	129	Wayland	wl-copy wl-paste : command line interface to Wayland clipboard
gpm	V:11, I:16	540	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: List of programs related to manipulating character clipboard

Chapter 8

I18N と L10N

アプリケーションソフトの多言語化 (M17N) と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

There are 17, 18, or 10 letters between "m" and "n", "i" and "n", or "l" and "n" in multilingualization, internationalization, and localization which correspond to M17N, I18N, and L10N. See [Introduction to i18n](#) for details.

8.1 ロケール

The behavior of programs supporting internationalization are configured by the environment variable "\$LANG" to support localization. Actual support of locale dependent features by the `libc` library requires to install `locales` or `locales-all` packages. The `locales` package requires to be initialized properly.

If neither `locales` or `locales-all` package are installed, support of locale features are lost and system uses US English messages and handles data as **ASCII**. This behavior is the same way as "\$LANG" is set by "LANG=", "LANG=C", or "LANG=POSIX".

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。

The current Debian desktop GUI system normally sets the locale under GUI environment as "LANG=xx_YY.UTF-8". Here, "xx" is [ISO 639 language codes](#) and "YY" is [ISO 3166 country codes](#). These values are set by the desktop configuration GUI dialogue and change the program behavior. See [項1.5.2](#)

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)。

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。

Text encoding system **UTF-8** fits Unicode code points into a sensible 8 bit data stream mostly compatible with the ASCII data processing system. This makes **UTF-8** the modern preferred choice. **UTF** stands for Unicode Transformation Format. When **ASCII** plain text data is converted to **UTF-8** one, it has exactly the same content and size as the original ASCII one. So you loose nothing by deploying UTF-8 locale.

Under **UTF-8** locale with the compatible application program, you can display and edit any foreign language text data as long as required fonts and input methods are installed and enabled. For example under "**LANG=fr_FR.UTF-8**" locale, **gedit(1)** (text editor for the GNOME Desktop) can display and edit Chinese character text data while presenting menus in French.

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守する際に、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。

The **locales** package does **not** come with pre-compiled locale data. You need to configure it as:

```
# dpkg-reconfigure locales
```

このプロセスは 2 段階あります。

1. Select all required locale data to be compiled into the binary form. (Please make sure to include at least one UTF-8 locale)
2. Set the system wide default locale value by creating "**/etc/default/locale**" for use by PAM (see 項4.5).

The system wide default locale value set in "**/etc/default/locale**" may be overridden by the GUI configuration for GUI applications.

注意

Actual traditional encoding system can be identified by "**/usr/share/i18n/SUPPORTED**". Thus, the "**LANG=en_US**" is "**LANG=en_US.ISO-8859-1**".

The **locales-all** package comes with all locale data pre-compiled but doesn't creating "**/etc/default/locale**".

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交換 (項10.1.7参照下さい) 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 メモリー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。このメモリースティックを次にマウントする機会には 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](#) をデフォルトで使います。この旧式クライアントへのデフォルトは"/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ーン](#)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 can accommodate priority list of translation languages with `$LANGUAGE` environment variable. For example:

```
$ export LANGUAGE="pt:pt_BR:es:it:fr"
```

For more, see `info gettext` and read the section "The LANGUAGE variable".

8.1.5 ロケールの効果

`sort(1)` のソート順はロケールの言語選択に影響されます。スペイン語と英語のロケールではソート順が違います。`ls(1)` の日付形式はロケールに影響されます。"`LANG=C ls -l`" と "`LANG=en_US.UTF-8`" の日付形式は違います (項9.3.4を参照)。

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一千一百一点一は"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かもしれません。

Each detail feature of "`$LANG`" environment variable may be overridden by setting "`$LC_*`" variables. These environment variables can be overridden again by setting "`$LC_ALL`" variable. See `locale(7)` manpage for the details. Unless you have strong reason to create complicated configuration, please stay away from them and use only "`$LANG`" variable set to one of the UTF-8 locales.

8.2 キーボード入力

8.2.1 The keyboard input for Linux console and X Window

Debian システムは `keyboard-configuration` と `console-setup` 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
# dpkg-reconfigure console-setup
```

For the Linux console and the X Window system, this updates configuration parameters in `/etc/default/keyboard` and `/etc/default/console-setup`. This also configures the Linux console font. Many non-ASCII characters including accented characters used by many European languages can be made available with [dead key](#), [AltGr key](#), and [compose key](#).

8.2.2 Wayland 向けのキーボード入力

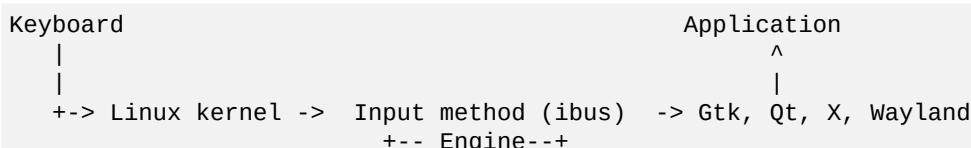
For GNOME on Wayland desktop system, 項8.2.1 can't support non-English European languages. [IBus](#) was made to support not only Asian languages but also European languages. The package dependency of GNOME Desktop Environment recommends "ibus" via "gnome-shell". The code of "ibus" has been updated to integrate `setxkbmap` and XKB option functionalities. You need to configure `ibus` from "GNOME Settings" or "GNOME Tweaks" for the multilingualized keyboard input.

注意

If `ibus` is active, your classic X keyboard configuration by the `setxkbmap` may be overridden by `ibus` even under classic X-based desktop environment. You can disable installed `ibus` using `im-config` to set input method to "None". For more, see [Debian Wiki on keyboard](#).

8.2.3 IBus を使うインプットメソッドのサポート

Since GNOME Desktop Environment recommends "ibus" via "gnome-shell", "ibus" is the good choice for input method. アプリケーションへの多言語入力は次のように処理されます。



IBus とそのエンジンパッケージの一覧は以下の通りです。

注意

For Chinese, "fcitx" may be an alternative input method framework. For Emacs aficionados, "uim" may be an alternative. Either cases, you may need to do extra manual configuration with `im-config`. Some old classic [input methods](#) such as "kinput2" may still exist in Debian repository but are not recommended for the modern environment.

8.2.4 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") 下で起動すると非常に便利です。Wayland 環境下で IBus を使ってどう実現したかを以下に記します。

1. Install the Japanese input tool package `ibus-mozc` (or `ibus-anthy`) with its recommended packages such as `im-config`.

パッケージ	バージョン	サイズ	サポートされたロケール
ibus	V:82, I:99	1584	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:1, I:2	1006	日本語
ibus-anthy	V:1, I:1	8790	, ,
ibus-kkc	V:0, I:0	214	, ,
ibus-skk	V:0, I:0	244	, ,
ibus-pinyin	V:0, I:1	1434	中国語 (zh_CN 用)
ibus-chewing	V:0, I:0	415	, , (zh_TW 用)
ibus-hangul	V:0, I:1	288	韓国語
ibus-table	V:0, I:1	1809	IBus 用のテーブルエンジン
ibus-table-thai	I:0	47	タイ語
ibus-unikey	V:0, I:0	318	ベトナム語
ibus-m17n	V:0, I:1	187	多言語: インド系言語、アラビア語、他

Table 8.1: IBus とエンジンパッケージのリスト

- 有効になっていない場合は”Settings” → ”Keyboard” → ”Input Sources” → click ”+” in ”Input Sources” → ”Japanese” → ”Japanese mozc (or anthy)”を選択し”Add”をクリックします。
- You may chose as many input sources.
- ユーザーアカウントに再ログインします。
- Setup each input source by right clicking the GUI toolbar icon.
- インプットソース間を、SUPER-SPACE を用いて切り替えます。(SUPER は通常 Windows キーです。)

ティップ

If you wish to have access to alphabet only keyboard environment with the physical Japanese keyboard on which shift-2 has " (double quotation mark) engraved, you select "Japanese" in the above procedure. You can enter Japanese using "Japanese mozc (or anthy)" with physical "US" keyboard on which shift-2 has @ (at mark) engraved.

- The GUI menu entry for `im-config(8)` is "Input method".
- 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 environment (第7章) can display any characters in the UTF-8 as long as required fonts are installed and enabled. (The encoding of the original font data is taken care and transparent to the user.)

8.4 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](#) 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 The console tips

There are some utility programs to help your console activities.

パッケージ	ポップコン	サイズ	説明
mc	V:55, I:234	1492	項1.3を参照
bsdutils	V:708, I:999	394	<code>script</code> command to make typescript of terminal session
screen	V:114, I:283	1014	VT100/ANSI ターミナルエミュレーションを使つてのターミナルマルチプレクサ
tmux	V:35, I:143	830	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)
fzf	V:2, I:7	2637	ファジーテキストファインダ
fzy	V:0, I:0	54	ファジーテキストファインダ
rlwrap	V:1, I:20	297	readline feature command line wrapper
ledit	V:0, I:14	306	readline feature command line wrapper
rlfe	V:0, I:0	49	readline feature command line wrapper

Table 9.1: List of programs to support console activities

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。このような事は次のようにして `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
```

- Use `gnome-terminal` with the `extend-line-buffer` for scrollbar.
- Use `screen` with `^A H` (see 項9.1.2) to perform recording of console.
- Use `vim` with `":terminal"` to enter the terminal mode. Use `"Ctrl-W N"` to exit from terminal mode to normal mode. Use `":w typescript"` to write the buffer to a file.
- Use `emacs` with `"M-x shell"`, `"M-x eshell"`, or `"M-x term"` to enter recording console. Use `"C-x C-w"` to write the buffer to a file.

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が実行中の作業を失いたくありません。
6. 次のようないかなる方法でも、`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き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
<code>^A ?</code>	ヘルプスクリーンを表示 (キーバインディングを表示)
<code>^A c</code>	新規ウィンドウを作成しそれに切り替える
<code>^A n</code>	次のウィンドウに切り替える
<code>^A p</code>	前のウィンドウに切り替える
<code>^A 0</code>	0 番のウィンドウに切り替える
<code>^A 1</code>	1 番のウィンドウに切り替える
<code>^A w</code>	ウィンドウのリストを表示
<code>^A a</code>	Ctrl-A を現在のウィンドウにキーボード入力として送る
<code>^A h</code>	現在のウィンドウのハードコピーをファイルに書く
<code>^A H</code>	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
<code>^A ^X</code>	ターミナルをロック (パスワードで保護)
<code>^A d</code>	ターミナルから screen のセッションをデタッチ
<code>^A DD</code>	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 Navigating around directories

In 項1.4.2, 2 tips to allow quick navigation around directories are described: `$CDPATH` and `mc`.

If you use fuzzy text filter program, you can do without typing the exact path. For `fzf`, include following in `~/.bashrc`.

```
FZF_KEYBINDINGS_PATH=/usr/share/doc/fzf/examples/key-bindings.bash
if [ -f $FZF_KEYBINDINGS_PATH ]; then
    . $FZF_KEYBINDINGS_PATH
fi
FZF_COMPLETION_PATH=/usr/share/doc/fzf/examples/completion.bash
if [ -f $FZF_COMPLETION_PATH ]; then
    . $FZF_COMPLETION_PATH
fi
```

たとえば:

- You can jump to a very deep subdirectory with minimal efforts. You first type `"cd ***"` and press `Tab`. Then you will be prompted with candidate paths. Typing in partial path strings, e.g., `s/d/b foo`, will narrow down candidate paths. You select the path to be used by `cd` with cursor and return keys.
- You can select a command from the command history more efficiently with minimal efforts. You press `Ctrl-R` at the command prompt. Then you will be prompted with candidate commands. Typing in partial command strings, e.g., `vim d`, will narrow down candidates. You select the one to be used with cursor and return keys.

9.1.4 Readline wrapper

Some commands such as `/usr/bin/dash` which lacks command line history editing capability can add such functionality transparently by running under `rlwrap` or its equivalents.

```
$ rlwrap dash -i
```

This provides convenient platform to test subtle points for `dash` with friendly `bash`-like environment.

9.2 Vim のカスタム化

After you learn basics of `vim`(1) through 項1.4.8, please read Bram Moolenaar's "[Seven habits of effective text editing \(2000\)](#)" to understand how `vim` should be used.

**注意**

Don't try to change the default key bindings without very good reasons.

9.2.1 Customizing vim with internal features

The behavior of vim can be changed significantly by enabling its internal features through the Ex-mode commands such as "set ..." to set vim options.

These Ex-mode commands can be included in user's vimrc file, traditional "~/.vimrc" or git-friendly "~/.vim/vimrc". Here is a very simple example ¹:

```
colorscheme murphy          " from /usr/share/vim/vim??/colors/*.vim
filetype plugin indent on   " filetype aware behavior
syntax enable              " Syntax highlight
"set spelllang=en_us       " Spell check language as en_us
"set spell                 " Enable spell check
set autoindent             " Copy indent from current line
set smartindent            " More than autoindent (Drop/Pop after {/})
set nosmarttab             " <Tab>-key always inserts blanks
set backspace=indent,eol,start " Back space through everything
set laststatus=2           " Always show status line
set statusline=%<%f%m%r%h%w%=%y[U+%04B]%2l/%2L=%P,%2c%V
```

9.2.2 Customizing vim with external packages

Simple customization to enable securemodelines and classical IDE can be enabled by installing [vim-scripts](#) package and appending the following to user's vimrc file.

```
packadd! securemodelines
packadd! winmanager
let mapleader = ' '
" Toggle paste mode with <SPACE>p
set pastetoggle=<leader>p
" IDE-like UI for files and buffers with <space>w
nnoremap <leader>w          :WMToggle<CR>
" Use safer keys <SPACE>? for moving to another window
nnoremap <leader>h          <C-W>h
nnoremap <leader>j          <C-W>j
nnoremap <leader>k          <C-W>k
nnoremap <leader>l          <C-W>l
```

The new native Vim package system works nicely with "git" and "git submodule". One such example configuration can be found at [my git repository: dot-vim](#). This does essentially:

- By using "git" and "git submodule", latest external packages, such as "*name*", are placed into ~/.vim/pack/*/opt/*name* and similar.
- By adding :packadd! *name* line to user's vimrc file, these packages are placed on runtimepath.
- Vim loads these packages on runtimepath during its initialization.
- At the end of its initialization, tags for the installed documents are updated with "helptags ALL".

¹More elaborate customization examples: "[Vim Galore](#)", "[sensible.vim](#)", "[#vim Recommendations](#)" ...

For more, please start vim with `"vim --startuptime vimstart.log"` to check actual execution sequence and time spent for each step.

Interesting external plugin packages can be found:

- [Vim - the ubiquitous text editor](#) -- The official upstream site of Vim and vim scripts
- [VimAwesome](#) -- The listing of Vim plugins
- [vim-scripts](#) -- Debian package: a collection of vim scripts

It is quite confusing to see too many ways² to manage and load these external packages to vim. Checking the original information is the best cure.

key strokes	information
:help package	explanation on the vim package mechanism
:help runtimepath	explanation on the runtimepath mechanism
:version	internal states including candidates for the vimrc file
:echo \$VIM	the environment variable "\$VIM" used to locate the vimrc file
:set runtimepath?	list of directories which will be searched for all runtime support files
:echo \$VIMRUNTIME	the environment variable "\$VIMRUNTIME" used to locate various system provided runtime support files

Table 9.3: Information on the initialization of vim

9.3 データの記録と表現

9.3.1 ログデーモン

多くのプログラムは`/var/log/`ディレクトリーの下にそれぞれの活動を記録します。

項3.4と項3.3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。

9.3.3 テキストデータのカスタム化表示

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11.1.8参照下さい) はテキストデータを綺麗に表示できますが、汎用エディター (項1.4.6参照下さい) 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。

²[vim-pathogen](#) was popular.

パッケージ	ポップコン	サイズ	説明
logwatch	V:16, I:18	2265	綺麗な出力の Perl で書かれたログアナライザー
fail2ban	V:116, I:129	2092	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
analog	V:3, I:105	3534	ウェブサーバーのログアナライザー
awstats	V:9, I:14	6910	強力で機能の多いウェブサーバーのログアナライザー
sarg	V:2, I:3	843	squid の分析レポートジェネレーター
pflogsumm	V:2, I:4	111	Postfix ログ項目サマライザー
syslog-summary	V:0, I:2	30	syslog ログファイルの内容をまとめる
fwlogwatch	V:0, I:0	478	ファイアウォールログアナライザー
squidview	V:0, I:1	189	squid の access.log ファイルのモニターと分析
swatch	V:0, I:0	101	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, I:0	1119	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, I:0	44	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

9.3.4 時間と日付のカスタム化表示

The default display format of time and date by the `ls -l` command depends on the **locale** (see 項1.2.6 for value). The `$LANG` variable is referred first and it can be overridden by the `$LC_TIME` or `$LC_ALL` exported environment variables.

The actual default display format for each locale depends on the version of the standard C library (the `libc6` package) used. I.e., different releases of Debian had different defaults. For iso-formats, see [ISO 8601](#).

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)。

時間スタイル値	ロケール	時間と日付の表示
<code>iso</code>	任意	01-19 00:15
<code>long-iso</code>	任意	2009-01-19 00:15
<code>full-iso</code>	任意	2009-01-19 00:15:16.000000000 +0900
<code>locale</code>	C	Jan 19 00:15
<code>locale</code>	en_US.UTF-8	Jan 19 00:15
<code>locale</code>	es_ES.UTF-8	ene 19 00:15
<code>+%d.%m.%y %H:%M</code>	任意	19.01.09 00:15
<code>+%d.%b.%y %H:%M</code>	C または en_US.UTF-8	19.Jan.09 00:15
<code>+%d.%b.%y %H:%M</code>	es_ES.UTF-8	19.ene.09 00:15

Table 9.5: Display examples of time and date for the `ls -l` command with the **time style value**

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'
```

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、次を試してみてください:

```
$ 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に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'` をエク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

対話環境でシェルの `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
gnome-screenshot	V:27, I:264	1173	Wayland
flameshot	V:5, I:11	1510	Wayland
gimp	V:58, I:315	19706	Wayland + X
x11-apps	V:27, I:454	2437	X
imagemagick	I:373	220	X
scrot	V:7, I:76	123	X

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

There are specialized tools to record changes in configuration files with help of DVCS and to make system snapshots on [Brfs](#).

パッケージ	ポプコン	サイ ズ	説明
etckeeper	V:28, I:32	176	Git (デフォルト) か Mercurial か Bazaar を使って設定ファイルとそのメタデーターを保存 (新規)
timeshift	V:2, I:4	3158	system restore utility using rsync or BTRFS snapshots
snapper	V:2, I:3	2119	Linux filesystem snapshot management tool

Table 9.7: 設定の履歴を記録するパッケージのリスト

You may also think about local script 項[10.2.2](#) approach.

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	ポップコン	サイズ	説明
coreutils	V:901, I:999	17478	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:708, I:999	394	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:742, I:999	1648	"/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:433, I:831	793	"/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:13, I:249	129	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:168, I:190	1923	sar(1)、iostat(1)、mpstat(1)、…: Linux 用のシステムパフォーマンスツール
isag	V:0, I:4	117	sysstat の対話型システムアクティビティグラフ化ソフト
lsof	V:393, I:944	451	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:16, I:151	2367	strace(1): システムコールやシグナルを追跡
ltrace	V:1, I:22	363	ltrace(1): ライブラリーコールを追跡
xtrace	V:0, I:0	353	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:8, I:204	662	powertop(1): システムの電力消費情報
cron	V:800, I:997	263	cron(8) デーモンからバックグラウンドでスケジュール通りプロセスを実行
anacron	V:399, I:463	104	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:154, I:278	161	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

コマンドが起動し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次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

プロ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)` でプロ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次のようにして追跡できます。

```
$ sudo strace ls
```

ティップ

Use **strace-graph** script found in `/usr/share/doc/strace/examples/` to make a nice tree view

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)` によって次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)` によって次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lobs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) に関する上級の使用法の詳細は項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を参照ください。

For the [graphical user interface \(GUI\)](#) compliant to the [freedesktop.org](#) standards, the *.desktop files in the /usr/share/applications directory provide necessary attributes for the GUI menu display of each program. Each package which is compliant to freedesktop.org’s xdg menu system installs its menu data provided by ”*.desktop” under ”/usr/share/applications/”. Modern desktop environments which are compliant to freedesktop.org standard use these data to generate their menu using the xdg-utils package. See ”/usr/share/doc/xdg-utils/README”.

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 3 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

同様に、もしこれらのベースディレクトリーの下に 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で自動実行されます。[Desktop Application Autostart Specification](#) を参照ください。

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。xdg-user-dirs-update(1) を参照ください。

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:
 - GNOME3 デスクトップ: "Settings" → "System" → "Details" → "Default Applications"
 - KDE デスクトップ: "K" → "Control Center" → "KDE Components" → "Component Chooser"
 - 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ブブラウザ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

X 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、次のようにして X 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) コマンドを次のように実行して 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 /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

連続的に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` has low level capability to schedule programs to run without `crond` daemon. For example, `/lib/systemd/system/apt-daily.service` and `/lib/systemd/system/apt-daily.service` set up daily apt download activities. See `systemd.timer(5)`.

9.4.15 Alt-SysRq キー

Pressing Alt-SysRq (PrtScr) followed by one keys does the magic of rescuing control of the system.

Alt-SysRq に続くキー	アクションの説明
k	kill all processes on the current virtual console (SAK)
s	データ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 sync (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讀出し専用で再マウント (アンマウント、 umount)
r	X クラッシュの後でキーボードを raw (生コード発生) モードから復旧

Table 9.12: List of notable SAK command keys

See more on [Linux kernel user's and administrator's guide » Linux Magic System Request Key Hacks](#)

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 **sync** (同期) して **umount** (アンマウント) します。

The current (2021) Debian amd64 Linux kernel has `/proc/sys/kernel/sysrq=438=0b110110110`:

- 2 = 0x2 - enable control of console logging level (ON)
- 4 = 0x4 - enable control of keyboard (SAK, unraw) (ON)
- 8 = 0x8 - enable debugging dumps of processes etc. (OFF)
- 16 = 0x10 - enable sync command (ON)
- 32 = 0x20 - enable remount read-only (ON)
- 64 = 0x40 - enable signaling of processes (term, kill, oom-kill) (OFF)
- 128 = 0x80 - allow reboot/poweroff (ON)
- 256 = 0x100 - allow nicing of all RT tasks (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、次のようにしてチェックできます。

- `who(1)` shows who is logged on.
- `w(1)` shows who is logged on and what they are doing.
- `last(1)` shows listing of last logged in user.
- `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、次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 ハードウェアの識別

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)` がハードウェア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ツリーを閲覧することでハードウェアの識別ができます (項[1.2.12](#)参照下さい)。

パッケージ	ポップコン	サイズ	説明
pciutils	V:200, I:991	196	Linux PCI ユーティリティ: lspci(8)
usbutils	V:73, I:858	325	Linux USB ユーティリティ: lsusb(8)
nvme-cli	V:2, I:10	779	NVMe utilities for Linux: nvme(1)
pcmciautils	V:11, I:17	97	Linux のための PCMCIA ユーティリティ: pccardctl(8)
scsitools	V:0, I:3	375	SCSI ハードウェア管理のためのツール集: lsscsi(8)
procinfo	V:0, I:12	135	”/proc” から得られるシステム情報: lsdev(8)
lshw	V:11, I:96	850	ハードウェア設定に関する情報: lshw(1)
discover	V:37, I:948	90	ハードウェア識別システム: discover(8)

Table 9.13: ハードウェア識別ツールのリスト

パッケージ	ポップコン	サイズ	説明
console-setup	V:133, I:960	419	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
x11-xserver-utils	V:272, I:516	511	X サーバユーティリティ: xset(1) 、 xmodmap(1)
acpid	V:135, I:281	176	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:15, I:266	45	ACPI デバイス上の情報を表示するユーティリティ
sleepd	V:0, I:0	86	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:446, I:717	256	ハードディスクアクセスの最適化 (項9.6.9参照下さい)
smartmontools	V:150, I:208	2117	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:5, I:9	117	シリアルポートの管理ツール集
memtest86+	V:1, I:28	2391	メモリーハードウェア管理のためのツール集
scsitools	V:0, I:3	375	SCSI ハードウェア管理のためのツール集
setcd	V:0, I:1	35	コンパクトデバイスアクセス最適化
big-cursor	I:1	27	X のための大きなマウスカーソル

Table 9.14: ハードウェア設定ツールのリスト

9.5.4 ハードウェア設定

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ェア設定が付随する 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。

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ェアの時間

以下はシステムとハードウェアの時間を 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通常地域の時間が表示されますが、ハードウェアとシステムの時間は通常 [UTC\(GMT\)](#) を使います。

If the hardware time is set to UTC, change the setting to "UTC=yes" in the `"/etc/default/rcS"`.

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、`ntp` や `ntpd` 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)` を参照ください。

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](#)
- `ntp-doc` パッケージ

ティップ

`ntp` パッケージ中の `ntptrace(8)` 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 ターミナルの設定

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、`"$TERM"`、を `"xterm"` から `"xterm-r6"` のような機能限定版に変更します。詳細は `/usr/share/doc/libncurses5/FAQ` を参照下さい。`"dumb"` は `"$TERM"` の最低機能の共通項です。

9.5.7 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。

Application softwares may be configured not only to access sound devices directly but also to access them via some standardized sound server system. Currently, PulseAudio, JACK, and PipeWire are used as sound server system. See [Debian wiki page on Sound](#) for the latest situation.

各ポピュラーなデスクトップ環境では普通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	ポップコン	サイズ	説明
alsa-utils	V:337, I:456	2283	ALSA を設定し使用するユーティリティ
oss-compat	V:1, I:27	20	ALSA の下で <code>"/dev/dsp not found"</code> エラーを防ぐ OSS 互換性
pipewire	V:59, I:78	30	audio and video processing engine multimedia server - metapackage (unify JACK and PulseAudio)
pipewire-bin	V:67, I:76	1848	audio and video processing engine multimedia server - programs (unify JACK and PulseAudio)
pipewire-audio-client-libraries	V:0, I:1	380	audio and video processing engine multimedia server - audio client libraries to replace ALSA, JACK and PulseAudio (experimental)
pulseaudio	V:357, I:453	6774	PulseAudio server (standard)
libpulse0	V:292, I:586	971	PulseAudio client library (standard)
jackd	V:3, I:26	9	JACK Audio Connection Kit. (JACK) サーバー (低遅延)
libjack0	V:1, I:12	338	JACK Audio Connection Kit. (JACK) ライブラリー (低遅延)
libgstreamer1.0-0	V:374, I:567	5283	GStreamer: GNOME サウンドエンジン
libphonon4	I:101	680	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

9.5.8 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、次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

9.5.9 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。

次のようにすると `bash(1)` が使う `readline(3)` プログラムが“\a” (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
```

9.5.10 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…」と不思議かも知れません。

“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 行下の (次の例では 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 では次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

9.5.11 システムのセキュリティ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と整合性のチェックには、次のこと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 `freshclam(1)` 参照下さい。
- [Debian セキュリティー FAQ](#).
- [Securing Debian Manual](#).

パッケージ	ポップコン	サイズ	説明
logcheck	V:8, I:10	102	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:5, I:44	108	MD5 チェックサムを使ってインストールされたパッケージファイルを検証するユーティリティ
chkrootkit	V:5, I:24	971	ルートキット検出ソフト
clamav	V:12, I:57	775	Unix 用アンチウイルスユーティリティ - コマンドラインインターフェース
tiger	V:2, I:3	7822	システムセキュリティの脆弱性を報告
tripwire	V:2, I:3	11618	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:2, I:12	460	アクティブなパスワードクラッキングツール
aide	V:1, I:2	2049	先進的進入検出環境 - 静的ライブラリー
integrit	V:0, I:0	329	ファイル整合性確認プログラム
crack	V:0, I:1	152	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や整合性確認のためのツールリスト

次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監査ツールとしては完全には信頼できません。

9.6 データ保存のティップ

Linux の **live CDs** とかレスキューモードで **debian-installer CDs** であなたのシステムをブートすることでブートデバイスのデータストレージの再設定が簡単にできます。

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(1x)` 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。

Older PCs use the classic [Master Boot Record \(MBR\)](#) scheme to hold [disk partitioning](#) data in the first sector, i.e., [LBA](#) sector 0 (512 bytes).

Recent PCs with [Unified Extensible Firmware Interface \(UEFI\)](#), including Intel-based Macs, use [GUID Partition Table \(GPT\)](#) scheme to hold [disk partitioning](#) data not in the first sector.

`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)` がそれを置き換えています。

パッケージ	ポプコン	サイズ	説明
util-linux	V:899, I:999	4599	<code>fdisk(8)</code> と <code>cdisk(8)</code> を含む雑多なシステムユーティリティ
parted	V:367, I:548	307	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:16, I:127	2063	libparted ベースの GNOME パーティションエディター
gdisk	V:298, I:501	880	partition editor for the GPT/MBR hybrid disk
kpartx	V:19, I:30	87	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](#)参照下さい)、`"parted /dev/sdx mklablel gpt"` か `"parted /dev/sdx mklablel 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覚えておきます。

9.6.3 UUID を使ってパーティションをアクセス

Although reconfiguration of your partition or activation order of removable storage media may yield different names for partitions, you can access them consistently. This is also helpful if you have multiple disks and your BIOS/UEFI doesn't give them consistent device names.

- `"-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。
You can also probe it and other information with `"lsblk -f"`.

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ッグします。(削除したファイルを回復する `unde1` 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](#) status can be found at [Debian wiki on btrfs](#) and [kernel.org wiki on btrfs](#). It is expected to be the next default filesystem after the `ext4` filesystem.

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](#)参照下さい)。

パッケージ	バージョン	サイズ	説明
e2fsprogs	V:616, I:999	1511	ext2/ext3/ext4 ファイルシステムのためのユーティリティ
btrfs-progs	V:44, I:68	3959	btrfs ファイルシステムのためのユーティリティ
reiserfsprogs	V:15, I:29	1132	Reiserfs ファイルシステムのためのユーティリティ
zfsutils-linux	V:19, I:23	1497	utilities for the OpenZFS filesystem
dosfstools	V:132, I:508	315	FAT ファイルシステムのためのユーティリティ (Microsoft: MS-DOS, Windows)
exfatprogs	V:0, I:1	161	utilities for the exFAT filesystem maintained by Samsung.
exfat-fuse	V:25, I:429	71	read/write exFAT filesystem (Microsoft) driver for FUSE.
exfat-utils	V:19, I:430	231	utilities for the exFAT filesystem maintained by the exfat-fuse author.
xfsprogs	V:22, I:101	3281	XFS ファイルシステムのためのユーティリティ (SGI: IRIX)
ntfs-3g	V:227, I:494	1483	read/write NTFS filesystem (Microsoft: Windows NT, ...) driver for FUSE.
jfsutils	V:1, I:12	1577	JFS ファイルシステムのためのユーティリティ (IBM: AIX, OS/2)
reiser4progs	V:0, I:4	1373	Reiser4 ファイルシステムのためのユーティリティ
hfsprogs	V:0, I:7	389	HFS と HFS Plus ファイルシステムのためのユーティリティ (Apple: Mac OS)
zerofree	V:3, I:100	25	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、再起動時に `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)` を参照ください。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9.6.3参照下さい) は、"/dev/hda3" や"/dev/hda3" 等の通常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。

Since Linux 2.6.30, the kernel defaults to the behavior provided by "relatime" option.

See `fstab(5)` and `mount(8)`.

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/hda1`" を実行するとそ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/hda1`" を実行すると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/hda1`" の実行は `ext2` から `ext3` へと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/hda1 && fsck -pf /dev/hda1`" の実行は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ェアをチェックし `hdparm(8)` のマンページをチェックします。

例えば"/dev/hda" に対して"`hdparm -tT /dev/hda`" 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/hda`" 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

Solid state drive (SSD) is auto detected now.

Reduce unnecessary disk accesses to prevent disk wear out by mounting "tmpfs" on volatile data path in `/etc/fstab`.

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/hda” 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/hda” 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/hda” として有効にします。
5. 次のようにして 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。
 - restart the smartd(8) daemon by ”sudo systemctl restart smartmontools”.

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

通常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ーを指定できます。

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](#)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](#)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 Stretch 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mlink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。ソフトウェアによっては「ディレクトリーへのシmlink」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ください。

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mlink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mlink”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mlink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

例えば 2 番目の SCSI もしくはシリアル ATA ドライブ”/dev/sdb”等の、アンマウントされたドライブのディスクイメージファイル”disk.img”は cp(1) か dd(1) を用いれば次のようにして作れます。

```
# cp /dev/sdb disk.img
# dd if=/dev/sdb of=disk.img
```

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 のマスタートレコード (MBR) (項9.6.2参照下さい) のディスクイメージは、dd(1) を用いれば次のようにして作れます。

```
# dd if=/dev/hda of=mbr.img bs=512 count=1
# dd if=/dev/hda of=mbr-nopart.img bs=446 count=1
# dd if=/dev/hda 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きの MBR。
- ”mbr-part.img”: MBR のパーティションテーブルのみ。

ブートディスクとして SCSI ドライブもしくはシリアル ATA デバイスが使われる場合、”/dev/hda”を”/dev/sda”に置き換えて下さい。

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/hda”を”/dev/hda1”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 dd(1) を使ってサイズがマッチする例えば”/dev/sdb”という 2 番目の SCSI ドライブに次のようにして書き込むことができます。

```
# dd if=disk.img of=/dev/sdb
```

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次のようにして書き込むことができます。

```
# dd if=partition.img of=/dev/sdb1
```

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”は次のように [loop デバイス](#) を使いマウントしアンマウントできます。

```
# losetup -v -f partition.img
Loop device is /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。loop デバイスはパーティションをデフォルトでは管理しないので、次のようにそれをリセットする必要があります。

```
# modinfo -p loop # verify kernel capability
max_part:Maximum number of partitions per loop device
max_loop:Maximum number of loop devices
# losetup -a # verify nothing using the loop device
# rmmod loop
# modprobe loop max_part=16
```

これで、loop デバイスは 16 パーティションまで管理出来ます。

```
# losetup -v -f disk.img
Loop device is /dev/loop0
# fdisk -l /dev/loop0

Disk /dev/loop0: 5368 MB, 5368709120 bytes
255 heads, 63 sectors/track, 652 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x452b6464

    Device Boot      Start         End      Blocks   Id  System
/dev/loop0p1            1           600     4819468+   83   Linux
/dev/loop0p2          601           652       417690    83   Linux
# mkdir -p /mnt/loop0p1
# mount -t ext4 /dev/loop0p1 /mnt/loop0p1
# mkdir -p /mnt/loop0p2
# mount -t ext4 /dev/loop0p2 /mnt/loop0p2
...hack...hack...hack
# umount /dev/loop0p1
# umount /dev/loop0p2
# 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](#) デバイスを用いて次のようにして実現も出来ます。

```
# kpartx -a -v disk.img
...
# mkdir -p /mnt/loop0p2
# mount -t ext4 /dev/mapper/loop0p2 /mnt/loop0p2
...
...hack...hack...hack
# umount /dev/mapper/loop0p2
...
# kpartx -d /mnt/loop0
```

注意

[MBR](#) 等をスキップするオフセットを使った [loop デバイス](#) によっても、このようなディスクイメージの単一パーティションをマウント出来ます。しかしこれは失敗しがちです。

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”に次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 -f -v disk.img
Loop device is /dev/loop3
# zerofree /dev/loop3
# cp --sparse=always disk.img new.img
```

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) と mke2fs(8) を使って次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
```

[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 -f -v disk.img
Loop device is /dev/loop1
# mkfs.ext4 /dev/loop1
...hack...hack...hack
# losetup -d /dev/loop1
$ 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](#)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

”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) を使って次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次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3.1.2参照下さい) がブートに使われています。

次のようにすると 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 は、[cdrkit](#) が提供する wodim(1) にとっては単に大きな CD です。

使えるデバイスは次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/hda” というこのデバイスに ISO9660 イメージファイル”cd.iso”に wodim(1) を使って次のようにして書込みます。

```
# wodim -v -eject dev=/dev/hda 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/hda 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 umount /dev/hda”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、次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10.1.7参照下さい)。

9.8 バイナリーデータ

次に、ストレージメディア上のバイナリーデータ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。

パッケージ	ポプコン	サイズ	説明
coreutils	V:901, I:999	17478	ファイルをダンプする <code>od(1)</code> がある基本パッケージ (HEX, ASCII, OCTAL, ...)
bsdmainutils	V:53, I:953	27	ファイルをダンプする <code>hd(1)</code> があるユーティリティーパッケージ (HEX, ASCII, OCTAL, ...)
hexedit	V:1, I:12	73	バイナリーエディターとビューワー (HEX, ASCII)
bless	V:0, I:4	1028	フル機能の 16 進エディター (GNOME)
okteta	V:1, I:14	1504	フル機能の 16 進エディター (KDE4)
ncurses-hexedit	V:0, I:2	132	バイナリーエディターとビューワー (HEX, ASCII, EBCDIC)
beav	V:0, I:0	133	バイナリーエディターとビューワー (HEX, ASCII, EBCDIC, OCTAL, ...)

Table 9.21: バイナリーデータを閲覧や編集するパッケージのリスト

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、通常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数十進数互換コードです。

9.8.2 ディスクをマウントせずに操作

ディスクをマウントせずに読み出しや書き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
mtools	V:9, I:82	399	MSDOS ファイルをマウントせずに使うツール
hfsutils	V:0, I:6	184	HFS や HFS+ ファイルをマウントせずに使うツール

Table 9.22: ディスクをマウントせずに操作するパッケージのリスト

9.8.3 データの冗長性

Linux カーネルが提供するソフトウェア 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:4, I:26	271	ファイルのチェックと修理のためのパリティアーカイブセット
dvdaster	V:0, I:2	1741	CD/DVD メディアのデータロス/傷つき/老化の防止
dvbackup	V:0, I:0	413	MiniDV カメラレコーダを使うバックアップツール (rsbep(1)を提供)
vdmfec	V:0, I:0	97	前方エラー修正を使って失われたブロックの復元

Table 9.23: ファイルにデータの冗長性を追加するツールのリスト

9.8.4 データファイルの復元と事故の証拠解析

データファイルの復元と事故の証拠解析のツールがあります。

パッケージ	ポプコン	サイズ	説明
testdisk	V:3, I:37	1426	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
magicrescue	V:0, I:3	259	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
scalpel	V:0, I:4	87	質素で高性能なファイル彫刻刀
myrescue	V:0, I:3	83	破壊したハードディスクからデータを救出
extundelete	V:0, I:11	147	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
ext4magic	V:0, I:5	233	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
ext3grep	V:0, I:3	293	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, I:3	50	NTFS ファイルシステム上のデータ復元プログラム
gzrt	V:0, I:0	33	gzip 復元ツールキット
sleuthkit	V:2, I:25	1511	証拠解析のためのツール (Sleuthkit)
autopsy	V:0, I:1	1027	SleuthKit のための GUI
foremost	V:0, I:7	100	データ復元のための証拠解析アプリケーション
guymager	V:0, I:1	1035	Qt 使用の証拠解析用イメージ作成ソフト
dcfldd	V:0, I:5	106	証拠解析とセキュリティのための dd の強化版

Table 9.24: データファイルの復元と事故の証拠解析のリスト

ティップ

e2fsprogs パッケージ中の debugfs(8) の `list_deleted_inodes` または `unde1` 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。

```
$ split -b 2000m large_file
$ cat x* >large_file
```

**注意**

名前がかち合わないよう"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しょう。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。次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

次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
```

次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です。
- "zero.file" は 0 バイト長のファイルかもしれませんが。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 メモリースティック等のハードディスク類似デバイス全体のデータを完全に消すいくつかの方法があります。

**注意**

次のコマンドを実行する前にまず USB メモリースティックの場所を `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 SCSI ハードディスクかも知れませんがあなたの全システムのあるシリアル ATA ハードディスクかも知れません。

次のようにしてデータを 0 にリセットして全消去します。

```
# dd if=/dev/zero of=/dev/sda
```

次のようにしてランダムデータを上書きして全消去します。

```
# dd if=/dev/urandom of=/dev/sda
```

次のようにしてランダムデータ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
```

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/hda"` や `"/dev/sda"` 等のシステムハードディスクに対して同様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 メモリースティック) 上の使用されていない領域には消去されたデータ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 メモリースティックではこれで普通十分です。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回復出来ます。

例えば、次を試してみてください:

```
$ 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、(`lsuf` 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次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
$ lsuf |grep bar|grep less
less 4775 bozo 4r REG 8,3 4 2228329 /home/bozo/bar
$ rm bar
$ lsuf |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れない場合は、それを例えば”2738404”という [inode](#) で次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、通常の du(1) では見えませんが、ディスクスペースを消費します。これらは次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いうことです。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](#) facilitates automatic data encryption via native Linux kernel modules with minimal user efforts using [device-mapper](#).

パッケージ	ポプコン	サイズ	説明
cryptsetup	V:41, I:77	407	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:4, I:5	228	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, I:0	4688	utilities for Linux filesystem encryption (fscrypt)
libpam-fscrypt	V:0, I:0	4904	PAM module for Linux filesystem encryption (fscrypt)

Table 9.25: データー暗号化ユーティリティーのリスト

**注意**

データの暗号化には CPU 時間等の負担がかかります。その利益と負担の両天秤をします。

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 メモリースティック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。それを単に次のようにフォーマットします。

```
# badblocks -c 1024 -s -w -t random -v /dev/sdx
# fdisk /dev/sdx
... "n" "p" "1" "return" "return" "w"
# cryptsetup luksFormat /dev/sdx1
...
# cryptsetup open --type luks /dev/sdx1 sdx1
...
# ls -l /dev/mapper/
total 0
crw-rw---- 1 root root 10, 60 2008-10-04 18:44 control
brw-rw---- 1 root disk 254, 0 2008-10-04 23:55 sdx1
# mkfs.vfat /dev/mapper/sdx1
...
# cryptsetup luksClose sdx1
```

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通常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](#)参照下さい)。全て書込まれるデータが暗号化されている点が相違点です。上記と違い、例えば”[mkfs.ext4 /dev/sdx1](#)”を使って [ext4](#) 等の異なったフォーマットで記録メディアをフォーマットしても良いです。

注意

もしデータのセキュリティが本当に偏執狂的に気になるなら、上記例で複数回の重ね書きをする必要があるかもしれません。でもこの操作は非常に時間がかかります。

9.9.2 dm-crypt を使って swap パーティションを暗号化

オリジナルの”/etc/fstab”が次の内容と仮定します。

```
/dev/sda7 swap sw 0 0
```

[dm-crypt](#) を使って swap パーティションの暗号化を次のようにして有効にします

```
# aptitude install cryptsetup
# swapoff -a
# echo "cswap /dev/sda7 /dev/urandom swap" >> /etc/crypttab
```

```
# perl -i -p -e "s/\s\/dev\/sda7\/\s\/dev\/mapper\/cswap/" /etc/fstab
# systemctl restart cryptdisks
...
# swapon -a
```

9.9.3 dm-crypt/LUKS で暗号化されたディスクのマウント

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。

If you are reading this documentation, you probably don't need to compile Linux kernel by yourself.

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次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](#)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](#)参照下さい)
- モジュールが起動された際の modprobe(8) の引数により設定されるモジュール変数 (項[9.7.3](#)参照下さい)

See ”[The Linux kernel user's and administrator's guide](#) » [The kernel's command-line parameters](#)” for the detail.

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

For compiling some kernel-specific programs such as the kernel modules from the external source and the automounter daemon (amd), you must include path to the corresponding kernel headers, e.g. ”-I/usr/src/linux-particular-version/include/”, to your command line.

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	ポップコン	サイズ	説明
build-essential	I:500	20	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:147, I:973	122	bz2 ファイルのための圧縮と解凍ユーティリティ
libncurses5-dev	I:113	6	ncurses のためのデベロッパ用ライブラリーと文書
git	V:312, I:494	36515	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:32, I:518	228	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:442, I:989	112	initramfs をビルドするツール (Debian 固有)
dkms	V:99, I:210	295	動的カーネルモジュールサポート (DKMS) (汎用)
module-assistant	V:1, I:30	391	helper tool to make module package (Debian specific)
devscripts	V:8, I:54	2654	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`(8) と `update-initramfs`(8) と `mkinitramfs`(8) と `initramfs.conf` 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mlink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) (or its short form `m-a`) helps users to build and install module package(s) easily for one or more custom kernels.

[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 http://www.kernel.org/pub/linux/kernel/v3.11/linux-version.tar.bz2
$ tar -xjvf linux-version.tar.bz2
```

```
$ cd linux-version
$ cp /boot/config-version .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-3.*
$ 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 ハードウェアドライバとファームウェア

The hardware driver is the code running on the main CPUs of the target system. Most hardware drivers are available as free software now and are included in the normal Debian kernel packages in the main area.

- **GPU** ドライバー
 - Intel GPU ドライバー (main)
 - AMD/ATI GPU ドライバー (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は non-free。)
- **ソフトモデム** ドライバー
 - `martian-modem` や `sl-modem-dkms` パッケージ (non-free)

The firmware is the code or data loaded on the device attach to the target system (e.g., CPU [microcode](#), rendering code running on GPU, or [FPGA](#) / [CPLD](#) data, ...). Some firmware packages are available as free software but many firmware packages are not available as free software since they contain sourceless binary data. Installing these firmware data is essential for the device to function as expected.

- The firmware data packages containing data loaded to the volatile memory on the target device.
 - `firmware-linux-free` (main)
 - `firmware-linux-nonfree` (non-free)
 - `firmware-linux-*` (non-free)
 - `*-firmware` (non-free)
 - `intel-microcode` (non-free)
-

- amd64-microcode (non-free)
- The firmware update program packages which update data on the non-volatile memory on the target device.
 - fwupd (main): Firmware update daemon which downloads firmware data from [Linux Vendor Firmware Service](#).
 - gnome-firmware (main): GTK front end for fwupd
 - plasma-discover-backend-fwupd (main): Qt front end for fwupd

non-free や contrib パッケージは Debian システムの一部でないことに注意して下さい。non-free や contrib エリアへのアクセスの有効化や無効化は項2.1.4 に説明されています。項2.1.5 に記載されているように non-free や contrib パッケージを使用に付帯するマイナスを認識すべきです。

Please also note that the firmware data downloaded by fwupd from [Linux Vendor Firmware Service](#) and loaded to the running Linux kernel may be non-free.

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加能となります。

ティップ

<http://wiki.debian.org/SystemVirtualization> を参照下さい。

9.11.1 Virtualization and emulation tools

There are several [virtualization](#) and emulation tool platforms.

- Complete [hardware emulation](#) packages such as ones installed by the [games-emulator](#) metapackage
- Mostly CPU level emulation with some I/O device emulations such as [QEMU](#)
- Mostly CPU level virtualization with some I/O device emulations such as [Kernel-based Virtual Machine \(KVM\)](#)
- OS level container virtualization with the kernel level support such as [LXC \(Linux Containers\)](#), [Docker](#), ...
- OS level filesystem access virtualization with the system library call override on the file path such as [chroot](#)
- OS level filesystem access virtualization with the system library call override on the file ownership such as [fakeroot](#)
- OS API emulation such as [Wine](#)
- Interpreter level virtualization with its executable selection and run-time library overrides such as [virtualenv](#) and [venv](#) for Python

The container virtualization uses 項4.7.4 and it is the backend technology of 項7.6.

Here are some packages to help you to setup the virtualized system.

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。

パッケージ	ポプコン	サイズ	説明
schroot	V:7, I:9	2708	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:1, I:4	271	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:5, I:62	298	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)
cdebootstrap	V:0, I:2	116	Debian システムのブートストラップ (C で書かれている)
virt-manager	V:10, I:43	2298	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:46, I:64	1130	libvirt ライブラリー用のプログラム
games-emulator	I:0	26	games-emulator : Debian's emulators for games
bochs	V:0, I:1	7194	Bochs : IA-32 PC エミュレーター
qemu	I:31	96	QEMU : 高速で汎用のプロセッサエミュレーター
qemu-system	I:21	97	QEMU : フルシステムエミュレーションのバイナリ
qemu-user	V:0, I:11	89683	QEMU : ユーザーモードエミュレーションのバイナリ
qemu-utils	V:11, I:108	6077	QEMU : ユーティリティ
qemu-kvm	V:8, I:55	107	KVM : ハードウェア補助仮想化 を利用する x86 ハードウェア上のフル仮想化
virtualbox	V:11, I:14	107018	VirtualBox : i386 と amd64 上での x86 仮想化解決策
xen-tools	V:0, I:4	727	Debian XEN 仮想サーバーの管理ツール
wine	V:16, I:76	191	Wine : Windows API の実装 (標準スイート)
dosbox	V:2, I:18	2702	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
dosemu	V:0, I:2	4891	DOSEMU : Linux 用 DOS エミュレーター
lxc	V:11, I:14	18971	Linux コンテナ - ユーザースペースツール
python3-venv	V:2, I:50	6	venv for creating virtual python environments (system library)
python3-virtualenv	V:9, I:64	410	virtualenv for creating isolated virtual python environments
python3-pipx	I:0	NOT_FOUND	pipx for installing python applications in isolated environments

Table 9.27: 仮想化ツールのリスト

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9.11.3参照下さい) としてマウント。
- 必要なシステムデータ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[ハードウェア補助仮想化](#)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[ハードウェア補助仮想化](#)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。

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](#) がサポートする次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](#)、`vfat` (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9.7.3参照下さい)。`"image.img"` の最初のパーティションは次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

If you wish to try a new Debian environment from a terminal console, I recommend you to use [chroot](#). This enables you to run console applications of Debian `unstable` and `testing` without usual risks associated and without rebooting. `chroot(8)` is the most basic way.

**注意**

Examples below assumes both parent system and chroot system share the same amd64 CPU architecture.

Although you can manually create a `chroot(8)` environment using `debootstrap(1)`. But this requires non-trivial efforts.

The [sbuild](#) package to build Debian packages from source uses the chroot environment managed by the [schroot](#) package. It comes with helper script `sbuild-createchroot(1)`. Let's learn how it works by running it under `script(1)` as follows.

```
$ sudo mkdir -p /srv/chroot
$ sudo sbuild-createchroot -v --include=eatmydata,ccache unstable /srv/chroot/unstable- ↵
amd64-sbuild http://deb.debian.org/debian
```

You see how `debootstrap(8)` populates system data for `unstable` environment under `"/srv/chroot/unstable-amd64-sbuild"` for a minimal build system.

You can login to this environment using `schroot(1)`.

```
$ sudo schroot -v -c chroot:unstable-amd64-sbuild
```

You see how a system shell running under `unstable` environment is created.

注意

The `"/usr/sbin/policy-rc.d"` file which always exits with 101 prevents daemon programs to be started automatically on the Debian system. See `"/usr/share/doc/sysv-rc/README.policy-rc.d.gz"`.

注意

Some programs under chroot may require access to more files from the parent system to function than `sbuild-createchroot` provides as above. For example, `"/sys"`, `"/etc/passwd"`, `"/etc/group"`, `"/var/run/utmp"`, `"/var/log/wtmp"`, etc. may need to be bind-mounted or copied.

ティップ

The `sbuild` package helps to construct a chroot system and builds a package inside the chroot using `schroot` as its backend. It is an ideal system to check build-dependencies. See more on [sbuild at Debian wiki](#) and [sbuild configuration example in "Guide for Debian Maintainers"](#).

9.11.5 複数のデスクトップシステム

If you wish to try a new GUI Desktop environment of any OS, I recommend you to use [QEMU](#), [KVM](#), or [VirtualBox](#) on a Debian `stable` system to run multiple desktop systems safely using [virtualization](#). These enable you to run any desktop applications including ones of Debian `unstable` and `testing` without usual risks associated with them and without rebooting. The configuration of these tools are relatively straight forward.

Since pure [QEMU](#) is very slow, it is recommended to accelerate it with [KVM](#) when the host system supports it.

[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](#) を使って次のように作成できます。

```
$ wget http://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
...
```

更なるティップに関しては [Debian wiki: QEMU](#) を参照下さい。

[VirtualBox](#) はQt の GUI ツールとして提供され非常に直感的に理解できます。その GUI とコマンドラインツールは [VirtualBox User Manual](#) と [VirtualBox User Manual \(PDF\)](#) で説明されています。

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。

Chapter 10

データ管理

バイナリーとテキストのデータを 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。

データのセキュリティ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

こういったことは次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:914, I:999	3152	.tar	tar(1)	標準アーカイバー (デファクト標準)
cpio	V:489, I:998	1144	.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:164, I:673	97	.ar	ar(1)	静的ライブラリー生成用のアーカイバー
fastjar	V:2, I:25	183	.jar	fastjar(1)	Java 用のアーカイバー (zip 類似)
pax	V:12, I:24	170	.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:891, I:999	242	.gz	gzip(1), zcat(1), ...	GNU LZ77 圧縮ユーティリティ (デファクト標準)
bzip2	V:147, I:973	122	.bz2	bzip2(1), bzip2cat(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
lzma	V:2, I:27	149	.lzma	lzma(1)	LZMA compression utility with higher compression ratio than gzip(1) (deprecated)
xz-utils	V:453, I:980	612	.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 (gzip より遅いが bzip2 より早い、 LZMA 圧縮ユーティリティの代替)
zstd	V:4, I:24	1902	.zstd	zstd(1), zstdcat(1), ...	Zstandard fast lossless compression utility
p7zip	V:79, I:454	987	.7z	7zr(1), p7zip(1)	高い圧縮比をもつ 7-Zip 圧縮ユーティリティ (LZMA 圧縮)
p7zip-full	V:102, I:469	4664	.7z	7z(1), 7za(1)	高い圧縮比をもつ 7-Zip 圧縮ユーティリティ (LZMA 圧縮、他)
lzop	V:10, I:85	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
zip	V:49, I:427	623	.zip	zip(1)	InfoZIP : DOS アーカイブと圧縮ツール
unzip	V:132, I:792	385	.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次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が。 `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のために `“zip -rX”` として使ってください。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ください。

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:901, I:999	17478	GNU cp	ファイルやディレクトリーのローカルコピー (<code>“-a”</code> で再帰的実行)
openssh-client	V:809, I:997	4298	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、 <code>“-r”</code> で再帰実行)
openssh-server	V:701, I:844	1567	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:311, I:564	677		単方向リモート同期とバックアップ
unison	V:3, I:16	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(8)` を使った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ッシュ (/) 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、次のようにも出来ます。


```
$ find ./source -xdev -print0 | tar -cvJf archive.tar.xz --null -F -
```

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ッシュ付加」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、次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 xvpf - )
# (cd ./source && tar cf - . ) | ssh user@host.dom '(cd /dest && tar xvpf - )'
```

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”ディレクトリーにコピーできます。

”.”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10.1.4参照下さい) のためや xargs(1) (項9.4.9参照下さい) のためにファイルを選択するのに find(1) が使われます。この操作は find(1) 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次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” はデフォルトでは emacs スタイルの BRE (項1.6.2参照下さい) 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
```

これは次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、デフォルトの代わりに ERE (項1.6.2参照下さい) 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” の慣用的な使い方に注目して下さい。

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

重要なデータアーカイブのための[コンピューターデータ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。著者は通常複数の安全な場所に保管された複数のセキュアな暗号化されたコピーに頼っています。

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ティカルなデータ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。

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次の何れも指します。

- [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

これらは次の何れかで接続できます。

- [USB](#)
- [IEEE 1394 / FireWire](#)
- [PC カード](#)

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。

- `udisks` 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次のようにしてカスタマイズ可能なマウント点として`/media/disk_label` 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(1)` 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](#)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)` コマンドを使いましょう。

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必要があります。ファイルシステム選択のリストを次に示します。

filesystem name	typical usage scenario
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	cross platform sharing of data on the large hard disk like device (<512TiB, supported by WindowsXP, Mac OS X Snow Leopard 10.6.5, and Linux kernel since 5.4 release)
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	最新の 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	sharing of data on the hard disk like device with current Linux systems with read-only snapshots

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](#)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次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)` を使って次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 では通常 ext4 ファイルシステムを使うべきです。

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する詳細は、["Filesystems HOWTO"](#) を参照下さい。

10.1.9 ネットワーク経由でのデータ共有

データ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 か samba-doc パッケージ参照下さい
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" 経由でのファイル共有、 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ソケットレイヤー (SSL) もしくは Transport Layer Security (TLS) を使ったウェブサーバー/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共有のために非常に便利ですが、インセキュア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。

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ファイル: `~/` 中のデータ
- 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ファイル: `/var/` (`/var/cache/` と `/var/run/` と `/var/tmp/` は除外) 中のデータ
- システム設定ファイル: `/etc/` 中のデータ
- ローカルデータ: `/usr/local/` とか `/opt/` 中のデータ
- 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
- 実証済みのデータセット: 事前に実験的復元操作をして確認済み

2. バックアップと復元の方法を知っていること

- セキュアなデータのストレージ: 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: データのミラーリング
- フルブールなプロセス: 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

- データがなくなった際の価値
- バックアップに必要なリソース: 人的、ハードウェア、ソフトウェア、…
- 故障モードとその確率

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です。

データのセキュアなストレージとして、好ましくはファイルシステム破壊に耐えるように異なるディスクや機器上に、少なくとも異なるディスクパーティション上に、データはあるべきです。重要データは上書き事故を防ぐために CD/DVD-R のような 1 回書き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"` で簡単に書き込みできます。)

注意

データをバックアップする際には MTA (項6.2.4参照下さい) 等のアプリケーションデーモンを停止するのモータです。

注意

"/etc/ssh/ssh_host_dsa_key" や"/etc/ssh/ssh_host_rsa_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/passwd.client" 等のアイデンティティ関連のデータファイルのバックアップと修復には特に注意が必要です。これらのデータの一部はシステムに同様な入力をしても再生成できません。

注意

ユーザプロセスで cron ジョブを実行している際には、"/var/spool/cron/crontabs" ディレクトリー中のファイルを回復し cron(8) を再スタートしなければいけません。cron(8) と crontab(1) に関しては項9.4.14を参照下さい。

10.2.1 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。

パッケージ	ポプコン	サイズ	説明
dump	V:1, I:6	352	ext2/ext3/ext4 ファイルシステム用の 4.4 BSD 由来の dump(8) と restore(8)
xfsdump	V:0, I:9	854	GNU/Linux と IRIX 上の XFS ファイルシステム用の xfsdump(8) と xfsrestore(8) を使う dump と restore
backupninja	V:3, I:4	367	軽量で拡張可のメタバックアップシステム
bacula-common	V:10, I:14	2158	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	I:3	183	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:1, I:4	107	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:1	183	Bacula : ネットワークバックアップ、復元および検証 - サーバメタパッケージ
amanda-common	V:1, I:2	10030	Amanda : Advanced Maryland Automatic Network Disk Archiver (ライブラリー)
amanda-client	V:1, I:2	1088	Amanda : Advanced Maryland Automatic Network Disk Archiver (クライアント)
amanda-server	V:0, I:0	1075	Amanda : Advanced Maryland Automatic Network Disk Archiver (サーバー)
backup-manager	V:1, I:1	571	コマンドラインのバックアップツール
backup2l	V:0, I:1	114	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)
backuppc	V:3, I:3	3183	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:9, I:19	1834	(リモート) 増分バックアップ
flexbackup	V:0, I:0	243	(リモート) 増分バックアップ
rdiff-backup	V:6, I:14	733	(リモート) 増分バックアップ
restic	V:1, I:4	22540	(リモート) 増分バックアップ
slbackup	V:0, I:0	151	(リモート) 増分バックアップ

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

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、通常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。
- Regular backups of user data can be realized by a simple script ([項10.2.2](#)).

[項10.1.1](#)や[項10.1.2](#)に記された基本的なツールを使うとカスタムスクリプト経由のシステムバックアップができます。そのようなスクリプトは次を使うと強化できます。

- `restic` パッケージは差分 (遠隔) バックアップを提供します。
- `rdiff-backup` パッケージは (リモートの) 増分バックアップを可能にします。
- `dump` パッケージは全ファイルシステムの効率的かつ増分のバックアップと復旧を補助します。

ティップ

`dump` パッケージに関して学ぶには、`/usr/share/doc/dump/` の中のファイルと ["Is dump really deprecated?"](#) を参照下さい。

10.2.2 Personal backup

For a personal Debian desktop system running `testing` suite, I only need to protect personal and critical data. I reinstall system once a year anyway. Thus I see no reason to backup the whole system or to install a full featured backup utility.

At the same time, it is very valuable to have frequent recent snapshots of personal data and system configuration, and occasional full backups of personal data.

I usually make these snapshots and backups with a simple shell script [bss](#). This script is a short shell which uses standard utilities: `btrfs subvolume snapshot`, `rsync`, `cp`.

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。

10.3 データーセキュリティのインフラ

データーのセキュリティ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。

See [項9.9](#) on [dm-crypt](#) and [ecryptfs](#) which implement automatic data encryption infrastructure via Linux kernel modules.

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。

次のようにすると私のキー `1DD8D791` 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 1DD8D791
```

パッケージ	ポプコン	サイズ	コマンド	説明
gnupg	V:498, I:942	864	gpg(1)	GNU プライバシーガード - OpenPGP 暗号化ト署名ツール
gpgv	V:890, I:999	882	gpgv(1)	GNU プライバシガード - 署名確認ツール
paperkey	V:1, I:13	58	paperkey(1)	OpenPGP の秘密キーから秘密の情報だけを抜粋
cryptsetup	V:41, I:77	407	cryptsetup(8) ...	Utilities for dm-crypt block device encryption supporting LUKS
coreutils	V:901, I:999	17478	md5sum(1)	MD5 メッセージダイジェストを計算やチェック
coreutils	V:901, I:999	17478	sha1sum(1)	SHA1 メッセージダイジェストを計算やチェック
openssl	V:814, I:994	1465	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)

Table 10.6: データーセキュリティインフラツールのリスト

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アップデート

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

”~/ .gnupg/gpg.conf” (もしくは古い場所”~/ .gnupg/options”) 中の良いデフォ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。

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 プライバシーガード](#) コマンドを次に記します。

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-sig 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

10.3.3 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” に次の内容を追加します。

```
macro index S ":toggle pgp_verify_sig\n"  
set pgp_verify_sig=no
```

10.3.4 Vim で GnuPG を使用

The gnupg plugin let you run GnuPG transparently for files with extension ".pgp", ".asc", and ".ppg".¹

```
$ 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す。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の整合性のために暗号学的に署名されます。

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。次のコマンドが著者の目に止まりました。

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次のようにして適用します。

```
$ patch -p0 file < file.patch0  
$ patch -p1 file < file.patch1
```

¹If you use "~/.vimrc" instead of "~/.vim/vimrc", please substitute accordingly.

パッケージ	ポップコン	サイズ	コマンド	説明
patch	V:123, I:721	248	<code>patch(1)</code>	差分ファイルをオリジナルに適用
vim	V:102, I:404	3286	<code>vimdiff(1)</code>	vim で 2 つのファイルを並べて比較
imediff	V:0, I:0	170	<code>imediff(1)</code>	対話型フルスクリーンの 2 方/3 方マージツール
meld	V:14, I:38	3065	<code>meld(1)</code>	ファイルを比較やマージ (GTK)
wiggle	V:0, I:0	174	<code>wiggle(1)</code>	リジェクトされたパッチを適用
diffutils	V:883, I:993	1598	<code>diff(1)</code>	1 行ごとにファイルを比較
diffutils	V:883, I:993	1598	<code>diff3(1)</code>	1 行ごとにファイルを比較やマージ
quilt	V:3, I:32	788	<code>quilt(1)</code>	パッチのシリーズを管理
wdiff	V:8, I:69	644	<code>wdiff(1)</code>	テキストファイル間のワードの相違表示
diffstat	V:14, I:146	81	<code>diffstat(1)</code>	差分ファイルによる変化のヒストグラム作成
patchutils	V:16, I:143	232	<code>combinediff(1)</code>	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:16, I:143	232	<code>dehtmldiff(1)</code>	HTML ページから差分ファイルを抽出
patchutils	V:16, I:143	232	<code>filterdiff(1)</code>	差分ファイルから差分ファイルを抽出や削除
patchutils	V:16, I:143	232	<code>fixcvsdiff(1)</code>	CVS により作成された <code>patch(1)</code> が誤解する差分ファイルを修正
patchutils	V:16, I:143	232	<code>flipdiff(1)</code>	古い 2 つのパッチを交換
patchutils	V:16, I:143	232	<code>grepdiff(1)</code>	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:16, I:143	232	<code>interdiff(1)</code>	2 つのユニファイド差分ファイル間の違いを表示
patchutils	V:16, I:143	232	<code>lsdiff(1)</code>	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:16, I:143	232	<code>recountdiff(1)</code>	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:16, I:143	232	<code>rediff(1)</code>	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:16, I:143	232	<code>splitdiff(1)</code>	増分パッチの分離
patchutils	V:16, I:143	232	<code>unwrapdiff(1)</code>	ワードラップされたパッチを復元
dirdiff	V:0, I:2	166	<code>dirdiff(1)</code>	ディレクトリツリー間で相違点の表示と変更のマージ
docdiff	V:0, I:0	555	<code>docdiff(1)</code>	2 つのファイルをワード毎/文字毎に比較
makepatch	V:0, I:0	102	<code>makepatch(1)</code>	拡張パッチファイルの生成
makepatch	V:0, I:0	102	<code>applypatch(1)</code>	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

10.4.3 Interactive merge

If you have two versions of a source code, you can perform 2-way merge interactively using `imediff(1)` by the following.

```
$ imediff -o file.merged file.old file.new
```

If you have three versions of a source code, you can perform 3-way merge interactively using `imediff(1)` by the following.

```
$ imediff -o file.merged file.yours file.base file.theirs
```

10.5 Git

Git is the tool of choice these days for the [version control system \(VCS\)](#) since Git can do everything for both local and remote source code management.

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。

Here are some Git related packages.

パッケージ	ポプコン	サイズ	コマンド	説明
git	V:312, I:494	36515	git(7)	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:5, I:41	1749	gitk(1)	GUI による履歴付き Git レポジトリブラウザ
git-gui	V:1, I:23	2343	git-gui(1)	Git 用の GUI (履歴無し)
git-email	V:0, I:11	993	git-send-email(1)	Git からパッチの集合の email として送信
git-buildpackage	V:1, I:11	4187	git-buildpackage(1)	Git を使って Debian パッケージ化を自動化
dgit	V:0, I:1	496	dgit(1)	git interoperability with the Debian archive
imediff	V:0, I:0	170	git-ime(1)	interactive git commit split helper tool
stgit	V:0, I:0	603	stg(1)	Git 上の quilt (Python)
git-doc	I:14	12063	N/A	正式 Git 文書
gitmagic	I:1	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

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次のようにします。

```
$ git config --global user.name "Name Surname"
$ git config --global user.email yourname@example.com
```

You may also customize the Git default behavior by the following.

```
$ git config --global init.defaultBranch main
$ git config --global pull.rebase true
$ git config --global push.default current
```

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次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次のようにするとチェックできます。

```
$ git config --global --list
```

10.5.2 Basic Git commands

Git operation involves several data.

- The working tree which holds user facing files and you make changes to them.
 - The changes to be recorded must be explicitly selected and staged to the index. This is `git add` and `git rm` commands
- The index which holds staged files.
 - Staged files will be committed to the local repository upon the subsequent request. This is `git commit` command.
- The local repository which holds committed files.
 - Git records the linked history of the committed data and organizes them as branches in the repository.
 - The local repository can send data to the remote repository by `git push` command.
 - The local repository can receive data from the remote repository by `git fetch` and `git pull` commands.
 - * The `git pull` command performs `git merge` or `git rebase` command after `git fetch` command.
 - * Here, `git merge` combines two separate branches of history at the end to a point. (This is default of `git pull` without customization and may be good for upstream people who publish branch to many people.)
 - * Here, `git rebase` creates one single branch of sequential history of the remote branch one followed by the local branch one. (This is `pull.rebase true` customization case and may be good for rest of us.
- The remote repository which holds committed files.
 - The communication to the remote repository uses secure communication protocols such as SSH or HTTPS.

The working tree is files outside of the `.git/` directory. Files inside of the `.git/` directory hold the index, the local repository data, and some git configuration text files.

Here is an overview of main Git commands.

10.5.3 Git tips

Here are some Git tips.



警告

たとえば `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。




注意

If a local branch which has been pushed to remote repository is rebased or squashed, pushing this branch has risks and requires `--force` option. This is usually not an acceptable for main branch but may be acceptable for a topic branch before merging to main branch.

Git command	機能
git init	(ローカル) レポジトリを作成
git clone URL	clone the remote repository to a local repository with the working tree
git pull origin main	update the local <code>main</code> branch by the remote repository <code>origin</code>
git add .	add file(s) in the working tree to the index for pre-existing files in index only
git add -A .	add file(s) in the working tree to the index for all files including removals
git rm filename	remove file(s) from the working tree and the index
git commit	commit staged changes in the index to the local repository
git commit -a	add all changes in the working tree to the index and commit them to the local repository (add + commit)
git push -u origin branch_name	update the remote repository <code>origin</code> by the local <code>branch_name</code> branch (initial invocation)
git push origin branch_name	update the remote repository <code>origin</code> by the local <code>branch_name</code> branch (subsequent invocation)
git diff treeish1 treeish2	show difference between <i>treeish1</i> commit and <i>treeish2</i> commit
gitk	GUI display of VCS repository branch history tree

Table 10.12: Main Git commands



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 年初以来推奨されません。

ティップ

If there is a executable file `git-foo` in the path specified by `$PATH`, entering "`git foo`" without hyphen to the command line invokes this `git-foo`. This is a feature of the `git` command.

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)
- [Everyday GIT With 20 Commands Or So](#) (/usr/share/doc/git-doc/giteveryday.html)
- [Git マジック](#) (/usr/share/doc/gitmagic/html/index.html)

10.5.5 Other version control systems

The [version control systems \(VCS\)](#) is sometimes known as the revision control system (RCS), or the software configuration management (SCM).

Here is a summary of the notable other non-Git VCS on the Debian system.

Git command line	機能
<code>gitk --all</code>	see complete Git history and operate on them such as resetting HEAD to another commit, cheery-picking patches, creating tags and branches ...
<code>git stash</code>	get the clean working tree without loosing data
<code>git remote -v</code>	check settings for remote
<code>git branch -vv</code>	check settings for branch
<code>git status</code>	show working tree status
<code>git config -l</code>	list git settings
<code>git reset --hard HEAD; git clean -x -d -f</code>	revert all working tree changes and clean them up completely
<code>git rm --cached filename</code>	revert staged index changed by <code>git add filename</code>
<code>git reflog</code>	get reference log (useful for recovering commits from the removed branch)
<code>git branch new_branch_name HEAD@{6}</code>	create a new branch from reflog information
<code>git remote add new_remote URL</code>	add a <code>new_remote</code> remote repository pointed by URL
<code>git remote rename origin upstream</code>	rename the remote repository name from <code>origin</code> to <code>upstream</code>
<code>git branch -u upstream/branch_name</code>	set the remote tracking to the remote repository <code>upstream</code> and its branch name <code>branch_name</code> .
<code>git remote set-url origin https://foo/bar.git</code>	change URL of <code>origin</code>
<code>git remote set-url --push upstream DISABLED</code>	disable push to <code>upstream</code> (Edit <code>.git/config</code> to re-enable)
<code>git checkout -b topic_branch ; git push -u topic_branch origin</code>	make a new <code>topic_branch</code> and push it to <code>origin</code>
<code>git branch -m oldname newname</code>	rename local branch name
<code>git push -d origin branch_to_be_removed</code>	remove remote branch (new method)
<code>git push origin :branch_to_be_removed</code>	remove remote branch (old method)
<code>git checkout --orphan unconnected</code>	create a new <code>unconnected</code> branch
<code>git rebase -i origin/main</code>	reorder/drop/squish commits from <code>origin/main</code> to clean branch history
<code>git reset HEAD^; git commit --amend</code>	squash last 2 commits into one
<code>git checkout topic_branch ; git merge --squash topic_branch</code>	squash entire <code>topic_branch</code> into a commit
<code>git ime</code>	split the last commit into a series of file-by-file smaller commits etc. (<code>imediff</code> package required)
<code>git repack -a -d; git prune</code>	repack the local repository into single pack (this may limit chance of lost data recovery from erased branch etc.)

Table 10.13: Git tips

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:6, I:44	1053	Mercurial	分散型	Python と一部 C で書かれた DVCS
darcs	V:0, I:7	23159	Darcs	分散型	パッチに関して賢い計算をする DVCS (遅い)
bazaar	V:1, I:15	28	Bazaar	分散型	DVCS influenced by tla written in Python (historic)
tla	V:0, I:2	1011	GNU arch	分散型	DVCS mainly by Tom Lord (historic)
subversion	V:18, I:104	4873	Subversion	リモート	"CVS done right", newer standard remote VCS (historic)
cvs	V:4, I:38	4620	CVS	リモート	previous standard remote VCS (historic)
tkcvs	V:0, I:1	1498	CVS, ...	リモート	VCS (CVS, Subversion, RCS) レポジトリツリーの GUI 表示
rcs	V:3, I:19	562	RCS	ローカル	" Unix SCCS done right" (historic)
cssc	V:0, I:2	2044	CSSC	ローカル	clone of the Unix SCCS (historic)

Table 10.14: List of other version control system tools

Chapter 11

データ変換

Debian システム上のデータ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 テキストデータ変換ツール

テキストデータ変換のための次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:946, I:999	12772	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:3, I:25	603	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:1, I:52	134	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, I:11	358	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, I:0	518	文字セット	文字セット翻訳ソフト
unaccent	V:0, I:0	29	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
tofromdos	V:1, I:23	55	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, I:0	298	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。

次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 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
Some encodings are only supported for the data conversion and are not used as locale values (項[8.1](#)).

[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覚えておいて下さい。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”を使いましょう: 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

次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。

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

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) と frommac(1) と tomac(1) は非常に便利です。recode(1) もまた役に立ちます。

注意
python-moinmoin パッケージ用の wiki のデーター等の Debian システム上の一部データーは、MSDOS スタイルの CR-LF を行末コードとして用います。あくまで上記は一般則と言うだけです。

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。

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 を使って次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 を使って次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 を使って次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 を使って次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

以下はウェブページを読みテキストファイルに落とします。ウェ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ブページに適用するときに非常に有用です。

```
$ w3m -dump http://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:25, I:252	2367	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:2, I:29	274	html → text	先進的 HTML からテキストへの変換ソフト (ISO 8859-1)
lynx	V:11, I:97	1948	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:5, I:27	1767	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:5, I:38	2250	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:1, I:14	5418	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:22, I:125	675	MSWord → text, TeX 変換	MSWord ファイルをプレーンテキストか TeX に変換
antiword	V:1, I:10	589	MSWord → text, ps 変換	MSWord ファイルをプレーンテキストか ps に変換
pstotext	V:0, I:2	126	ps/pdf → text	PostScript と PDF ファイルからテキストを抽出
unhtml	V:0, I:0	43	html → text	HTML ファイルからマークアップタグを削除
odt2txt	V:1, I:7	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抽出ツールのリスト

11.1.8 プレーンテキストデータをハイライトとフォーマット

次のようにしてプレーンテキストデータをハイライトとフォーマット出来ます。

11.2 XML データ

[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。

パッケージ	ポプコン	サイズ	キーワード	説明
vim-runtime	V:18, I:439	32134	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, I:0	1193	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, I:0	622	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, I:7	1992	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:1, I:10	1083	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, I:3	208	text → color	汎用着色化ソフト (Python)
python-docutils	V:9, I:112	1771	text → any	ReStructured テキスト文書の XML へのフォーマット化ソフト (Python)
pandoc	V:8, I:48	151714	text → any	汎用マークアップコンバーター (Haskel)
markdown	V:0, I:8	57	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)
txt2html	V:0, I:3	259	text → html	テキストから HTML への変換ソフト (Perl)
asciidoc	I:12	81	text → any	AsciiDoc テキスト文書の XML/HTML へのフォーマット化ソフト (Python)

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

- “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 ファイルは次のような標準的な 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次のようにマークアップされます。

```
name attribute="value"content/name
```

空の XML 要素は次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次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 は次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

When SGML style user defined entities, e.g. "&some-tag:", are used, the first definition wins over others. The entity definition is expressed in `!ENTITY some-tag "entity value"`.

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。

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格言語 \(DSSSL\)](#) 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
docbook-xml	V:14, I:264	2133	xml	DocBook 用 XML ドキュメントタイプ定義 (DTD)
docbook-xsl	V:12, I:154	14870	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:14, I:100	160	xslt	XSLT コマンドラインプロセスソフト (XML → XML, HTML, plain text, 他)
xmlto	V:1, I:22	130	xml/xslt	XSLT を用いて XML から全てへの変換ソフト
fop	V:1, I:20	291	xml/xsl-fo	Docbook XML ファイルを PDF に変換
dblatex	V:2, I:15	4643	xml/xslt	XSLT を使って Docbook ファイルを DVI, PostScript, PDF 文書へ変換
dbtoepub	V:0, I:0	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

パッケージ	ポプコン	サイズ	キーワード	説明
openjade	V:2, I:36	1019	dsssl	ISO/IEC 10179: 1996 標準 DSSSL プロセッサ (最新版)
docbook-dsssl	V:1, I:21	2604	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, I:15	281	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
sgml2x	V:0, I:0	90	SGML/dsssl	DSSSL スタイルシートを使う SGML や XML からの変換ソフト

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, I:2	138	manpage → html	manpage から HTML への変換ソフト (CGI サポート)
doclifter	V:0, I:0	451	troff → xml	troff から DocBook XML への変換ソフト
texi2html	V:0, I:7	1833	texi → html	Texinfo から HTML への変換ソフト
info2www	V:1, I:3	76	info → html	GNU info から HTML への変換ソフト (CGI サポート)
wv	V:0, I:6	717	MSWord → any	Microsoft Word から HTML や LaTeX 等への文書変換ソフト
unrtf	V:0, I:4	148	rtf → html	RTF から HTML 等への文書変換ソフト
ooo2dbk	V:0, I:0	217	sxw → xml	OpenOffice.org SXW 文書から DocBook XML への変換ソフト
wp2x	V:0, I:0	202	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 The XML data lint

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。

Syntax of XML files and goodness of URLs found in them may be checked.

パッケージ	ポップコン	サイ ズ	機能	説明
libxml2-utils	V:21, I:234	181	xml ↔ html ↔ xhtml	xmlLint(1) (シンタクスチェック、リフォーマット、整形、他) を含むコマンドライン XML ツール
tidy	V:1, I:13	84	xml ↔ html ↔ xhtml	HTML シンタクスチェックソフトとリフォーマットソフト
weblint-perl	V:0, I:1	32	lint	HTML 用のシンタクス最小限の文体チェックソフト
linklint	V:0, I:0	344	link check	高速リンクチェックソフトとウェ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	ポップコン	サイ ズ	キーワード	説明
texlive	V:3, I:48	72	(La)TeX	組版、校正、印刷のための TeX システム
groff	V:3, I:59	11870	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、ほとんどの一般的なタスクに十分な [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 - TEX Live 2007](#) (`" /usr/share/doc/texlive-doc-base/english/texlive-en/live.html"`) (`texlive-doc-base` パッケージ)
- [A Simple Guide to Latex/Lyx](#)
- [Word Processing Using LaTeX](#)
- [Local User Guide to teTeX/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/95N"` 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

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

次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, I:12	191	SGML → manpage	DocBook SGML から roff man マクロへの変換ソフト
help2man	V:0, I:10	491	text → manpage	--help からの自動マンページ生成ソフト
info2man	V:0, I:0	135	info → manpage	GNU info から POD かマンページへの変換ソフト
txt2man	V:0, I:1	114	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。

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。

パッケージ	ポプコン	サイズ	説明
ghostscript	V:256, I:576	231	GPL 版 Ghostscript PostScript/PDF インタープリタ
ghostscript-x	V:12, I:63	223	GPL 版 Ghostscript PostScript/PDF インタープリタ - X ディスプレーサポート
libpoppler102	V:55, I:89	4274	xpdf PDF ビューワー派生 PDF レンダリングライブラリー
libpoppler-glib8	V:228, I:463	445	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:102, I:621	13090	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

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
```

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 文書进行操作するのに便利です。pdftk パッケージの pdftk(1) も PDF 文書进行操作するのに便利です。

11.4.3 印刷可能データユーティリティー

印刷可能なデータに用いる次のパッケージが著者の目に止まりました。

パッケージ	ポップコン	サイズ	キーワード	説明
poppler-utils	V:245, I:413	689	pdf → ps,text, ...	PDF ユーティリティー: pdftops, pdfinfo, pdfimages, pdftotext, pdffonts
psutils	V:6, I:98	219	ps → ps	PostScript 文書変換ツール
poster	V:0, I:4	58	ps → ps	PostScript ページから大きなポスターを作る
enscript	V:1, I:19	2132	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, I:14	3648	text → ps	全てを PostScript に変換するソフトと綺麗印刷ソフト
pdftk	I:50	28	pdf → pdf	PDF 文書変換ツール: pdftk
html2ps	V:0, I:3	249	html → ps	HTML から PostScript への変換ソフト
gnuhtml2latex	V:0, I:1	27	html → latex	html から latex への変換ソフト
latex2rtf	V:0, I:6	480	latex → rtf	LaTeX から MS Word で読める RTF へと文書変換
ps2eps	V:3, I:62	99	ps → eps	PostScript から EPS (カプセル化済み PostScript) への変換ソフト
e2ps	V:0, I:0	109	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, I:0	119	ps → ps	PostScript ユーティリティー
trueprint	V:0, I:0	146	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, I:4	30	ps → 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, I:0	71	ps → 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のユーティリティーのリスト

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 メールデータ変換

テキストデータ変換のための次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
sharutils	V:4, I:52	1421	メール	shar(1) と unshar(1) と uuencode(1) と uudecode(1)
mpack	V:1, I:17	106	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:1, I:10	110	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, I:4	109	メール	次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: メールデータ変換を補助するパッケージのリスト

ティップ

The [Internet Message Access Protocol](#) version 4 (IMAP4) server may be used to move mails out from proprietary mail systems if the mail client software can be configured to use IMAP4 server too.

11.5.1 メールデータの基本

Mail ([SMTP](#)) data should be limited to series of 7 bit data. So binary data and 8 bit text data are encoded into 7 bit format with the [Multipurpose Internet Mail Extensions \(MIME\)](#) and the selection of the charset (see 表 11.2).

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” が通常メールに使われます。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

If your non-Unix mail data is accessible by a non-Debian client software which can talk to the IMAP4 server, you may be able to move them out by running your own IMAP4 server.

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 グラフィクスデーターツール

印刷可能なデーターに用いる次のパッケージが著者の目に止まりました。

ティップ

`aptitude(8)` の正規表現 `~Gworks-with::image` (項2.2.6参照下さい) を使ってさらなる画像ツールを探しましょう。

`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10.1.7参照下さい。

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2.2.6参照下さい) を使い次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次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:58, I:315	19706	画像 (bitmap)	GNU イメージ操作プログラム
imagemagick	I:373	220	画像 (bitmap)	画像操作プログラム
graphicsmagick	V:2, I:17	5242	画像 (bitmap)	画像操作プログラム (imagemagick のフォーク)
xsane	V:14, I:165	2346	画像 (bitmap)	GTK に基づく SANE (Scanner Access Now Easy) 用の X11 フロントエンド
netpbm	V:29, I:382	5056	画像 (bitmap)	画像変換ツール
icoutils	V:18, I:115	221	png ↔ ico(bitmap)	MS Windows のアイコンやカーソールと PNG フォーマット間の変換 (favicon.ico)
scribus	V:1, I:22	30370	ps/pdf/SVG/...	Scribus DTP エディター
libreoffice-draw	V:150, I:416	14596	画像 (vector)	LibreOffice office スイート - ドロー
inkscape	V:42, I:188	84892	画像 (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:4, I:30	3727	画像 (vector)	ダイアグラムエディター (Gtk)
xfig	V:2, I:14	1793	画像 (vector)	X11 下でインタラクティブ生成するソフト
pstoedit	V:3, I:87	988	ps/pdf → 画像 (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:8, I:191	113	Windows/画像 (vector)	Windows メタファイル (ベクトル画像データ) 変換ツール
fig2sxd	V:0, I:0	149	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:19	460	画像 → 画像	OCR 用のスキャンしたページの後処理ツール
tesseract-ocr	V:8, I:37	1507	画像 → テキスト	HP の商用 OCR エンジンの基づくフリーの OCR ソフトウェア
tesseract-ocr-eng	V:6, I:37	4032	画像 → テキスト	OCR エンジンデータ: tesseract-ocr の英文用言語ファイル
gocr	V:1, I:11	531	画像 → テキスト	フリー OCR ソフト
ocrad	V:0, I:4	303	画像 → テキスト	フリー OCR ソフト
eog	V:63, I:253	10189	画像 (Exif)	画像ビューアープログラム Eye of GNOME
gthumb	V:5, I:21	5483	画像 (Exif)	画像ビューアー兼ブラウザ (GNOME)
geeqie	V:6, I:19	14643	画像 (Exif)	GTK を用いた画像ビューアー
shotwell	V:16, I:217	6451	画像 (Exif)	デジタル写真オーガナイザー (GNOME)
gtkam	V:0, I:5	1154	画像 (Exif)	デジタルカメラからメディアを回収するアプリケーション (GTK)
gphoto2	V:0, I:12	955	画像 (Exif)	gphoto2 デジタルカメラコマンドライン版クライアント
gwenview	V:27, I:94	10651	画像 (Exif)	画像ビューア (KDE)
kamera	I:93	798	画像 (Exif)	KDE アプリケーション用デジタルカメラサポート
digikam	V:2, I:12	2646	画像 (Exif)	デジタル写真管理アプリケーション
exiv2	V:2, I:40	321	画像 (Exif)	EXIF/IPTC メタデータ操作ツール
exiftran	V:1, I:19	70	画像 (Exif)	デジタルカメラの jpeg 画像を変換
jhead	V:1, I:10	110	画像 (Exif)	Exif に準拠の JPEG (デジタルカメラ写真) ファイルの非画像部を操作
exif	V:1, I:12	339	画像 (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
exiftags	V:0, I:4	292	画像 (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
exifprobe	V:0, I:4	499	画像 (Exif)	デジタル写真からメタデータを読み出す
dcraw	V:2, I:18	535	画像 (Raw) → ppm	生のデジタルカメラ画像のデコード
			画像	

パッケージ	ポプコン	サイ ズ	キーワード	説明
alien	V:2, I:31	163	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	V:0, I:0	421	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:8, I:35	55048	any → EPUB	e-book コンバーターとライブラリーの管理

Table 11.18: その他のデータ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。

Please consider to use version control system tools. See [?].



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プログラムは、システムプログラムとかわち合わないよう
に、`/usr/local` か `/opt` の中にインストールします。

ティップ

"99 ボトルのビールの歌" 作成のコード例はほとんど全てのプログラム言語に関する理解のための非常に好適です。

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、シェルスクリプト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" (<http://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

Many system scripts may be interpreted by any one of [POSIX](#) shells (see [表 1.13](#)).

- The default non-interactive POSIX shell `/bin/sh` is a symlink pointing to `/usr/bin/dash` and used by many system programs.
- The default interactive POSIX shell is `/usr/bin/bash`.

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しょう。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
<code>if ["\$foo" = "\$bar"] ; then ...</code>	<code>if ["\$foo" == "\$bar"] ; then ...</code>
<code>diff -u file.c.orig file.c</code>	<code>diff -u file.c{.orig,}</code>
<code>mkdir /foobar /foobaz</code>	<code>mkdir /foo{bar,baz}</code>
<code>funcname() { ... }</code>	<code>function funcname() { ... }</code>
8 進表記: <code>"\377"</code>	16 進表記: <code>"\xff"</code>

Table 12.1: 典型的 bashisms 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次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。

シェル変数	変数値
<code>\$0</code>	シェルまたはシェルスクリプトの名前
<code>\$1</code>	最初 (1 番目) のシェル引数
<code>\$9</code>	9 番目のシェル引数
<code>\$#</code>	シェル引数の数
<code>"\$*"</code>	<code>"\$1 \$2 \$3 \$4 ..."</code>
<code>"\$@"</code>	<code>"\$1" "\$2" "\$3" "\$4" ...</code>
<code>\$?</code>	最新のコマンドの終了状態
<code>\$\$</code>	このシェルスクリプトの PID
<code>\$_</code>	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
<code>\${var:-string}</code>	<code>"\$var"</code>	<code>"string"</code>
<code>\${var:+string}</code>	<code>"string"</code>	<code>"null"</code>
<code>\${var:=string}</code>	<code>"\$var"</code>	<code>"string"</code> (合わせて <code>"var=string"</code> を実行)
<code>\${var:?string}</code>	<code>"\$var"</code>	<code>"string"</code> を stderr に出力 (エラーとともに <code>exit</code> する)

Table 12.3: シェル変数展開のリスト

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

パラメーター置換形	結果
<code>\${var%suffix}</code>	最短のサフィックスパターンを削除
<code>\${var%%suffix}</code>	最長のサフィックスパターンを削除
<code>\${var#prefix}</code>	最短のプレフィックスパターンを削除
<code>\${var##prefix}</code>	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"`
- 次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 が存在し書き込み可能である
-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

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

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。
- ”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” によって次のループ初めに戻りループを再開します。

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 シェルコマンドライン処理シーケンス

シェルはおおよそ次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 行を次のによってトークンに分割します。
 - 空白: `space tab newline`
 - Metacharacters: `| ; & ()`
- `"..."` 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、シェルの `-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。これはデバッグをするのに非常に便利です。

パッケージ	ポップコン	サイ ズ	説明
dash	V:913, I:993	221	small and fast POSIX-compliant shell for sh
coreutils	V:901, I:999	17478	GNU コアユーティリティ
grep	V:837, I:999	1091	GNU grep, egrep and fgrep
sed	V:834, I:999	883	GNU sed
mawk	V:392, I:997	242	小さく高速な awk
debianutils	V:932, I:999	230	Debian 特有の雑多なユーティリティ
bsdutils	V:708, I:999	394	4.4BSD-Lite 由来の基本ユーティリティ
bsdextrautils	V:80, I:103	384	4.4BSD-Lite 由来の追加のユーティリティ
moreutils	V:12, I:37	237	追加の Unix ユーティリティ

Table 12.7: シェルスクリプト用の小さなユーティティープログラムを含むパッケージのリスト

12.1.6 シェルスクリプトのためのユーティティープログラム

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

Although `moreutils` may not exist outside of Debian, it offers interesting small programs. Most notable one is `sponge(8)` which is quite useful when you wish to overwrite original file.

See [項1.6](#) for examples.

12.2 Scripting in interpreted languages

When you wish to automate a task on Debian, you should script it with an interpreted language first. The guide line for the choice of the interpreted language is:

- Use `dash`, if the task is a simple one which combines CLI programs with a shell program.
- Use `python3`, if the task isn't a simple one and you are writing it from scratch.
- Use `perl`, `tcl`, `ruby`, ... if there is an existing code using one of these languages on Debian which needs to be touched up to do the task.

If the resulting code is too slow, you can rewrite only the critical portion for the execution speed in a compiled language and call it from the interpreted language.

12.2.1 Debugging interpreted language codes

Most interpreters offer basic syntax check and code tracing functionalities.

- `"dash -n script.sh"` - Syntax check of a Shell script

パッケージ	ポップコン	サイズ	文書
dash	V:913, I:993	221	sh : small and fast POSIX-compliant shell for sh
bash	V:796, I:999	6470	sh : "info bash" provided by bash-doc
mawk	V:392, I:997	242	AWK : small and fast awk
gawk	V:359, I:440	2456	AWK : "info gawk" provided by gawk-doc
perl	V:609, I:992	705	Perl : perl (1) and html pages provided by perl-doc and perl-doc-html
libterm-readline-gnu-perl	V:2, I:31	379	Perl extension for the GNU ReadLine/History Library: perlsh (1)
libreply-perl	V:0, I:0	170	Perl の REPL: reply (1)
libdevel-repl-perl	V:0, I:0	237	Perl の REPL: re.pl (1)
python3	V:676, I:901	89	Python : python-doc が提供する python3 (1) と html ページ
tcl	V:28, I:390	22	Tcl : tcl-doc が提供する tcl (3) と詳細なマンページ
tk	V:28, I:382	22	Tk : tk-doc が提供する tk (3) と詳細なマンページ
ruby	V:141, I:308	35	Ruby : ruby (1), erb (1), irb (1), rdoc (1), ri (1)

Table 12.8: List of interpreter related packages

- **"dash -x script.sh"** - シェルスクリプトのトレース
- **"python -m py_compile script.py"** - Syntax check of a Python script
- **"python -mtrace --trace script.py"** - Trace a Python script
- **"perl -I ./libpath -c script.pl"** - Syntax check of a Perl script
- **"perl -d:Trace script.pl"** - Trace a Perl script

For testing code for [dash](#), try [項9.1.4](#) which accommodates [bash](#)-like interactive environment.

For testing code for [perl](#), try REPL environment for Perl which accommodates [Python](#)-like [REPL \(=READ + EVAL + PRINT + LOOP\)](#) environment for [Perl](#).

12.2.2 GUI program with the shell script

The shell script can be improved to create an attractive GUI program. The trick is to use one of so-called dialog programs instead of dull interaction using [echo](#) and [read](#) commands.

パッケージ	ポップコン	サイズ	説明
x11-utils	V:171, I:581	712	xmessage (1): window 中にメッセージや質問を表示 (X)
whiptail	V:116, I:996	71	シェルスクリプトからユーザーフレンドリーなダイアログボックスを表示 (newt)
dialog	V:15, I:118	1222	シェルスクリプトからユーザーフレンドリーなダイアログボックスを表示 (ncurses)
zenity	V:79, I:393	384	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, I:0	75	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:47, I:302	5843	"/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

Here is an example of GUI program to demonstrate how easy it is just with a shell script.

This script uses `zenity` to select a file (default `/etc/motd`) and display it.

GUI launcher for this script can be created following [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
```

This kind of approach to GUI program with the shell script is useful only for simple choice cases. If you are to write any program with complexities, please consider writing it on more capable platform.

12.2.3 究極の短い Perl スクリプト

In order to process data, `sh` needs to spawn sub-process running `cut`, `grep`, `sed`, etc., and is slow. On the other hand, `perl` has internal capabilities to process data, and is fast. So many system maintenance scripts on Debian use `perl`.

Let's think following one-liner AWK script snippet and its equivalents in Perl.

```
awk '($2=="1957") { print $3 }' |
```

これは次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 の次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。
- Perl execution tricks via command line options: `perlrun(1)`
- Perl special variables: `perlvar(1)`

This flexibility is the strength of Perl. At the same time, this allows us to create cryptic and tangled codes. So be careful.

For more crazy Perl scripts, [Perl Golf](#) may be interesting.

パッケージ	ポップコン	サイズ	説明
gcc	V:185, I:598	45	GNU C コンパイラ
libc6-dev	V:241, I:615	14358	GNU C Library: Development Libraries and Header Files
g++	V:57, I:519	15	GNU C++ compiler
libstdc++-10-dev	V:16, I:76	17752	GNU Standard C++ Library v3 (development files)
cpp	V:318, I:755	42	GNU C プリプロセッサ
gettext	V:47, I:302	5843	GNU 国際化ユーティリティ
glade	V:0, I:7	1730	GTK User Interface Builder
valac	V:0, I:7	684	C# like language for the GObject system
flex	V:8, I:91	1279	LEX-compatible fast lexical analyzer generator
bison	V:9, I:101	2858	YACC互換のパーサジェネレータ
susv2	I:0	16	”The Single UNIX Specifications v2” を取得
susv3	I:0	16	”The Single UNIX Specifications v3” を取得
golang	I:21	11	Go programming language compiler
rustc	V:1, I:12	5720	Rust systems programming language
haskell-platform	I:5	12	標準 Haskell ライブラリーとツール
gfortran	V:10, I:97	16	GNU Fortran 95 compiler
fpc	I:3	122	Free Pascal

Table 12.10: コンパイラ関連のパッケージのリスト

12.3 Coding in compiled languages

Here, 項12.3.3 and 項12.3.4 are included to indicate how compiler-like program can be written in C language by compiling higher level description into C language.

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次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”
- 各 C ライブラリー関数名 (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 ”libc” ライブラリーを使って実行プログラム ”run_example” に次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推奨できません。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。

自分で作った“`main()`”と“`yywrap()`”を供給する必要があります。そうでない場合にはあなたの `flex` プログラムは次のようでなければライブラリー無しにコンパイル出来ません。これというのは“`yywrap`”はマクロで、“`%option main`”とすると“`%option noyywrap`”が暗示的に有効になるからです。

```
%option main
%%
.|\\n    ECHO ;
%%
```

上記の代わりにとして、`cc(1)` のコマンドラインの最後に (ちょうど `AT&T-Lex` が“-ll”付きであるように) “-lfl”リンカーオプションを使いコンパイルすることが出来ます。この場合、“`%option`”は必要なくなります。

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。

`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、しばしば `Flex` によって提供される“`yylex()`”を呼び出す“`yparse()`”を呼び出します。

パッケージ	ポップコン	サイ ズ	説明
bison	V:9, I:101	2858	GNU LALR パーサー生成ソフト
byacc	V:0, I:5	160	Berkeley LALR パーサー生成ソフト
btyacc	V:0, I:0	243	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

%%
%%

12.4 静的コード分析ツール

[Lint](#) like tools can help automatic [static code analysis](#).
[Indent](#) like tools can help human code reviews by reformatting source codes consistently.
[Ctags](#) like tools can help human code reviews by generating an index (or tag) file of names found in source codes.

ティップ
Configuring your favorite editor (emacs or vim) to use asynchronous lint engine plugins helps your code writing. These plugins are getting very powerful by taking advantage of [Language Server Protocol](#). Since they are moving fast, using their upstream code instead of Debian package may be a good option.

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、あなたも意味あるバグレポートを作成できるような良い Debian ユーザーになれます。

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次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ください:

- “info gdb”
- “Debugging with GDB” in /usr/share/doc/gdb-doc/html/gdb/index.html
- “[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。

パッケージ	ポップコン	サイズ	説明
vim-ale	I:0	2022	Asynchronous Lint Engine for Vim 8 and NeoVim
vim-syntastic	I:3	1240	Syntax checking hacks for vim
elpa-flycheck	V:0, I:1	792	modern on-the-fly syntax checking for Emacs
elpa-relint	V:0, I:0	133	Emacs Lisp regexp mistake finder
cppcheck-gui	V:0, I:1	5529	静的 C/C++ コード分析ツール (GUI)
shellcheck	V:2, I:10	17977	シェルスクリプトのLintツール
pyflakes3	V:1, I:15	23	passive checker of Python 3 programs
pylint	V:2, I:14	1430	Python コード静的チェックソフト
perl	V:609, I:992	705	静的コードチェックソフト付きのインタプリタ: B::Lint(3perl)
rubocop	V:0, I:0	2390	Ruby 静的コード分析ツール
clang-tidy	V:0, I:5	22	clang-based C++ linter tool
splint	V:0, I:3	2320	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, I:0	181	C/C++ ソースコードを検査してセキュリティの脆弱性を探す ツール
black	V:1, I:3	475	uncompromising Python code formatter
perltidy	V:0, I:5	1674	Perl スクリプトのインデントとリフォーマット
indent	V:0, I:11	425	C language source code formatting program
astyle	V:0, I:4	761	Source code indenter for C, C++, Objective-C, C#, and Java
bcpp	V:0, I:1	111	C(++) beautifier
xmlindent	V:0, I:1	53	XML ストリームリフォーマッタ
global	V:0, I:3	1896	ソースコードの検索と閲覧のツール
exuberant-ctags	V:4, I:34	349	build tag file indexes of source code definitions

Table 12.12: 静的コード分析ツールのリスト

パッケージ	ポップコン	サイズ	文書
gdb	V:12, I:113	9799	gdb-doc が提供する”info gdb”
ddd	V:0, I:10	4092	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

```
$ 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

Since all installed binaries should be stripped on the Debian system by default, most debugging symbols are removed in the normal package. In order to debug Debian packages with gdb(1), *-dbgsym packages need to be installed (e.g. coreutils-dbgsym in the case of coreutils). The source packages generate *-dbgsym packages automatically along with normal binary packages and those debug packages are placed separately in [debian-debug](#) archive. Please refer to [articles on Debian Wiki](#) for more information.

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、次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。

例えば次のように、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 バージョンとかわち合わないようパッケージバージョンを増やします。

```
$ dch -i
```

次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。

The backtrace can be obtained by `gdb(1)` using one of the following approaches:

- Crash-in-GDB approach:
 - 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。
- Crash-first approach:
 - Update the `“/etc/security/limits.conf”` file to include the following:


```
* 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。
 - Load the [core dump](#) file to GDB as `“gdb gdb ./program_binary core”`.
 - GDB プロンプトで”bt” と打ちます。

For infinite loop or frozen keyboard situation, you can force to crash the program by pressing `Ctrl-\` or `Ctrl-C` or executing `“kill -ABRT PID”`. (See [項9.4.12](#))

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”`$MALLOCCHECK_`” 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次のようにします。

```
$ MALLOCCHECK_=2 gdb hello
```

12.5.4 上級 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 (デフォルトは”gdb.txt”)

Table 12.14: 上級 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

次のように `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 /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 Dynamic call tracing tools

There are several dynamic call tracing tools available in Debian. See 項9.4.

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、次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しょう。

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:241, I:615	14358	mtrace(1): glibc 中の malloc デバッグ機能
valgrind	V:5, I:45	80378	メモリーデバッガとプロファイラ
electric-fence	V:0, I:5	70	malloc(3) デバッガ
libdmalloc5	V:0, I:3	393	メモリーアロケーションのデバグ用ライブラリー
duma	V:0, I:0	259	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
leaktracer	V:0, I:2	57	C++ プログラム用のメモリーリーク追跡ソフト

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センブリ

次のように `objdump(1)` を使ってバイナリーコードをディスアセンブルできます。

```
$ 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:174, I:603	1592	make-doc が提供する”info make”
autoconf	V:42, I:280	1846	autoconf-doc が提供する”info autoconf”
automake	V:41, I:278	1830	automake1.10-doc が提供する”info automake”
libtool	V:35, I:264	1198	libtool-doc が提供する”info libtool”
cmake	V:16, I:122	22944	cmake(1) cross-platform, open-source make system
ninja-build	V:4, I:30	347	ninja(1) small build system closest in spirit to Make
meson	V:2, I:18	2761	meson(1) high productivity build system on top of ninja
xutils-dev	V:1, I:11	1466	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、例えば次のように書けます。

```
%.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

”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](#) is a suite of programming tools designed to assist in making source code packages portable to many [Unix-like](#) systems.

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

- [Autoconf](#) is a tool to produce a shell script "configure" from "configure.ac".
 - "configure" is used later to produce "Makefile" from "Makefile.in" template.
- [Automake](#) is a tool to produce "Makefile.in" from "Makefile.am".
- [Libtool](#) is a shell script to address the software portability problem when compiling shared libraries from source code.

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、次のように実行してソフトウェアをアンインストールします。

```
$ ./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、次のようにしてその内容を消すことができます。

```
# find /usr/local -type f -print0 | xargs -0 rm -f
```

If you are not sure where files are installed, you should consider using `checkinstall(8)` from the `checkinstall` package, which provides a clean path for the uninstall. It now supports to create a Debian package with "-D" option.

12.6.3 Meson

The software build system has been evolving:

- [Autotools](#) on the top of [Make](#) has been the de facto standard for the portable build infrastructure since 1990s. This is extremely slow.

- [CMake](#) initially released in 2000 improved speed significantly but was still build on the top of inherently slow [Make](#).
- [Ninja](#) initially released in 2012 is meant to replace Make for the further improved build speed but is also designed to have its input files generated by a higher-level build system.
- [Meson](#) initially released in 2013 is the new popular and fast higher-level build system which uses [Ninja](#) as its backend.

See documents found at "[The Meson Build system](#)" and "[The Ninja build system](#)".

12.7 ウェブ

基本的な対話式動的ウェブページは次のようにして作られます。

- 質問 (クエリー) はブラウザのユーザーに [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次の符号化されたパラメーター付きの [URL](#) 文字列をブラウザからウェブサーバーに送信します。
 - `"http://www.foo.dom/cgi-bin/program.pl?VAR1=VAL1&VAR2=VAL2&VAR3=VAL3"`
 - `"http://www.foo.dom/cgi-bin/program.py?VAR1=VAL1&VAR2=VAL2&VAR3=VAL3"`
 - `"http://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。
- 環境変数は次のように設定されます: `"QUERY_STRING="VAR1=VAL1 VAR2=VAL2 VAR3=VAL3"`。
- ウェブサーバー上の [CGI](#) プログラム ("`program.*`" のいずれでも) が環境変数 `"$QUERY_STRING"` とともに起動されます。
- CGI プログラムの STDOUT (標準出力) がウェブブラウザに送られ対話式の動的なウェブページとして表示されます。

セキュリティ上、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](#) はこの様な機能とともに提供されます。クライアントでのデータ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。

<http://www.google.com/search?hl=en&ie=UTF-8&q=CGI+tutorial> を URL として直接ブラウザのアドレスに入れ 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。

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](#)

パッケージ	ポプコン	サイズ	キーワード	説明
perl	V:609, I:992	705	AWK → PERL	AWK から PERL へのソースコード変換シフト: a2p(1)
f2c	V:0, I:5	442	FORTTRAN → C	FORTTRAN 77 から C/C++ へのソースコード変換 ソフト: f2c(1)
intel2gas	V:0, I:0	178	intel → 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

- [debuild\(1\)](#), and [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づく点の良い例でした。(最近のインストレーションでは 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 2.85 版)” (2021-09-17 09:11:56 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。

I have updated ”Debian Reference (version 2)” to address new topics (Systemd, Wayland, IMAP, PipeWire, Linux kernel 5.10) and removed outdated topics (SysV init, CVS, Subversion, SSH protocol 1, Linux kernels before 2.5). References to Jessie 8 (2015-2020) release situation or older are mostly removed.

This ”Debian Reference (version 2.85)” (2021-09-17 09:11:56 UTC) covers mostly Bullseye (=stable) and Bookworm (=testing) Debian releases.

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 月 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

Many manual pages and info pages on the Debian system as well as upstream web pages and [Wikipedia](#) documents were used as the primary references to write this document. To the extent Osamu Aoki considered within the [fair use](#), many parts of them, especially command definitions, were used as phrase pieces after careful editorial efforts to fit them into the style and the objective of this document.

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 \(第 2.85 版\)"](#) (2021-09-17 09:11:56 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ーによっても更新されています。

["Debian リファレンス \(第 1 版\)"](#) は、角田慎一さんがすべて日本語訳しました。

["Debian リファレンス \(第 2 版\)"](#) は、英文原著者の青木修自身がすべてを日本語訳しました。その際に["Debian リファレンス \(第 1 版\)"](#) から内容が比較的変更されていない「第 1 章 GNU/Linux チュートリアル」等では、角田さんの旧訳文を青木が文体や内容を調整した上で一部再利用させて頂きました。

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

The source of the English original document is currently written in [DocBook](#) XML files. This Docbook XML source are converted to HTML, epub, plain text, PostScript, and PDF. (Some formats may be skipped for distribution.)