



LFS307

Linuxシステム 管理

バージョン1.3



バージョン1.3

© Copyright the Linux Foundation 2023.すべての著作権を保有します。

© Copyright the Linux Foundation 2023.すべての著作権を保有します。

トレーニング サービスに関連して The Linux Foundation が提供または開発したトレーニング資料は、著作権およびその他の知的財産権によって保護されています。

ここに組み込まれたオープンソースコードは、他の著作権者がいる場合があり、該当するオープンソースライセンスに従って使用されています。

トレーニング資料は、提供された形式で参加者が個人的に使用するために提供されています。The Linux Foundation の書面による事前の同意なしに、これらをコピー、修正、参加者以外への配布、または他者へのトレーニング提供に使用することはできません。

本書のいかなる部分も、事前の書面による明示的な同意なしに、複製、複写、検索システムへの保存、または転送することはできません。

発行しています：

リナックスファウンデーション

<https://www.linuxfoundation.org>

この資料の内容または使用に関して、いかなる表明または保証も行われず、商品性または特定目的への適合性に関する明示または黙示の保証は、特に否認されます。

本書では、第三者のアプリケーションソフトウェアパッケージを参照することがありますが、これはデモンストレーションを目的としたものであり、これらのソフトウェアアプリケーションを推奨するものではありません。

Linuxは、Linus Torvaldsの登録商標です。その他、本コース教材内の商標は、各所有者の所有物です。

ここにある素材の適切かつ公正な使用について質問がある場合は、<https://trainingsupport.linuxfoundation.org>にアクセスしてください。

機密情報の非開示について

「(a)オープンソースまたはオープンスタンダード・プロジェクト（総称して「オープンプロジェクト」）のコードベースに関連する情報（既存または将来の貢献を含む）、(b)オープンプロジェクトの形成または運営に一般的に関連または関係する情報、(c)オープンプロジェクトに関わる一般ビジネス事項に関する情報、は機密または専有と表示されていても次のいずれにも含まれないものとします：。

このコースには機密情報は含まれておらず、また機密情報を授業中に漏らしてはならない。

目次

1	はじめに	1
1.1	リナックスファウンデーション	2
1.2	Linux Foundationトレーニング	4
1.3	Linux Foundation認定資格	8
1.4	Linux Foundationデジタルバッジ	11
1.5	研究室の練習問題、解決策、リソース	12
1.6	Linuxとオープンソースプロジェクトにおける物事の変化	14
1.7	Eラーニングコースです：LF207	15
1.8	配信内容	16
1.9	研究室	23
2	Linuxファイルシステムのツリーレイアウト	25
2.1	一つの大きなファイルシステム	26
2.2	データディスティネーション	27
2.3	FHS Linux標準ディレクトリツリー	28
2.4	ルートディレクトリ	31
2.5	/bin	32
2.6	/boot	34
2.7	/dev	35
2.8	/エトセトラ	36
2.9	/home	37
2.10	/libおよび/lib64	39
2.11	/media	40
2.12	/mnt	41
2.13	/オプト	42
2.14	/proc	43
2.15	/sys	45
2.16	/root	46
2.17	/sbin	47
2.18	/srv	49
2.19	/tmp	50
2.20	/usr	51
2.21	/var	52
2.22	/run	53
2.23	研究室	54
3	ユーザー環境	57
3.1	環境変数	58
3.2	キーショートカット	62
3.3	コマンドヒストリー	64
3.4	コマンドエイリアス	68
3.5	研究室	69
4	ユーザーアカウント管理	73
4.1	ユーザーアカウント	74
4.2	シェルの起動ファイル	77

4.3	ユーザーアカウントの管理	80
4.4	ロックされたアカウント	82
4.5	パスワード	83
4.6	/etc/shadow	84
4.7	パスワード管理	86
4.8	パスワードエイジング	87
4.9	ルートアカウント	88
4.10	SSH	90
4.11	研究室	97
5	グループ経営	101
5.1	グループ	102
5.2	所属団体	103
5.3	グループ経営	104
5.4	ユーザープライベートグループ	105
5.5	研究室	106
6	ファイルのパーミッションと所有権	109
6.1	ファイルパーミッションとオーナーシップ	110
6.2	ファイルアクセス権	111
6.3	chmod、chown、chgrp	112
6.4	ウマスク	115
6.5	ファイルシステムACL	117
6.6	研究室	118
7	パッケージマネジメントシステム	121
7.1	なぜパッケージを使うのか？	122
7.2	ソフトウェアパッケージの概念	123
7.3	パッケージの種類	124
7.4	利用可能なパッケージマネジメントシステム	125
7.5	パッケージングツールのレベル・バリエーション	126
7.6	パッケージのソース	127
7.7	ソフトウェアパッケージの作成	128
7.8	リビジョンコントロールシステム	129
7.9	利用可能なソースコントロールシステム	130
7.10	Linuxカーネルとgit	131
7.11	研究室	133
8	デバッグ	137
8.1	DPKG (Debian/パッケージ)	138
8.2	パッケージのファイル名とソース	139
8.3	DPKGクエリ	140
8.4	インストール/アップグレード/アンインストール	141
8.5	研究室	142
9	じどうプログラムツール	143
9.1	じどうプログラムツール	144
9.2	APTユーティリティ	145
9.3	クエリ	146
9.4	パッケージのインストール/リムーブ/アップグレード	147
9.5	クリーンアップ	148
9.6	研究室	149
10	回転数	151
10.1	RPM (レッドハットパッケージマネージャー)	152
10.2	パッケージのファイル名	153
10.3	RPMデータベースとヘルパープログラム	154
10.4	クエリ	156
10.5	パッケージの検証	157

10.6	パッケージのインストールと削除	159
10.7	RPMパッケージの更新、アップグレード、フレッシュ化	161
10.8	Linuxカーネルのアップグレード	163
10.9	rpm2archive と rpm2cpio	164
10.10	研究室	165
11	dnfとyum	169
11.1	ダーンフ	170
11.2	うん	171
11.3	クエリ	172
11.4	パッケージのインストール/リムーブ/アップグレード	173
11.5	dnfの追加コマンド	174
11.6	研究室	175
12	ザッパー	179
12.1	ザッパー	180
12.2	クエリ	181
12.3	パッケージのインストール/リムーブ/アップグレード	182
12.4	その他のzypperコマンド	183
12.5	研究室	184
13	GITの紹介	187
13.1	リビジョンコントロール	188
13.2	コードの出どころを知る : DCOとCLA	190
13.3	利用可能な改訂制御システム	192
13.4	グラフィカルなインターフェイス	193
13.5	ドキュメンテーション	195
13.6	研究室	197
14	Gitの使用法: 一例	199
14.1	基本コマンド	200
14.2	簡単な例	203
14.3	コミットメントへのサインオフ	209
14.4	しゅとけん	210
14.5	研究室	212
15	プロセス	215
15.1	プログラムおよびプロセス	216
15.2	プロセスリミット	220
15.3	プロセスの作成	223
15.4	プロセスコントロール	225
15.5	未来へのスタートラインに立つプロセス	228
15.6	プロセスステート	232
15.7	実行モード	233
15.8	デーモン	236
15.9	にこにこ	237
15.10	研究室	239
16	プロセスモニタリング	247
16.1	プロセスモニタリング	248
16.2	トラブルシューティング	249
16.3	ps	251
16.4	プスツリー	253
16.5	トップ	255
16.6	研究室	257
17	メモリの監視、使用状況、スワップの設定	259
17.1	メモリの監視とチューニング	260
17.2	/proc/sys/vm	262
17.3	バイエムスタット	263
17.4	スワップ	264

17.5	アウトオブメモリーキラー (OOM)	265
17.6	研究室	267
18	I/Oモニタリングとチューニング	271
18.1	I/Oモニタリング	272
18.2	アイオスタット	273
18.3	ヨートップ	274
18.4	研究室	275
19	仮想化の概要	279
19.1	仮想化入門	280
19.2	ホストとゲスト	282
19.3	エミュレーション	283
19.4	ハイパーバイザー	285
19.5	リブパート	289
19.6	けいさんきそうち	291
19.7	KVM	294
19.8	クラウドイット	296
19.9	研究室	299
20	コンテナの概要	313
20.1	コンテナ	314
20.2	アプリケーションの仮想化	315
20.3	コンテナと仮想マシンの比較	316
20.4	ドッカー	317
20.5	Dockerコマンド	319
20.6	ポッドマン	320
20.7	研究室	323
21	LinuxのファイルシステムとVFS	331
21.1	ファイルシステムの基礎知識	332
21.2	ファイルシステム概念	333
21.3	仮想ファイルシステム(VFS)	335
21.4	利用可能なファイルシステム	336
21.5	ジャーナリングファイルシステム	338
21.6	特殊なファイルシステム	339
21.7	研究室	340
22	ディスクパーティション	343
22.1	一般的なディスクの種類	344
22.2	ディスクのジオメトリ	345
22.3	パーティショニング	346
22.4	パーティションテーブル	348
22.5	ディスクデバイスの命名	350
22.6	blkid と lsblk	351
22.7	パーティションのサイズアップ	353
22.8	パーティションテーブルのバックアップとリストア	354
22.9	パーティションテーブルエディタ	356
22.10	エフディスク	357
22.11	研究室	358
23	ファイルシステムの機能: 属性、作成、チェック、使用法、マウンティング	363
23.1	エクステンダトリビュート	364
23.2	ファイルシステムの作成と書式設定	365
23.3	ファイルシステムのトラブルシューティング	366
23.4	ファイルシステムのチェックと修復	368
23.5	ファイルシステムの使用方法	369
23.6	ディスク使用量	370
23.7	ファイルシステムのマウント	371
23.8	エヌエフエス	375
23.9	ブート時および/etc/fstab時のマウントについて	376

23.10	オートマウント	378
23.11	ネットワークブロックデバイス	380
23.12	研究室	384
24	Ext4ファイルシステム	393
24.1	ext4 機能	394
24.2	ext4 レイアウトとスーパーブロック、ブロックグループ	395
24.3	ダンプ2フ	398
24.4	tune2fs	399
24.5	研究室	400
25	論理ボリューム管理 (LVM)	405
25.1	論理ボリューム管理 (LVM)	406
25.2	ボリュームとボリュームグループ	407
25.3	論理ボリュームの操作	408
25.4	論理ボリュームのサイズ変更	411
25.5	LVM Snapshots **について	412
25.6	研究室	413
26	カーネルサービスとコンフィギュレーション	415
26.1	カーネルの概要	416
26.2	カーネルブートパラメーター	417
26.3	カーネルコマンドライン	418
26.4	ブートプロセスの不具合	419
26.5	シスセクタ	420
26.6	研究室	421
27	カーネルモジュール	423
27.1	カーネルモジュール	424
27.2	モジュールユーティリティ	426
27.3	モデム情報	428
27.4	モジュール構成	429
27.5	研究室	430
28	デバイスとudev	431
28.1	udevとデバイス管理	432
28.2	デバイスノード	433
28.3	ルール	436
28.4	研究室	439
29	ネットワークアドレス	441
29.1	IPアドレス	442
29.2	IPv4アドレスの種類	443
29.3	IPv6アドレスの種類	445
29.4	IPアドレスクラス	446
29.5	ネットマスク	447
29.6	ホスト名	448
29.7	エヌティーピー	449
29.8	研究室	455
30	ネットワーク機器と構成	457
30.1	ネットワーク機器	458
30.2	アイピー	459
30.3	イフコンフィグ	461
30.4	予測可能なネットワークインターフェイスデバイス名	463
30.5	ネットワーク構成ファイル	465
30.6	ネットワークマネージャー	467
30.7	ルーティング	472
30.8	仮想ネットワークインターフェイス	475
30.9	DNSと名前解決	478
30.10	ネットワークトラブルシューティング	481

30.11	ネットワーク診断.....	482
30.12	研究室.....	486
31	LDAP **について	493
31.1	LDAP認証.....	494
31.2	研究室 **.....	498
32	ファイアウォール	507
32.1	ファイアウォール.....	508
32.2	インターフェイス.....	511
32.3	ファイアウォール.....	513
32.4	ゾーン.....	516
32.5	ソース管理.....	520
32.6	サービス・ポート管理.....	521
32.8	研究所.....	529
33	システム初期化：systemd、SystemV、アップスタート	533
33.1	init プロセス.....	534
33.2	スタートアップの代替品.....	535
33.3	システムデー.....	536
33.4	システムセクター.....	538
33.5	研究室.....	540
34	バックアップとリカバリーの方法	541
34.1	バックアップの基本.....	542
34.2	バックアップとアーカイブ.....	544
34.3	バックアップの方法と戦略.....	546
34.4	ター.....	549
34.5	圧縮：gzip、bzip2、xz、バックアップ.....	552
34.6	ディーディー.....	553
34.7	アールエスアイエヌシー.....	554
34.8	バックアッププログラム**について.....	555
34.9	研究室.....	556
35	Linuxセキュリティモジュール	559
35.1	Linuxセキュリティモジュール.....	560
35.2	セリナックス.....	562
35.3	AppArmor.....	577
35.4	研究室.....	583
36	システムレスキュー	589
36.1	レスキューメディアとトラブルシューティング.....	590
36.2	レスキュー／リカバリーメディアの使用.....	591
36.3	システムレスキューとリカバリー.....	592
36.4	エマージェンシーブートメディア.....	593
36.5	レスキューメディアの利用.....	594
36.6	エマージェンシーモード.....	596
36.7	シングルユーザーモード.....	597
36.8	研究室.....	598
37	クローキングと評価アンケート	601
37.1	評価アンケート.....	601

**ご注意ください**

** これらのセクションは、一部または全部をオプションとして考慮することができる。これらのセクションには、背景となる参考資料、専門的なトピック、または高度なテーマが含まれています。インストラクターは、教室での経験や時間的制約に応じて、これらをカバーするかしないかを選択することができます。

図表の一覧

2.1	/bin ディレクトリです : Ubuntu 18.04.....	33
2.2	Ubuntu 20.04の/bootディレクトリ	34
2.3	デバイスノード.....	35
2.4	/dev ディレクトリ	35
2.5	/home ディレクトリ	38
2.6	/proc ディレクトリ	44
2.7	/proc/[pid]ディレクトリ	44
2.8	proc/interruptsの内容	44
2.9	/sys ディレクトリ	45
2.10	/root ディレクトリ	46
2.11	/sbin ディレクトリです : Ubuntu 18.04.....	48
2.12	/var ディレクトリ	52
2.13	/run ディレクトリ	53
4.1	usermodを使う	81
4.2	チャージ使用.....	87
6.1	chmodの使用.....	113
7.1	RPMとAPT.....	125
8.1	Ubuntuでのソースパッケージの取得	139
10.1	RPM/パッケージのアンインストール	160
11.1	rpm リポジトリ	170
13.1	ギットギ.....	194
14.1	基本的なgitコマンド	201
15.1	かげんにする	220
15.2	ユーザーモードとカーネルモード、システムコール	234
16.1	BSDオプションでpsを使う	251
16.2	ps出力のカスタマイズ	252
16.3	UNIXオプションでpsを使う	252
16.4	トップ使用.....	255
16.5	/proc 内容	256
17.1	/proc/sys/vm	262
17.2	バイムスタット	263
17.3	vmstatの使用	263
17.4	1つのディスクでvmstatを使う	263
18.1	アイオスタット	273
18.2	iostatを使う	273

18.3	iotopの利用について	274
18.4	イートップオプション	274
19.1	エミュレータ	283
19.2	専用ハイパーバイザー	287
19.3	カーネル内のハイパーバイザー	288
19.4	libvirtベースのユーティリティ	290
19.5	virt-managerの起動	300
19.6	virt-managerで仮想マシンを作成する	301
19.7	virt-managerで TinyCoreLinux の iso イメージを選択する。	301
19.8	virt-managerでメモリとCPUSを設定する。	302
19.9	virt-managerでディスクストレージを設定する	302
19.10	virt-managerでVMのインストールを開始する	303
19.11	virt-managerでVMに入力デバイスを追加する。	304
19.12	virt-managerを使ったインストールメディアへの起動	304
19.13	TinyCoreLinuxの最初の画面	305
19.14	TinyCoreLinuxの最初の画面をリサイズしました。	305
19.15	tc-installの実行	306
19.16	VMでディスクを選択する	306
19.17	virt-managerでのインストールを終了する	307
19.18	virt-managerで新しいVMを実行する	307
20.1	コンテナ	314
20.2	アプリケーションの仮想化	315
20.3	Podmanのソケットステータスを確認する	323
20.4	ポッドマンサーチを使う	324
20.5	dockerサービスのステータスを確認する	326
20.6	ドッカーサーチを使う	327
21.1	ハードリンクとソフトリンク	334
22.1	fdiskを使う	345
22.2	MBRディスクパーティションテーブル	348
22.3	GPT ディスクパーティションテーブル	349
22.4	blkidの使用	351
22.5	lsblkの使用	352
23.1	マックフーズ	365
23.2	ファスニング	368
23.3	dfを使用する	369
23.4	デュの使用	370
23.5	据え付ける	372
23.6	現在マウントされているファイルシステム	373
23.7	/etc/fstab の例	377
23.8	automount 例	379
23.9	ネットワークブロック装置	383
24.1	ext[3,4] ファイルシステムレイアウト	395
24.2	ブロックグループ	396
25.1	LVMコンポーネント	407
25.2	論理ボリュームユーティリティ	408
25.3	物理ボリュームユーティリティ	408
25.4	ボリュームグループユーティリティ	410
26.1	シスセクトル	420
27.1	lsmodを使用する	425
27.2	モデム情報	428

28.1	デバイスノード.....	433
28.2	udevルール.....	435
29.1	IPアドレス.....	442
29.2	ネットワークタイムプロトコル.....	449
30.1	ipの使用.....	460
30.2	ifconfig 出力.....	461
30.3	nmtui メイン画面.....	469
30.4	nmtui Edit画面.....	469
30.5	nmtui ワイヤレスコンフィギュレーション.....	470
30.6	エヌエムシーリ.....	471
30.7	routeとip routeを使う.....	472
30.8	DNS.....	480
30.9	ビュー.....	483
30.10	トレースルート.....	484
30.11	マートゥル.....	485
31.1	LDAPクライアント認証処理.....	496
31.2	OpenLDAPアプライアンスのターンキー・サービス.....	498
31.3	Wireshark LDAPパケットトレース.....	504
32.1	デスティネーションNAT.....	526
32.2	ソースNATとマスカレード.....	528
32.3	SNATまたはMASQUERADEの例.....	531
35.1	SELinuxエンフォースメントモード.....	563
35.2	SELinux Permissive Mode (パーミッシブモード).....	563
35.3	セマナージュ.....	575
37.1	コース調査.....	601

テーブルのリスト

2.1	主なディレクトリーその1.....	29
2.2	主なディレクトリーその2.....	30
2.3	usr 以下のディレクトリ.....	51
2.4	var 以下のディレクトリ.....	52
7.1	利用可能なソースコントロールシステム.....	130
10.1	rpm Queryコマンドの例.....	156
13.1	利用可能なソースコントロールシステム.....	192
13.2	Gitで利用可能なグラフィカル・インターフェイス.....	193
16.1	プロセス・負荷監視ツール.....	248
17.1	メモリ監視ユーティリティ.....	260
20.1	ポッドマンと分布状況.....	323
20.2	Dockerとディストリビューション状況.....	325
21.1	特殊なファイルシステム.....	339
30.1	アイピー.....	459
35.1	AppArmor ユーティリティ.....	581

第1章

はじめに



- 1.1 リナックスファウンデーション 2
- 1.2 Linux Foundationトレーニング 4
- 1.3 Linux Foundation認定資格 8
- 1.4 Linux Foundationデジタルバッジ 11
- 1.5 研究室の練習問題、解決策、リソース 12
- 1.6 Linuxとオープンソースプロジェクトにおける物事の変化 14
- 1.7 Eラーニングコースです：LF207 15
- 1.8 配信内容 16
- 1.9 研究室 23

1.1 リナックスファウンデーション

Linux Foundationとは何ですか？

- の成長を促進することを目的とした非営利のコンソーシアムです：
 - リナックス
 - その他多くのオープンソースソフトウェア（OSS）プロジェクトやコミュニケーション
- 提供することで、持続可能なOSSエコシステムの構築を支援する：
 - 財務・知的資源・サービス
 - トレーニング
 - イベント情報
- 元々はLinuxの開発を保護、サポート、改善するために設立され、Linuxの生みの親であるリーナス・トーバルズの活動を後援しています。
- 先進的なテクノロジー企業やデベロッパーが、ニュートラのコラボレーション環境でサポートします。

The Linux Foundation は、開発者がオープンテクノロジープロジェクトをコーディング、管理、拡張するための中立的で信頼できるハブを提供しています。2000年に設立されたThe Linux Foundationは、1,000人以上のメンバーに支えられ、オープンソースソフトウェア、オープンスターダート、オープンデータ、オープンハードウェアに関するコラボレーションのための世界有数の拠点となっています。The Linux Foundation の手法は、ベストプラクティスを活用し、貢献者、ユーザー、ソリューションプロバイダーのニーズに応えることで、オープンなコラボレーションのための持続可能なモデルを構築することに焦点を当てています。

Linux Foundationは、歴史上、世界最大かつ最も普及しているオープンソースソフトウェアプロジェクトであるLinuxを主催しています。また、Linuxの生みの親であるリーナス・トーバルズとリードメンテナーのグレッグ・クロア-ハートマンの本拠地でもあります。Linuxの成功は、オープンソースコミュニティの成長を促進し、オープンソースの商業的有効性を実証し、あらゆる業界や技術スタックのレベルにわたって無数の新しいプロジェクトを鼓舞しています。

その結果、Linux Foundation は今日、Linux をはるかに超えるものをホストしており、事実上すべての産業分野にまたがる、今日の企業に力を与える多くの重要なオープンソース プロジェクトの傘となっています。私たちが注目している技術には、ビッグデータとアナリティクス、ネットワークキング、組み込みシステムとIoT、ウェブツール、クラウドコンピューティング、エッジコンピューティング、自動車、セキュリティ、ブロックチェーンなど、さまざまなものがあります。

Linux Foundation イベント

これは、**Linux Foundation** のイベントのごく一部のリストでしかありません。北米、ヨーロッパ、アジアなど、毎年複数の場所で開催されるものもあります。

- エンベデッド リナックス 会議
- オープンネットワークワーキング & エッジサミット
- KubeCon + CloudNativeCon
- オートモーティブLinuxサミット
- KVMフォーラム
- Linuxストレージファイルシステムとメモリ管理サミット
- Linuxセキュリティサミット
- Linux Kernel Maintainer Summit (Linuxカーネルメインテナーサミット)
- ザ・リナックスファウンデーション・メンバーサミット

世界中の 85,000 人以上のオープンソース技術者やリーダーが、毎年 **Linux Foundation** のイベントに集まり、アイデアを共有し、学び、コラボレーションしています。**Linux Foundation** のイベントは、オープンソースのメンテナ、開発者、アーキテクト、インフラストラクチャ マネージャ、オープンソース プログラム オフィスを率いるシニア技術者、その他の重要なリーダーシップ機能にとって重要なミーティングの場となっています。

これらのイベントは、オープンソースコミュニティの中で迅速に知名度を上げ、次世代のテクノロジーを評価し創造する人々をつながりを持つことで、オープンソース開発作業を進めるのに最適な場所です。知識を共有し、得るためのフォーラムを提供し、組織がソフトウェアのトレンドを早期に把握し、将来の技術投資に役立てることを支援し、雇用者と人材を結びつけ、世界中の影響のあるオープンソース専門家、メディア、アナリストに技術やサービスを紹介することができます。

1.2 Linux Foundation トレーニング

トレーニング会場

Linux Foundationは、いくつかのタイプのトレーニングを提供しています：

- 物理的な教室（多くの場合、オンサイト）
- オンライン・バーチャルクラスルーム
- インターネットを利用した個人向け自習型E-ラーニング
- イベントベース

Linux Foundation のトレーニングは、コミュニティのための、コミュニティによるものであり、開発者コミュニティのリーダーから直接指導を受ける講師とコンテンツを特徴としています。

受講者は、OS や **Linux** ディストリビューションに柔軟に対応でき、技術的に高度なトレーニングを、開発コミュニティの実際のリーダーたちと共に受けることができます。**Linux Foundation** のコースでは、今日のキャリアで成功するために必要な幅広い基礎知識とネットワークが身につきます。オンラインまたは対面でのトレーニングにより、**Linux Foundation** のコースは、オープンソースの管理および開発のエッセンスについて、お客様またはお客様の開発者を常に先取りしておくことができます。

トレーニングの提供

現在提供しているコースは以下の通りです：

- Linuxプログラミング & 開発トレーニング
- エンタープライズIT & Linuxシステムアドミニストレーションコース
- オープンソースコンプライアンスコース

詳しくは、<https://training.linuxfoundation.org> をご覧ください。

Linux Foundationは、**edX** (<https://edx.org>) を通じて、無料のMOOC (Massively Open Online **Courses**) を幅広く提供しています。これらのコースでは、オープンソースに関連する基本的なトピックから、かなり高度なトピックまでカバーしています。

edXのサイトで探すには、"Linux Foundation "で検索してください。

著作権について

- 本コースの内容およびハンドアウトを含むすべての関連資料は、© Copyright the Linux Foundation 2023.すべての著作権を有します。



コピーや頒布はしないでください

本書のいかなる部分も、The Linux Foundation の書面による事前の許可なく、コピー、写真、磁気、電子、その他の記録など、いかなる方法によっても検索システムに保存、送信、複製することを禁じます。

本トレーニングは、ここで提供されるすべての資料を含め、The Linux Foundationからのいかなる保証もなく提供されるものです。The Linux Foundationは、ここに含まれる内容や詳細の使用または誤用から生じる損害や法的措置について、いかなる責任も負いません。

The Linux Foundation の資料が使用、コピー、またはその他の方法で不適切に配布されていると思われる場合は、<https://trainingsupport.linuxfoundation.org> にアクセスしてください。

機密情報について



機密情報の非開示について

「(a)オープンソースまたはオープンスタンダード・プロジェクト（以下、総称して「オープンプロジェクト」）のコードベースに関連する情報（既存または将来の貢献を含む）です；
(b) 一般的にオープンプロジェクトの形成または運営に関連する情報
(c) 一般的なビジネスに関する情報。
オープン・プロジェクトに関わるすべての事柄について。

このコースには機密情報は含まれておらず、また機密情報を授

与えるものではありません。

1.3 Linux Foundation認定資格

Linux Foundation認定試験

- The **Linux Foundation** は、包括的な認定プログラムを提供しています。
- このプログラムの詳細については、<https://training.linuxfoundation.org/certification> で確認できます。この情報には、各試験でカバーされるドメインとコンピテンシーについての詳細な説明が含まれています。
- LFCS (Linux Foundation Certified **Sysadmin**) 試験以外にも、**Linux Foundation**は多くのオープンソースプロジェクトの認定試験を提供しています。リストは常に拡大しているので、[https](https://training.linuxfoundation.org/certification)を参照してください：
[//training.linuxfoundation.org/certification](https://training.linuxfoundation.org/certification) に現在のリストがあります。
- コースの説明、技術的な必要条件、その他のロジスティクスなどの追加情報については、<https://training.linuxfoundation.org> を参照してください。

認定/研修 ファイアウォール

- **Linux Foundation**には、2つの独立したトレーニング部門があります：
 - 認証取得
 - コースの配信
- これらはファイアウォールで分離されています：
 - 第三者機関による**LF認定試験対策講座**の開発・実施を可能にする。
 - **LF**コースで**秘伝のタレ**を使用することを防止する
 - **LF**コースでの**テスト指導**を防止する
- インストラクターは（今日も含めて）、公開されているインフォメーションにすべて誘導されます。

Linux Foundation トレーニング部門のカリキュラム開発・保守部門は、認定試験の開発、管理、採点に直接的な役割を果たすことはありません。

このような自主的なファイアウォールを設けることで、独立した組織や企業が、受験者が資格試験に合格することを目的とした第三者のトレーニング教材を開発できるようにしています。

さらに、成功するために熟知していなければならない秘密の "ヒント"（あるいは一般的な秘密）がないことを保証しています。

また、**Linux Foundation** は、**テストを教えるだけでなく**、**Linux** システム管理の分野で成功するために必要な多くの分野の幅広い知識を受講者に提供する、非常に堅牢なコース群を開発することができます。

準備のためのリソース

- 何もしない前に、<https://training.linuxfoundation.org/download-free-certification-prep-guide> をダウンロードしてください。
- セクションを設けています：
 - ドメインとコンピテンシー
 - 無料トレーニング教材
 - 有償トレーニングリソース
 - 試験を受ける
- また、候補者ハンドブックは、<https://training.linuxfoundation.org/go/candidate> ハンドブックをご覧ください。
- また、試験別の情報はこちら（<https://training.linuxfoundation.org/certification/lfcs>

私たちは、上で引用した文書にある問題のいくつか（すべてではありません！）を議論します。これらの文書はすべて、Linux Foundationのウェブページ（<https://training.linuxfoundation.org/certification>）のリンクから簡単に見つけることができます。

取り上げたトピックは以下の通りです：

- 試験で扱われる素材は何か
- 候補者要件（識別、認証、適格性、アクセシビリティ、機密保持の要件を含む）。
- 試験の申し込みと受験料、返金規定など。
- Linuxディストリビューションの選択
- ハードウェアとソフトウェアの環境が試験に適しているかどうかを確認します。
- 試験の開始と終了の方法
- 試験のインターフェースと形式
- 試験結果、採点・再採点依頼、再受験方針について
- 証明書の発行、検証、有効期限、更新など。
- テクニカルサポートへのアクセス

1.4 Linux Foundationデジタルバッジ

Linux Foundationデジタルバッジ

- デジタルバッジで能力・資格を伝える
- 電子メールの署名、デジタル履歴書、ソーシャルメディアサイト（**LinkedIn**、**Facebook**、**Twitter**など）で使用できます。
- 資格とそれを取得したプロセスを記述した検証済みのメタデータを含む。
- **Linux Foundation**を修了した学生が対象です。
講座・資格
- 詳細は <https://training.linuxfoundation.org/badges/> でご確認ください。

The **Linux Foundation** は、あなたのプロフェッショナルな目標を達成するために必要なツールを提供することを約束します。私たちは、自分の能力や資格を伝えることが困難であることを理解しています。このため、**Credly** と提携し、**Acclaim** プラットフォームを通じて、デジタル版のクレデンシャルを提供しています。このバッジは、電子メールの署名やデジタル履歴書、**LinkedIn**、**Facebook**、**Twitter**などのソーシャルメディアサイトで使用することができます。このデジタル画像には、あなたの資格とそれを取得するために必要なプロセスを説明する検証済みのメタデータが含まれています。

- バッジは、ソーシャルメディア、履歴書、電子メールの署名、ウェブなど、あらゆるデジタルプラットフォームで共有することが可能です。
- すべてのバッジは、バッジをクリックするだけで、誰でも確認することができます。
- バッジは、Linux Foundation の認定試験に合格した方、および Linux Foundation または認定トレーニング パートナーから直接トレーニング コースを購入した方全員に発行されます。
- また、試験や講座の開発チームで貢献された方にもバッジを発行します。

その仕組みについて

1. パートナーであるCredlyのAcclaimプラットフォームサイトでバッジを請求するよう通知するメールが届きます。
2. そのメールに記載されているリンクをクリックします。
3. アクレイムプラットフォームサイトでアカウントを作成し、Eメールを確認します。

5. 共有を開始します。

1.5 研究室の練習問題、解決策、リソース

研究室

- 各セッションの最後に提供されるハンズオンエクササイズ。
- 仮想マシンでもベアメタルでも可
 - 特に指定がない限り
- オプションとしてマークされた一部の練習問題
- 各演習の最後に解答を用意しています。

コースのソリューションとリソースの入手

- このコースにそのような教材がある場合、ラボ演習の解答、提案、その他のリソースは以下からダウンロードできます: <https://training.linuxfoundation.org/cm/LFS307>
- ブラウザをお持ちでない方は、こちらで入手できます :

```
$ wget --user=LFtraining --password=<ask instructor> \
  https://training.linuxfoundation.org/cm/LFS307/LFS307
  V1.3 SOLUTIONS.tar.xz
```

 となり、RESOURCESファイルでも同様になります。

```
$ wget --user=LFtraining --password=<ask instructor> \
  https://training.linuxfoundation.org/cm/LFS307/LFS307 V1.3
  RESOURCES.tar
```
- 正誤表やアップデートされたソリューションなども同サイトに掲載される予定です
- で解凍することができるファイルです :

```
$ tar xvf LFS307 V1.3 SOLUTIONS.tar.xz
$ tar xvf LFS307 V1.3 RESOURCES.tar
```

SOLUTIONSとRESOURCESのディレクトリには、各セクションのs 01、s 10、s 13のようなサブディレクトリがあり、必要なファイルや興味のあるファイルが含まれています。これらのファイルは、今後のセクションで参照することがあります。

コースによっては、アーカイブやビデオなどのバイナリファイルを対象としたRESOURCESファイルが存在しない場合があります。様々なラボのソースtarballのようなバイナリファイルは、必要に応じて指示とともに再ソースされます。

コースにデモビデオが含まれている場合、これらはほとんど講義時間外の補助的な勉強を目的としています。しかし、インストラクションでは、うまくいけばクリーンなデモンストレーションを得るために、そのうちのいくつかを見せることを選択したり、別のLinuxディストリビューションでの実践例を示したりすることもあります。

1.6 Linuxとオープンソースプロジェクトにおける物事の変化

オープンソースソフトウェアは頻繁に変更

- **Linux Foundation** のコースは、**Kubernetes** や **Linux** カーネルなどのプロジェクトのアップストリームバージョンと同期するように常に更新されています。
- 時折、避けられない破損が発生し、非推奨の機能は消滅します。
- 問題を最小限に抑えるよう努力はしていますが、コントロールできない上流の変化に対応しなければならないことも多々あります。
- その代わり、重要な機能が欠落した古くさい素材になってしまいます。
- こうしたソースやターゲットの移り変わりに対応することも、オープンソースソフトウェアを扱う 楽しみのひとつです！

Linux Foundation のコースは常に更新されており、中には、以下のような依存するプロジェクトの新しいバージョンのリリースごとに更新されるものもあります。

Linuxカーネルまたは**Kubernetes**。

例えば、私たちがどんなに最新の情報を得るために努力してきたとしても、Linuxはカーネル機能を含む技術レベルでも、ディストリビューションやインターフェースレベルでも、常に進化しています。

ですから、このクラスがリリースされた時点で、できるだけ最新の情報を提供するように努めましたが、私たちが説明していない変更や新機能があることを念頭に置いてください。それはやむを得ないことです。

その結果、どんなに努力しても、次のような問題が発生する可能性があるのです：

- 私たちがサポートするLinuxディストリビューションでの動作を停止させる。
- **API**が変更されると、特定のカーネルや上流のライブラリや製品を使用した実験実習ができなくなる。
- これまでサポートしてきた非推奨の機能がついに姿を消すことになります。

クラスのすべてが常に機能することを期待する生徒がいます。それを実現するために努力する一方で、私たちは知っています：

- 非現実的な期待である。
- 問題点とこれを解決する方法を見つけ出すことは、授業の積極的な部分であり、最先端のオープンソースプロジェクトに取り組む「冒険」の一部でもあるのです。動く標的は、いつも当てるのが難しい。

このような問題をすべて回避しようとする、教材がすぐに古くなり、変化についていけなくなります。私たちは、講師や受講生がそうした問題に気づき、コースウェアの設計者に知らせて、次のリリースに適切な改善策を盛り込んでくれることを期待しています。

これこそ真のオープンソースのあり方だと思います。

1.7 Eラーニングコースです: LF207

コンパニオンE-ラーニングコース : LF207

- **Linux Foundation**は、自習型のE-ラーニングによる密接な関連コースを提供しています :

LF207: Linuxシステム管理エッセンシャルズ

- このコースは、本コースと同じ内容をカバーし、多くの実験演習を共有しています。
- オンライン上の教材へのアクセスは1年間可能で、すべてのコンテンツとエクササイズをカバーするために必要なだけの時間を取ることができます。
- なお、インストラクターによる授業では、時間の都合上、すべてのテーマに十分な時間を割くことができないため、一部のトピックはオプションとして表記しています。
- 本コースの受講に伴い、**LF207**のサブスクリプションを受け取っている場合があります。そうでない場合は、<https://training.linuxfoundation.org/>、別途ご連絡ください。

1.8 配信内容

ソフトウェア環境

- クラス教材は、複数の環境に対応できるように設計されています
- 現在の主要な3つのLinuxディストリビューションファミリーに焦点を当てます：

- レッドハット / フェドラ
- OpenSUSE / SUSE



– デビアン

ご注意ください

このセクションの資料は、Linuxベースのオペレーティングシステム上でのみ実行することを要求または強く推奨するコースを対象としています。コースによっては、ブラウザとssh（セキュアシェル）ユーティリティがあれば、どんな環境でも実行できるものもあります。しかし、その場合、この資料を読み飛ばすことはできますが、そ

Linux Foundation が作成する教材は、ディストリビューションに柔軟に対応しています。つまり、技術的な説明、ラボ、手順は、ほとんどの最新のディストリビューションで動作するはずであり、特定のベンダーが販売する製品を宣伝することはありません（ただし、特定のシナリオについては言及することがあります）。

実際のところ、私たちの資料のほとんどは、3つの主要なLinuxディストリビューションファミリーを念頭に置いて書かれています：Red Hat / Fedora、Open- SUSE / SUSE、Debianです。生徒が使用するディストリビューションは、この3つのうちの1つ、またはそこから派生した製品であることが多いようです。

どのディストリビューションを選ぶ

- いくつかの考慮すべき要素があります：
 - あなたの雇用主はすでに標準化していますか？
 - もっと勉強したいですか？
 - 認定しますか？
- 複数のディストリビューションで実験することが推奨されます

新しいディストリビューションを選択する際には、いくつかの質問を自分に投げかける必要があります。あるLinuxディストリビューションと他のディストリビューションのどちらかに絞らざるを得ない理由はたくさんありますが、すべてのディストリビューションで経験を積むことをお勧めします。技術的な違いは、主にパッケージ管理システム、ソフトウェアのバージョン、ファイルの位置に関するものであることにすぐに気がつくはずです。これらの違いを把握すれば、Linuxディストリビューションから別のディストリビューションへの切り替えは比較的簡単です。

ツールやユーティリティの中には、特に特殊なレポートや複雑なレポートを作成するために、ベンダーが提供するフロントエンドを備えているものもあります。本文に含まれる手順は、異なるプラットフォームで実行するために修正する必要がある場合があります。

Red Hat / Fedoraファミリー

- **Red Hat Enterprise Linux (RHEL)**の最新リリースに基づく最新の資料です。
- **x86、x86-64、Itanium、PowerPC、IBM System Z**をサポートします。
- **RPM**ベース、**dnf**（または**yum**）を使用してインストールとアップデートを行う。
- リリースサイクルが長く、エンタープライズサーバー環境をターゲットとしている
- **CentOS、Oracle Linux**向けアップストリーム
- **ダウンストリーム for CentOS Stream**

CentOS Stream

CentOS Streamは、無償で利用できるため、デモやラボに利用されています。

Fedoraは、**Red Hat Enterprise Linux、CentOS、Oracle Linux**の基盤を形成するコミュニティ・ディストリビューションです。Fedoraには、Red Hatのエンタープライズ版よりもかなり多くのソフトウェアが含まれています。その理由の一つは、**Fedora**の構築には多様なコミュニティが関わっており、1つの企業だけではないことです。

Fedoraコミュニティは、半年に1回程度、新しいバージョンを作成しています。このため、教材の**Red Hat / Fedora**の部分は、リリースサイクルがずっと長い最新バージョンの**CentOS/CentOS Stream**に標準化することにしました。**CentOS Stream**は、インストールすれば、エンタープライズ環境で最も人気のあるLinuxディストリビューションである**Red Hat Enterprise Linux (RHEL)**にも非常に近くなります。



CentOSとCentOS Stream

- **CentOS**は歴史的に見ても、基本的には**RHEL**のコピーで、アップデート後に多少の時間差があります。
- **CentOS Stream**は**RHEL**より先にアップデートが行われますが、それ以外は**RHEL**にかなり近い状態です。そのため、より新しい機能をいち早く吸収することができます。
- **Red Hat**は2021年末に**CentOS 8**のサポートを終了します。したがって、本コースは現在、**CentOS Stream**ディストリビューションでテストされており、**CentOS 8**や**RHEL 8**との差異は軽微であり、気づくこともないは

OpenSUSEファミリー

- **OpenSUSEの最新リリースに基づく現在の資料は、それ以降のバージョンでも問題なく動作するはずです。**
- RPMベースで、インストールとアップデートに**zypper**を使用します。
- 管理用として**YaST**を利用可能
- **x86およびx86-64**
- **SUSE Linux Enterprise Server (SLES) 向けアップストリーム**



ご注意ください

OpenSUSEは無償で利用できるため、デモやラボに利用されています

。

OpenSUSEと**SUSE Linux Enterprise Server**の関係は、先ほど説明したFedoraと**Red Hat Enterprise Linux**の関係と似ています。ただし、この場合、**SUSE Linux Enterprise Server**の無償版を入手することが困難なため、**OpenSUSE**を**OpenSUSE**ファミリーのリファレンスディストリビューションとして使用することにしました。この2つの製品は非常によく似ており、**OpenSUSE**を取り上げた資料は、通常、**SUSE Linux Enterprise Server**にも問題なく適用できます。

Debianファミリー

- サーバーとデスクトップの両方で共通に使用される
- **DPKG**ベースで、インストールやアップデートにaptやフロントエンドを使用する。
- **Ubuntu**、**Linux Mint**などに対応したアップストリーム
- 現在の資料は、**Ubuntu**の最新リリースに基づくもので、それ以降のバージョンでも問題なく動作するはずです。
- **x86**および**x86-64**



ロングタームリリース (LTS)

ご注意ください

Ubuntuは、**Debian**と同様に無償で利用できるため、デモやラボに利用されていますが、新しい**Linux**ユーザーの裾野が広いのが特徴です。

。

Debianディストリビューションは、**Ubuntu**、**Linux Mint**など、他のいくつかのディストリビューションの上流となっています。Debianは純粋なオープンソースプロジェクトであり、安定性という重要な側面に重点を置いています。また、最大かつ最も完全なソフトウェアリポジトリをユーザーに提供しています。

Ubuntuは、長期的な安定性と使いやすさの間の良い妥協点を提供することを目的としています。**Ubuntu**はパッケージのほとんどをDebianの不安定ブランチから取得しているため、Ubuntuも非常に大きなソフトウェアリポジトリにアクセスすることができます。このような理由から、私たちはラボの演習で使用する**Debian**ベースのディストリビューションのリファレンスとして**Ubuntu**を使用することにしました。

新配信の類似性

- 現在の傾向や分配金の変更により、分配金間の差はある程度小さくなっています。
 - **systemd**、システムのスタートアップとサービス管理
 - **journald**、システムログを管理する
 - **firewalld**、ファイアウォール管理デーモン
 - **IP**、ネットワーク表示・設定ツール



ご注意ください

これらのユーティリティはディストリビューション間で共通なので、講義やラボの情報はほとんどこれらに基づいています。

もし、あなたが選んだディストリビューションやリリースがこれらのコマンドをサポートしていない場合は、それに応じて翻訳してください。

systemdは、**SysVinit**と**Upstart**パッケージの代わりに、最も一般的なディストリビューションで使用されています。サービスおよび

chkconfigコマンドを使用します。

journaldは、ロギングデータの収集と保存を行う **systemd** サービスです。様々なソースから受け取ったロギング情報に基づいて、構造化されたインデックス付きのジャーナルを作成・維持します。ディストリビューションによっては、テキストベースのシステムログが置き換えられるかもしれません。

firewalldは、ネットワーク接続やインターフェイスの信頼レベルを定義するためのネットワーク/ファイアウォールゾーンをサポートし、動的に管理されるファイアウォールを提供します。IPv4、IPv6ファイアウォール設定とイーサネットブリッジのサポートがあります。これは、**iptables**の設定に取って代わるものです。

ip プログラムは **net-tools** パッケージの一部で、**ifconfig** コマンドの代替となるように設計されています。**ip** はコマンドは、ルーティング、ネットワークデバイス、ルーティング情報、トンネルを表示したり操作したりする。

これらのドキュメントは、古いコマンドを **systemd** の対応するコマンドに翻訳するのに役立つかもしれ

ません: [https://fedoraproject.org/wiki/SysVinit to Systemd Cheatsheet](https://fedoraproject.org/wiki/SysVinit_to_Systemd_Cheatsheet)

<https://wiki.debian.org/systemd/CheatSheet>

[https://en.opensuse.org/openSUSE:Cheat-シート 13.1#Services](https://en.opensuse.org/openSUSE:Cheat-シート_13.1#Services)

AWS Free Tier

- **Amazon Web Services (AWS)** は、リモートユーザーがクラウド上でアクセスできる仮想マシン製品 (インスタンス) を幅広く提供しています。
- 特に、**Free Tier** のアカウントレベルを最大 1 年間使用することができ、利用可能なシミュレートされたハードウェアとソフトウェアの選択肢は、**Linux Foundation** のトレーニングコースの演習を実行し、**Linux** を経験するために必要なすべてかもしれません。あるいは、ローカルのハードウェアで作業するための非常に教育的な補足となり、複数の **Linux** ディストリビューションを簡単に学習する機会を提供するかもしれません。<https://training.linuxfoundation.org/cm/prep/aws.pdf> から、**AWS free tier** を使って実験するためのガイドをダウンロードしてください。
- **AWS** はコンソールへのアクセスやグラフィカルインターフェースを提供しないので、これらの機能を必要とするタスクを実行することができないことに注意してください。さらに、カーネルレベルのコースでは、いくつかの特別な課題があります。

1.9 研究室

練習問題1.1 : sudoのためのシステム設定

たった一つのタイプミスやその他の間違いが、重大な（致命的な）損害を引き起こす可能性があるため、絶対に必要な場合を除き、**ルートシェル**を実行することは非常に危険です。

したがって、賢明な方法は、**sudo**メカニズムを使用して、単一のコマンドをスーパーユーザー特権で実行できるように設定することです。**sudo**では、ユーザーは自分のパスワードだけを知っていればよく、**root**のパスワードを知る必要はない。

Ubuntuなどのディストリビューションを使用している場合、コース用に**sudo**を適切に設定するために、このラボを行う必要はないかもしれませんが、ただし、それでも手順を理解しているかどうかは確認しておく必要があります。

お使いのシステムが、お使いのユーザーアカウントに**sudo**を実行させるようにすでに設定されているかどうかを確認するには、次のような簡単なコマンドを実行するだけです：

```
$ sudo ls
```

ユーザーパスワードの入力を求められ、コマンドが実行されるはずです。もし、エラーメッセージが表示された場合は、次の手順を実行してください。

suと入力してrootシェルを起動し、ユーザーパスワードではなく、**root**パスワードを入力する。

最近のすべての **Linux** ディストリビューションでは、**/etc/sudoers.d** サブディレクトリに移動して、通常 **root** が **sudo** アクセスを許可したいユーザの名前を含むファイルを作成する必要があります。しかし、**sudo**は必要に応じてこのディレクトリのすべてのファイルをスキャンするため、この規約は実際には必要ありません。このファイルには、単に

```
student ALL=(ALL) ALL
```

ユーザが学生である場合

古くからある方法(確かに今でも有効)は、**/etc/sudoers** ファイルの最後にそのような行を追加することです。**visudo** プログラムは、編集時に正しいシンタックスを使用するように注意してくれますので、これを使用するのがベストです。

また、ファイルに対して適切なパーミッションを設定する必要があると思いますので、以下のように入力してください：

```
$ sudo chmod 440 /etc/sudoers.d/student
```

(一部のLinuxディストリビューションでは、パーミッションに440の代わりに400が必要な場合があります。)

これらの手順を行った後、**exit**と入力してrootシェルを終了し、再度**sudo ls**を実行してみてください。

管理者が **sudo** を設定する方法は他にもたくさんあり、特定のユーザーに対して特定の権限のみを指定する、検索されるパスを制限する、などがあります。**etc/sudoers** ファイルは、非常によく自己文書化されています。

しかし、たとえあなたのシステムにすでに**sudo**が設定されていたとしても、もう1つの設定を行うことを強くお勧めします。ほとんどのディストリビューションでは、一般ユーザーとルートユーザーとでは、実行可能ファイルを見つけるためのパスが異なります。特に、**/sbin**と**/usr/sbin**ディレクトリは検索されません。**sudo**は完全なルートユーザーではなく、ユーザーのPATHを継承するためです。

このため、このコースでは、多くのシステム管理ユーティリティへのフルパスを常に思い出させる必要があります。セキュリティの向上は、余計な入力やこれらのプログラムがどのディレクトリにあるのかの把握に見合わないでしょう。したがって、ホームディレクトリの **.bashrc** ファイルに次の行を追加することをお勧めします：

```
PATH=$PATH:/usr/sbin:/sbin
```

ログアウトした後、再度ログインすれば（再起動する必要はありません）、完全に有効になります。

第2章

Linuxファイルシステムのツリーレイアウト



2.1	一つの大きなファイルシステム.....	26
2.2	データディスタネーション.....	27
2.3	FHS Linux標準ディレクトリツリー.....	28
2.4	ルートディレクトリ	31
2.5	/bin.....	32
2.6	/boot.....	34
2.7	/dev.....	35
2.8	/エトセトラ	36
2.9	/home.....	37
2.10	/libおよび/lib64.....	39
2.11	/media.....	40
2.12	/mnt.....	41
2.13	/オプト	42
2.14	/proc.....	43
2.15	/sys.....	45
2.16	/root.....	46
2.17	/sbin.....	47
2.18	/srv.....	49
2.19	/tmp.....	50
2.20	/usr.....	51
2.21	/var.....	52
2.22	/run	53
2.23	研究室	54

学習目標

このセッションが終わるころには、次のことができるようになっているはずです：

- Linuxが1つの大きなファイルシステム・ツリーの構成を必要とする理由と、その方法に関する主な考慮事項を説明する。
- Filesystem Hierarchy Standardが果たす役割について説明する。(FHS)
- 起動時にルート(/)ディレクトリで利用可能でなければならないものと、システムが起動した後にのみ利用可能なものを説明します。
- 主なサブディレクトリの各ツリーを探索し、その目的を説明し、内容を検討する。

25

2.1 一つの大きなファイルシステム

一つの大きなファイルシステム

- 一つの大きなファイルシステムとして捉えられる
- ルートディレクトリを頂点とする倒立木
- サブディレクトリに追加のパーティションやファイルシステムをマウントすることができる
- Linuxディストリビューションの標準的な処方箋に従う。

Linuxは、他のUNIXベースのオペレーティングシステムと同様に、1つの大きなファイルシステムツリーで構成されています。実際には、ルート・ディレクトリである/を頂点とする逆さツリーとして図式化されるのが普通です。

この1つの大きな論理ファイルシステムの中には、サブディレクトリとして表示される、さまざまなポイントにマウントされた、1つ以上、さらには多くの異なるファイルシステムが存在する可能性があります。これらの異なるファイルシステムは、通常、異なるパーティション上にあり、ネットワーク上のデバイスを含む、任意の数のデバイス上にあることができます。

アプリケーションは通常、ファイルが実際にどの物理デバイスに存在するかは全く気にしません。

かつて、さまざまなUNIX系オペレーティングシステムが、この1本の大きな木をさまざまな方法で整理していました。そのため、複数のシステムでアプリケーションを作成したり、システム管理を行ったりすることは難しく、しばしば苛立ちを覚えることもありました。

その結果、Linuxエコシステムは、このような痛みを最小限に抑えるために、標準化された手順を確立するために努力してきました。

2.2 データディステーション

データディステーション

- 共有可能か非共有可能か
- バリアブルとスタティック

ファイルやデータが大きなディレクトリツリーの中でどのように構成されているかを語るとき、どのような情報を読み書きしなければならないかについて、いくつかの分類法を学ぶことが重要である。特に、2種類の区別があります：

1. 共有可能なものとそうでないもの

共有可能なデータとは、異なるホスト間で共有可能なデータです。非共有データとは、特定のホストに固有である必要があるものです。例えば、ユーザーホームディレクトリは共有可能ですが、デバイスロックファイルは共有できません。

2. バリアブルとスタティック

静的データとは、バイナリ、ライブラリ、ドキュメントなど、システム管理者の手助けがなければ変化しないものを指します。可変データは、システム管理者の助けがなくても変化する可能性があるものです。

このような論理的な区別は、さまざまなディレクトリ、あるいはパーティションやファイルシステムに存在するさまざまな種類の情報として具現化されることがよくあります。

2.3 FHS Linux標準ディレクトリツリー

ファイルシステムヒエラルキー規格

- ファイルの種類と場所を標準的にレイアウトしています。
 - ファイル位置の予測を可能にする
- BSDなどの歴史的な規格から発展したもの。
- **Linux Foundation** の下で管理されています。

https://refspecs.linuxfoundation.org/FHS_3.0/fhs-3.0.pdf.

Filesystem Hierarchy Standard (FHS) は、当初は Free Standards Group によって、現在は Linux Foundation によって管理されており、存在する必要がある主なディレクトリを指定し、その目的を説明しています。
https://refspecs.linuxfoundation.org/FHS_3.0/fhs-3.0.pdf からダウンロードすることができます。

標準的なレイアウトを指定することで、FHSはファイルの位置の予測を簡素化します。ほとんどのLinuxディストリビューションは、標準レイアウトを尊重していますが

FHSは、おそらくどれも正確には守っていないでしょうし、最後の公式バージョンは、いくつかの新しい開発を考慮していないかもしれません。

ディストリビューションは実験を好み、最終的にはいくつかの実験が一般に受け入れられるようになる。

FHS Linux標準ディレクトリツリー

表2.1：主なディレクトリー Part I

/	ファイルシステム階層全体のプライマリーディレクトリ。
/bin ディレクトリの目	シングルユーザーモードで利用可能でなければならない必須の実行プログラム。
/boot	カーネル、initrdまたはinitramfsイメージ、ブートなど、システムを起動するために必要なファイル。 設定ファイルやブートルoaderプログラム
/dev	デバイスノード、ハードウェアおよびソフトウェアデバイスと対話するために使用される。
/etc エト セト ラ	システム全体の設定ファイルです。
/home	個人設定やファイルなどを含むユーザーホームディレクトリ。
/lib	binおよび/sbinにある実行バイナリが必要とするライブラリ。
/lib64	64-bit /binおよび/sbinの実行バイナリが必要とするライブラリ。 32ビットと64ビットの両方のプログラムに対応しています。
/media	CD、DVD、USBメモリなどのリムーバブルメディアのマウントポイント。
/mnt	一時的にマウントされたファイルシステムです。

ルートディレクトリの下には、ディストリビューション固有のディレクトリが追加されている場合があります。これには、雑多なデータに使用できる `/misc` や、`tftp` を使用したブートに使用する `/tftpboot` があります。このディレクトリにファイルがある場合は、ディスクレスワークステーションのブートに関連するものです。他のディレクトリを持つことはFHSに違反しませんが、標準によって指示されたディレクトリ以外のディレクトリにコンポーネントを持つことは、FHSに違反することに注意してください。

FHS Linux標準ディレクトリツリー、続き

ディレクトリ 目的

クター

/オプション

オプションのアプリケーションをインストールするためのディレクトリ

/proc

システムおよびその上で動作するプロセスに関する情報を与える仮想擬似ファイルシステム。システムパラメーターを変更するために使用できます。

/run

ランタイム変数データで、起動時からのシステムを説明する情報を含む。古い/var/runを置き換えます。

/sys

システムおよびその上で動作するプロセスに関する情報を提供する仮想の擬似ファイルシステムです。システムパラメーターを変更するために使用することができます。デバイスツリーに似ており、ユニファイドコンピューティングの一部である。デバイスモデルです。

/root

rootユーザーのホームディレクトリです。

/sbin

必須システムバイナリです。

/srv

システムから提供されるサイト固有のデータ。めったに使われない。

/tmp

一時ファイル。多くのディストリビューションでは、再起動すると失われ、メモリ内のラムディスクである可能性があります。

/usr

マルチユーザーアプリケーション、ユーティリティ、データ、理論上は読み取り専用です。

/var

システム運用中に変化する可変データ。

これらは、メインのルート (/) ディレクトリから離れた追加ディレクトリです。

2.4 ルートディレクトリ

ルートディレクトリ

- に十分でなければならない。
 - ブート
 - リストア
 - リカバー
 - リペア
- 必要に応じ、システム
- アプリケーションやパッケージは、ルートディレクトリのサブディレクトリを新たに作成してはならない



/ 対 /root

スーパーユーザーであるrootのホームディレクトリである/rootの話ではない。

ファイルシステム全体を1つの大きな木として捉えることができますが、これまで指摘してきたように、複数のパーティションやファイルシステムが結合している場合もあります。

ルートディレクトリ自体が含まれるパーティションやファイルシステムはかなり特殊で、/home、/var、/optなどのディレクトリを持つ他のコンポーネントは後でマウントするため、特別な専用パーティションにあることが多いです。

ルートパーティションには、システムを起動し、他のすべてのファイルシステムをマウントするために必要なすべての必須ファイルが含まれている必要があります。このため、ユーティリティ、設定ファイル、ブートローダ情報、その他の重要な起動データが必要です。それには十分な容量が必要です：

- システムを起動します。
- テープなどのリムーバブルメディアやNASなどの外部メディアにあるシステムバックアップからシステムを復元する。
- 故障したシステムを診断し、再構築するためのツールを、経験豊富なメンテナンス担当者が持っている必要があります。

FHSによると、アプリケーションやパッケージは、ルートディレクトリの新しいサブディレクトリを作成してはいけいとされています。



/ は /root ではありません

/rootはスーパーユーザーのホームディレクトリです。ここでの言葉の選び方は、混乱を招きかねません！

2.5 /bin

/bin

- システム管理者と非特権ユーザーの両方が必要とする実行可能なプログラムとスクリプトが含まれており、他のファイルシステムがまだマウントされていない場合、例えば、罪のあるユーザーやリカバリーモードでブートするときに必要となります。
- また、スクリプトが間接的に使用する実行ファイルが含まれている場合があります。
- サブディレクトリを含めてはいけません。

/binおよび/usr/bin

最近のディストリビューションでは、`/bin`と`/usr/bin`（および`/sbin`と`/usr/sbin`）を分離する戦略を放棄し、シンボリックリンクを持つ1つのディレクトリを持つことで、2ディレクトリ表示を維持しているものがあります。また、`/usr`を別パーティションに配置し、起動後にマウントすることを可能にするという、古くからあるコンセプトは時代遅れと考えられています。

`bin/`に存在しなければならない必須プログラムは以下の通りです：

`cat`、`chgrp`、`chmod`、`chown`、`cp`、`date`、`dd`、`df`、`dmesg`、`echo`、`false`、`hostname`、`kill`、`ln`、`login`、`ls`、`mkdir`、`mknod`、`more`、`mount`、`mv`、`ps`、`pwd`、`rm`、`rmdir`、`sed`、`sh`、`stty`、`su`、`sync`、`true`、`umount` および `uname`。

[や`test`もあってもよいし、任意に含まれてもよい：

`csch`、`ed`、`tar`、`cpio`、`gunzip`、`zcat`、`netstat`、`ping`。

`bin/`に置くほど重要でないと判断されたコマンドバイナリは、`/usr/bin`に置かれます。非 root ユーザーのみが必要とするプログラムは、このカテゴリに配置されます。

/bin 例

```

student@ubuntu:~$ ls /bin
bash          chacl         fsck.btrfs   lowntfs-3g   ntfscomp     run-parts    tempfile
brltty        chgrp         fuser        ls            ntfsfallocate sed           touch
btrfs         chmod         fusermount  lsblk        ntfsfix      setfacl      true
btrfsck       chown         getfacl     lsmod        ntfsinfo     setfont      udevadm
btrfs-debug-tree chvt         grep        mkdir        ntfsls       setupcon     ulockmgr_server
btrfs-find-root cp            gunzip      mkfs.btrfs   ntfsmove     sh           umount
btrfs-image   cpio         gzexe       mknod        ntfsrecover  sh.distrib   uname
btrfs-map-logical dash         gzip        mktemp       ntfssecaudit sleep         uncompress
btrfs-select-super date         hciconfig   more         ntfstruncate ss            unicode_start
btrfstune     dd           hostname    mount         ntfsusermap  static-sh    vdir
btrfs-zero-log df           ip          mountpoint   ntfswipe     stty         wdctl
bunzip2       dir          journalctl  nt           ntfswipe     su           which
busybox       dmesg        kbd_mode    nt-gnu       open          sync         whiptail
bzip          dnsdomainname keyctl       mv           openvt       systemctl   ypdomainname
bzip2         domainname   kill        nano         pidof        systemd     zcat
bzdiff        dumpkeys     kmod        nc           ping         systemd-ask-password zcmp
bzegrep       echo         less        nc.openbsd   ping4        systemd-escape zdiff
bzeke         ed           lessecho   netcat       ping6        systemd-hwdb zegrep
bzfgrep       efibootdump lessfile    netstat      plymouth     systemd-inhibit zfgrep
bzgrep        efibootmgr  lesskey    networkctl   ps           systemd-machine-id-setup zforce
bzip2         egrep        lesspipe   nisdomainname readlink     systemd-notify zgrep
bzip2recover false        ln          ntfs-3g      red          systemd-sysusers zless
bzless        fgconsole   loadkeys   ntfs-3g.probe rm           systemd-tmpfiles znore
bzmore        fgrep       login      ntfscluster  rmdir        systemd-tty-ask-password-agent znew
cat           findmnt     loginctl   ntfscluster  nano         tar

```

図2.1 : /bin ディレクトリです : Ubuntu 18.04

RHEL、CentOS、Fedora、Ubuntuの最近のディストリビューションバージョンでは、/binと/usr/binがシンボリックリンクされているので、実際には同じものである。

上記のスクリーンショットはUbuntu 18.04のもので、それ以降のバージョンでは、リンクされていない別のディレクトリが存在しなくなりました。

2.6 /boot

/boot

- ブートプロセスに必要なすべてのものが含まれています。絶対に必要なファイルは以下の2つです：
 - **vmlinuz**です：圧縮されたLinuxカーネル。
 - **initramfs**：本当のルートファイルシステムが利用可能になる前にマウントされる初期RAMファイルシステムです。
- カーネルがユーザーモードプログラムの実行を開始する前に使用するデータを格納します。
- また、情報提供やデバッグに使用する2つのファイルも含まれています：
 - **config** カーネルコンパイルの設定に使用します。
 - **System.map**：カーネルシンボルテーブル、デバッグに使用されます。
- ディストリビューションによっては、他のファイルが存在する場合があります。

ブートの正確な内容は、ディストリビューションや時間によって異なります：

```
student@ubuntu: ~
student@ubuntu:~$ ls -lF /boot
total 391800
-rw-r--r-- 1 root root 261694 Jul 12 03:51 config-5.15.0-43-generic
-rw-r--r-- 1 root root 111027 Aug 1 09:38 config-5.19.0
drwx----- 3 root root 4096 Dec 31 1969 eft/
drwxr-xr-x 6 root root 4096 Aug 1 09:38 grub/
lrwxrwxrwx 1 root root 17 Aug 1 09:38 initrd.img -> initrd.img-5.19.0
-rw-r--r-- 1 root root 62362983 Aug 1 09:12 initrd.img-5.15.0-43-generic
-rw-r--r-- 1 root root 59866723 Aug 1 09:38 initrd.img-5.19.0
-rw-r--r-- 1 root root 182800 Feb 6 14:35 memtest86+.bin
-rw-r--r-- 1 root root 184476 Feb 6 14:35 memtest86+.elf
-rw-r--r-- 1 root root 184980 Feb 6 14:35 memtest86+_multiboot.bin
-rw----- 1 root root 6250707 Jul 12 03:51 System.map-5.15.0-43-generic
-rw-r--r-- 1 root root 4491643 Aug 1 09:38 System.map-5.19.0
-rwxr-xr-x 1 root root 254417056 Aug 1 09:38 vmlinuz-5.19.0*
lrwxrwxrwx 1 root root 14 Aug 1 09:38 vmlinuz -> vmlinuz-5.19.0
-rw----- 1 root root 11090688 Jul 12 04:29 vmlinuz-5.15.0-43-generic
-rw-r--r-- 1 root root 8260592 Aug 1 09:38 vmlinuz-5.19.0
student@ubuntu:~$
```

図2.2 : Ubuntu 20.04の/bootディレクトリ

これらのファイルは、Linuxディストリビューションとカーネルのバージョンに依存する長い名前を持っています。また、**initramfs**の代わりに**initrd**（初期ラムディスク）がある場合もあります。

2.7 /dev

/dev

- キャラクター と ブロック デバイスノード、ま
- 適切なシステム運用のために不可欠な
- 最近のシステムはudevでデバイスファイルを自動的に管理して
- 古いシステム（または埋め込みデバイス）では、インストール時または必要に応じてMAKEDEVまたはmknodによっ

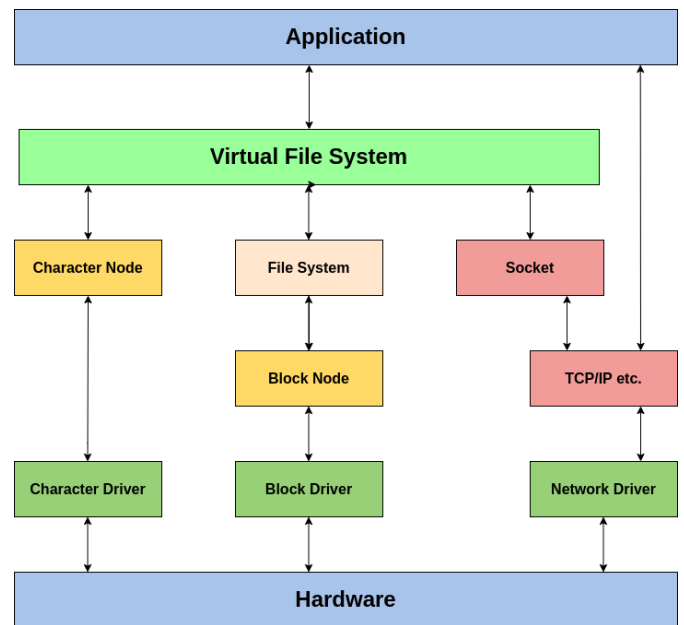


図2.3：デバイスノード

このディレクトリには、システムに組み込まれている、または接続されているデバイスを表す特別なデバイスファイル（デバイスノードとも呼ばれる）が含まれています。これらの特殊なファイルは、システムが正しく機能するために不可欠です。このようなデバイスファイルは、文字（バイトストリーム）およびブロックI/Oデバイスを表します。ネットワークデバイスは、Linuxではデバイスノードを持たず、代わりにeth1やwlan0のような名前参照されます。

```
File Edit View Search Terminal Help
c7:/tmp>ls -l /dev
total 0
....
crw----- 1 root root      5,  1 May 26 07:05 console
....
lrwxrwxrwx 1 root root      13 May 26 07:04 fd -> /proc/self/fd
....
brw-rw---- 1 root disk      7,  0 May 26 07:05 loop0
crw-rw---- 1 root disk    10, 237 May 26 07:05 loop-control
....
crw-rw---- 1 root lp         6,  0 May 26 07:05 lp0
crw-rw---- 1 root lp         6,  1 May 26 07:05 lp1
....
brw-rw---- 1 root disk      8,  0 May 26 07:05 sda
brw-rw---- 1 root disk      8,  1 May 26 07:05 sda1
brw-rw---- 1 root disk      8,  2 May 26 07:05 sda2
brw-rw---- 1 root disk      8,  3 May 26 07:05 sda3
....
lrwxrwxrwx 1 root root      15 May 26 07:04 stderr -> /proc/self/fd/2
lrwxrwxrwx 1 root root      15 May 26 07:04 stdin -> /proc/self/fd/0
lrwxrwxrwx 1 root root      15 May 26 07:04 stdout -> /proc/self/fd/1
crw-rw-rw- 1 root tty        5,  0 May 26 07:05 tty
crw--w---- 1 root tty        4,  0 May 26 07:05 tty0
....
c7:/tmp>
c7:/tmp>
```

図 2.4: /dev ディレクトリ

最近のLinuxディストリビューションはすべてudevシステムを使用しており、デバイスが見つかったときに必要な分だけ/devにノードを作成します。マウントされていないファイルシステムで/devディレクトリを見ると、空っぽであることがわかるでしょう。

2.8 /エトセトラ

/エトセトラ

- マシンのローカルにある設定ファイル（および一部のスタートアップスクリプト）が含まれています。
- このディレクトリにはバイナリを置くことはできません
- ディストリビューションでは、特定のファイルやディレクトリを追加することがよくあります。例えば、Red Hatは、`/etc/sysconfig`を含む多くの他のディレクトリを追加します。

このディレクトリには、マシンローカルな設定ファイルといくつかのスタートアップスクリプトが含まれます。実行可能なバイナリプログラムはありません。

FHSがこのディレクトリにあることを要求するファイルやディレクトリは以下の通りです：

`csh.login`, `exports`, `fstab`, `ftpusers`, `gateways`, `gettydefs`, `group`, `host.conf`, `hosts.allow`, `hosts.deny`, `hosts.equiv`, `hosts.lpd`, `inetd.conf`, `inittab`, `issue`, `ld.so.conf`, `motd`, `mtab`, `mttools.conf`, `networks`, `passwd`, `printcap`, `profile`, `protocols`, `resolv.conf`, `rpc`, `securetty`, `services`, `shells`, `syslog.conf`.

これらのファイルの中には、フロッピーディスクで使用されている`mttools.conf`のように、今日ではかなり無関係なものもある。また、ソフトウェアの陳腐化により、**FHS**が何と言おうと、もう見つからないものもあります。

ディストリビューションでは、`/etc`に設定ファイルやディレクトリを追加することがよくあります。例えば、Red Hatは、多くのシステム設定ファイルやディレクトリが存在する`/etc/sysconfig`を含む他の多くのディレクトリを追加しています。

その他、重要なサブディレクトリがあります：

- `/etc/skel`：新しく作成されたホームディレクトリに入力するために使用されるスケルトンファイルが含まれています。
- `/etc/systemd`：`systemd`を使用する際に、システムサービスを開始、停止、制御するための設定スクリプトが含まれているか、またはそれを指す。
- `/etc/init.d`：**System V**の初期化を使用する場合のスタートアップとシャットダウンのスクリプトが含まれています。

2.9 /home

/home

- 通常、ユーザーのホームディレクトリが含まれます
- 別の名前がついている可能性があります
- ユーザーグループ用のサブディレクトリを含む場合がある

Linuxシステムでは、`/home/coop`、`/home/student`などのように、ユーザーディレクトリは慣習的に`/home`の下に置かれます。個人設定、データ、実行可能なプログラムはすべてこのディレクトリ階層に配置されます。また、`/home`には、`/home/students`、`/home/staff`、`/home/aliens`などのように、さまざまなグループやユーザーのさまざまな関連付けのためのサブディレクトリを含めることができる。

他のUNIX系OSでも、`/home`ディレクトリツリーの概念は存在するが、微妙に異なる場合がある。例えば、**Solaris**では、ユーザーディレクトリは`/export/home`に作成され、その後、自動マウント機能により最終的に`/home`にマウントされます。これは、ホームディレクトリが企業ネットワーク上のどこにあってもよく、おそらくNFSサーバー上にあり、使用時にホームディレクトリが自動的にマウントされるというのが通常の状態だからです。

Linuxにも同様の自動マウント機能がありますが、多くのユーザーはその存在に気づいていませんし、自己完結型のシステムでは、NFSマウントの概念はおそらく適用されないと思われます。

あるユーザーは、常に環境変数`$HOME`を自分のルートディレクトリに置き換えるか、あるいは略語の`~`に置き換えることができます; すなわち、以下は等価です:

```
$ ls -l $HOME/public_html
$ ls -l ~/public_html
```

ただし、1つだけ例外があります。Linuxシステムのルートユーザーのホームディレクトリは、常に`/root`の下にあります。一部の古い

UNIX システムでは、代わりに `/` を使用することがありますが、これは乱雑になる可能性があります。

/home 例

```
student@ubuntu: ~  
student@ubuntu:~$ ls -la /home/student  
total 132  
drwxr-xr-x 19 student student 4096 May 26 09:36 .  
drwxr-xr-x  3 root    root    4096 Apr 14 10:08 ..  
-rw----- 1 student student 1494 May 25 12:45 .bash_history  
-rw-r--r-- 1 student student 220 Apr 14 10:08 .bash_logout  
-rw-r--r-- 1 student student 3771 Apr 14 10:08 .bashrc  
drwx----- 14 student student 4096 May 19 12:27 .cache  
drwx----- 3 student student 4096 May  1 10:07 .compiz  
drwx----- 14 student student 4096 Apr 20 10:55 .config  
drwx----- 3 root    root    4096 May  1 10:04 .dbus  
drwxr-xr-x  2 student student 4096 Apr 14 10:13 Desktop  
-rw-r--r--  1 student student  25 Apr 14 10:13 .dmrc  
drwxr-xr-x  2 student student 4096 Apr 14 10:13 Documents  
drwxr-xr-x  2 student student 4096 Apr 14 10:13 Downloads  
drwx----- 3 root    root    4096 May  1 10:06 .emacs.d  
-rw-r--r--  1 student student 8980 Apr 14 10:08 examples.desktop  
drwx----- 2 student student 4096 Apr 14 10:35 .gconf  
-rw-----  1 student student 4134 May 26 08:05 .ICEauthority  
drwxrwxr-x  2 student student 4096 May  1 07:26 LFT  
drwxrwxr-x  3 student student 4096 Apr 14 10:13 .local  
drwxr-xr-x  2 student student 4096 Apr 14 10:13 Music  
drwxr-xr-x  2 student student 4096 Apr 14 10:13 Pictures  
-rw-r--r--  1 student student  675 Apr 14 10:08 .profile  
drwxr-xr-x  2 student student 4096 Apr 14 10:13 Public  
drwx----- 2 student student 4096 May 19 12:28 .ssh  
-rw-r--r--  1 student student  0 Apr 14 10:15 .sudo_as_admin_successful  
drwxr-xr-x  2 student student 4096 Apr 14 10:13 Templates  
drwxr-xr-x  2 student student 4096 Apr 14 10:13 Videos  
-rw-r--r--  1 student student  183 Apr 14 10:17 .wget-hsts  
-rw-----  1 student student  51 May 26 08:05 .Xauthority  
-rw-----  1 student student 3462 May 26 08:05 .xsession-errors  
-rw-----  1 student student 3462 May 25 08:49 .xsession-errors.old  
student@ubuntu:~$
```

図2.5 : /homeディレクトリ

2.10 /libおよび/lib64

/libおよび/lib64

- のバイナリを実行するために必要なライブラリのみを格納します。
`/bin`と`/sbin`
- 一部のディストリビューションでは、64ビットライブラリを`/lib64`に配置しています。
- カーネルモジュールは`/lib/modules`に置く。
- PAMモジュールは、`/lib64/security`や`/lib/x86_64-linux-gnu/security`など、ディストリビューションに依存する場所に置く。

`/lib`および`/usr/lib`

最近のディストリビューションでは、`/lib`と`/usr/lib`（および`/lib64`と`/usr/lib64`）を分離する戦略を放棄し、シンボリックリンクを持つ1つのディレクトリを持つだけで、2ディレクトリ表示を実現しているものがあります。また、`/usr`を別パーティションに配置し、起動後にマウントすることを可能にするという、古くからあるコンセプトも時代遅れとみなしています。

これらのディレクトリには、`/bin`と`/sbin`にあるバイナリを実行するために必要なライブラリのみを格納する必要があります。これらのライブラリは、システムの起動やルートファイルシステム内のコマンドの実行に特に重要です。

カーネルモジュール（多くの場合、デバイスやファイルシステムのドライバ）は、`/lib/modules/<kernel-version-number>`の下に、次のように配置されます。

`/lib/modules/5.19.4.`

PAM (Pluggable Authentication Modules) ファイルは、`/lib64/security` などのディストリビューションに依存する場所に格納されています。

`/lib/x86_64-linux-gnu/security.`

32ビットと64ビットのバイナリをサポートするシステムでは、システム上に両方の種類のライブラリを保持する必要があります。ディストリビューションによっては、32ビット・ライブラリ（`/lib`）と64ビット・ライブラリ（`/lib64`）のための別々のディレクトリが存在します。

`$ ls -l /lib*`

```
lrwxrwx 1 root root 7 Apr 23 2020 /lib -> usr/lib
lrwxrwx 1 root 9 Apr 23 2020 /lib64 -> usr/lib64
```

2.11 /media

/media

- リムーバブルメディアのファイルシステムのマウントに使用
 - CD ROMS
 - DVD
 - USBメモリー
- 動的に取り付け可能



/メディアとランの比較

- 最近のシステムでは、**/media**の代わりに**/run/media/[username]/...**を使用します。

このディレクトリは、通常、リムーバブルメディアにファイルシステムをマウントするために使用されていました。CD、DVD、USBドライブ、さらには旧石器時代のフロッピーなども含まれる。

Linuxでは、メディアの挿入時に動的にマウントされ、**udev**は、設定ファイルで指定された**udev**ルールで設定された名前で、ディレクトリを作成し、そこにリムーバブルファイルシステムをマウントする。アンマウントして取り外すと、マウントポイントとして使用されていたディレクトリは消滅する。

メディアに複数のパーティションとファイルシステムがある場合、複数のエントリが表示されます。多くのLinuxディストリビューションでは、メディアをマウントするとファイルマネージャー（**Nautilus**など）がポップアップ表示されます。



/run は /media に取って代わられました。

注：現在のLinuxディストリビューションでは、リムーバブルメディアは、**/run/media/[username]/...**の代わりにポップアップします。

2.12 /mnt

/mnt

- 他のファイルシステムを一時的にマウントするための、ファイルシステム内の特別な場所。
 - ネットワークマウントされたファイルシステム（NFS、SAMBAAなど）
 - 一時的なパーティションや論理ボリューム

```
$ sudo mount c8:/ISO_IMAGES /mnt
```

このディレクトリは、システム管理者が必要なときにファイルシステムを一時的にマウントできるように提供されています。一般的な使用法は、ネットワークファイルシステムなどです：

- エヌエフエス
- サンバ
- シーアイエフエス
- エーエフエス

歴史的には、/mntは、最近のシステムではmedia（または/run/media）の下にマウントされている種類のファイルにも使用されていました。

一般的に、このディレクトリはインストールプログラムによって使用されるべきではありません。現在使用されていない別の一時ディレクトリを使用するのがよいでしょう。

2.13 /opt

/opt

- アドオンアプリケーションソフトウェアパッケージのインストールに使用します。
- パッケージの静的ファイルは `/opt/[somepackage]` ディレクトリにある必要があります。

`$ ls -l /opt`

```
合計20K
drwxr-xr-x. 4 root root 4.0K Jul 26 2021 brother
drwxr-xr-x. 4 root ルート 4.0K 2019年9月1日 google
drwxr-xr-x   3 root ルート 4.0K Jul 7 08:03 rh
drwxr-xr-x 12 root root 4.0K Jul 1 09:55 VirtualBox
drwxr-xr-x.30 root root 4.0K 5月30日 13:08 zoom
```

このディレクトリは、他のソフトウェアが共有するディレクトリにファイルを分散させるのではなく、すべてまたは大部分のファイルを1つの孤立した場所に保管することを望むソフトウェアパッケージのために設計されています。

例えば、**dolphy app**が `/opt` 以下に存在するパッケージの名前だった場合、そのファイルはすべて `/opt/dolphy_app` 以下のディレクトリに存在し、バイナリは `/opt/dolphy_app/bin`、マニュアルページは `/opt/dolphy_app/man` に含まれるはずです。

このため、ソフトウェアのインストールやアンインストールが比較的容易になります。また、システム管理者がパッケージ内の各ファイルの性質を判断することも容易になります。

ただし、後述するRPMやAPTなどのパッケージングシステムを利用すれば、このような反社会的な振る舞いをしなくても、ファイルマニフェストやロケーションを明確に意識したインストールやアンインストールも容易にできる。

Linuxでは、`/opt` ディレクトリは、独自のソフトウェアを持つアプリケーションプロバイダーや、配布のばらつきによる複雑さを避けたい人たちによって、しばしば使用されます。たとえば、あるシステムでは、パッケージは `/opt/brother`、`/opt/zoom`、および `/opt/zoom` にあります。

`/opt/google` で、後者には `chrome` と `earth` のサブディレクトリがあります。

`/opt/bin`、`/opt/doc`、`/opt/include`、`/opt/info`、`/opt/lib`、`/opt/man` は、ローカルシステム管理者が使用するために予約されているディレクトリです。パッケージは、これらの予約ディレクトリにリンクまたはコピーされるファイルを提供することができますが、パッケージは、これらの特別なディレクトリにあるプログラムなしで機能することもできなくてはなりません。ほとんどのシステムでは、これらのディレクトリは作成されません。

2.14 /proc

/proc

- 擬似ファイルシステム：ディスク上のどこにも保存されません。
- カーネルデータ構造へのインターフェイス
- 各アクティブプロセスは、このディレクトリの中にサブディレクトリを持ちます。

このディレクトリは、すべての情報がディスク上ではなく、メモリ上にのみ存在する擬似ファイルシステムのマウントポイントである。`dev`と同様に、`/proc`ディレクトリは、実行されていないシステムでは空です。

カーネルは、`/proc` エントリを通じていくつかの重要なデータ構造を公開しています。さらに、システム上でアクティブな各プロセスは、プロセスの状態、使用しているリソース、およびその履歴に関する詳細な情報を提供する独自のサブディレクトリを持ちます。

`proc`のエントリは、しばしば**仮想ファイル**と呼ばれ、興味深い性質を持っています。ほとんどの場合、サイズは0バイトと表示されていますが、見てみると大量の情報が含まれていることがあります。

また、仮想ファイルの時刻や日付の設定の多くは、現在の時刻や日付を反映しており、常に変化していることを示すものである。実際には、これらのファイルの情報は閲覧時にのみ取得され、常時または定期的に更新されているわけではありません。

`proc/interrupts`、`/proc/meminfo`、`/proc/mounts`、`/proc/partitions`などの重要な擬似ファイルは、システムのハードウェアを最新に垣間見ることができます。

その他、`/proc/filesystems`や`/proc/sys`ディレクトリのように、システム構成情報やインターフェイスを提供するものもあります。組織的な目的のために、類似したトピックの情報を含むファイルは、仮想ディレクトリとサブディレクトリにグループ化されます。例えば、`/proc/scsi`には、すべての物理的なSCSIデバイスの情報が含まれています。同様に、`プロセス`ディレクトリには、システム上で実行中の各プロセスに関する情報が含まれています。

このコースでは、`/proc`のエントリを調べますが、カーネル設定とシステム監視に関する次の章では、より詳細に調べます。

/proc 例

```
student@ubuntu: ~
student@ubuntu:~$ ls -F /proc
1/ 13/ 200/ 2288/ 250/ 33/ 40/ 5383/ 99/ misc
10/ 137/ 201/ 229/ 251/ 3315/ 4047/ 54/ 990/ modules
100/ 14/ 202/ 23/ 2529/ 3336/ 41/ 5410/ acpi/ mounts@
1005/ 1457/ 203/ 230/ 2572/ 3353/ 4156/ 5436/ buddyinfo mpt/
1007/ 1465/ 204/ 231/ 26/ 3366/ 4169/ 5438/ bus/ mtrr
1008/ 1478/ 205/ 232/ 27/ 3394/ 418/ 581/ cgroups net@
1009/ 15/ 206/ 233/ 274/ 34/ 419/ 583/ cmdline pagetypeinfo
101/ 16/ 207/ 234/ 2756/ 3419/ 42/ 6/ consoles partitions
1010/ 1649/ 208/ 235/ 276/ 3430/ 420/ 601/ cpuinfo sched_debug
1011/ 1780/ 209/ 236/ 28/ 3439/ 422/ 7/ crypto schedstat
1012/ 1782/ 21/ 2368/ 280/ 3441/ 43/ 715/ devices scsi/
1013/ 1798/ 210/ 237/ 281/ 35/ 44/ 717/ diskstats self@
1014/ 18/ 211/ 2373/ 2979/ 3512/ 45/ 718/ dma slabinfo
102/ 1822/ 212/ 238/ 298/ 3513/ 4595/ 721/ driver/ softirqs
1028/ 188/ 213/ 2385/ 2989/ 3514/ 4596/ 723/ execdomains stat
103/ 189/ 214/ 239/ 30/ 3515/ 4599/ 728/ fb swaps
104/ 19/ 2142/ 24/ 300/ 3532/ 46/ 731/ filesystems sys/
105/ 190/ 215/ 240/ 3081/ 3534/ 4601/ 733/ fs/ sysrq-trigger
106/ 191/ 216/ 241/ 31/ 3569/ 47/ 735/ interrupts sysvipc/
107/ 1916/ 217/ 242/ 32/ 3581/ 473/ 743/ iomem thread-self@
108/ 192/ 218/ 243/ 3228/ 3589/ 478/ 746/ ioports timer_list
1083/ 1922/ 219/ 2436/ 3243/ 36/ 4792/ 750/ irq/ timer_stats
1085/ 193/ 2196/ 244/ 3244/ 3600/ 483/ 753/ kallsyms tty/
109/ 1930/ 2197/ 2443/ 3245/ 3613/ 485/ 755/ kcore uptime
11/ 194/ 22/ 2449/ 3246/ 3621/ 486/ 781/ keys version
110/ 1940/ 220/ 245/ 3247/ 3655/ 492/ 798/ key-users version_signature
1121/ 195/ 221/ 2456/ 3251/ 37/ 50/ 8/ kmsg vmallocinfo
1134/ 196/ 222/ 246/ 3255/ 370/ 502/ 828/ kpagecgroup vnstat
1140/ 197/ 223/ 2461/ 3257/ 38/ 51/ 9/ kpagecount zoneinfo
1141/ 198/ 224/ 247/ 3264/ 39/ 52/ 96/ kpageflags
1147/ 1986/ 225/ 248/ 3268/ 3910/ 53/ 98/ loadavg
116/ 199/ 226/ 2488/ 3272/ 393/ 5310/ 981/ locks
1180/ 2/ 227/ 249/ 3274/ 399/ 5332/ 983/ mdstat
12/ 20/ 228/ 25/ 3294/ 4/ 5358/ 988/ meminfo
student@ubuntu:~$
```

図2.6 : /procディレクトリ

ある特定のプロセスに属する **/proc** ディレクトリの中身を紹介します。

```
student@ubuntu: ~
student@ubuntu:~$ ls -F /proc/3589
attr/ coredump_filter gid_map mountinfo oom_score schedstat status
autogroup cpuset io mounts oom_score_adj sessionid syscall
auxv cwd@ limits mountstats pagemap setgroups task/
cgroup environ loginuid net/ personality snaps timers
clear_refs exe@ map_files/ ns/ projid_map stack timerslack_ns
cmdline fd/ maps numa_maps root@ stat uid_map
comm fdinfo/ mem oom_adj sched statm wchan
student@ubuntu:~$
```

図2.7 : /proc/[pid]ディレクトリ

重要なシステム情報を示す多くのファイルの1つ、 **/proc/interrupts** です。

```
File Edit View Search Terminal Help
x7:/home/coop>cat /proc/interrupts
CPU0 CPU1 CPU2 CPU3
0: 88 0 0 0 IR-IO-APIC 2-edge timer
1: 566 1 2153 0 IR-IO-APIC 1-edge i8042
8: 1 0 0 0 IR-IO-APIC 8-edge rtc0
9: 17990 11 552 27 IR-IO-APIC 9-fasteoi acpi
12: 64336 21 186157 20 IR-IO-APIC 12-edge i8042
16: 0 0 0 0 IR-IO-APIC 16-fasteoi i801_smbus
120: 0 0 0 0 DMAR-MSI 0-edge dmar0
121: 0 0 0 0 DMAR-MSI 1-edge dmar1
122: 217295 1607 4039 59571 IR-PCI-MSI 376832-edge ahci[0000:00:17.0]
123: 3 0 46 0 IR-PCI-MSI 514048-edge snd_hda_intel:card0
124: 95360 74 49535 192 IR-PCI-MSI 327680-edge xhci_hcd
125: 591286 3 272983 2 IR-PCI-MSI 32768-edge i915
126: 84 16 149 20 IR-PCI-MSI 520192-edge enp0s3l1f6
127: 225390 29 383 46 IR-PCI-MSI 2097152-edge iwlwifi
NMIO: 24 120 130 119 Non-maskable interrupts
LOC: 2255506 2201465 2360863 2239138 Local timer interrupts
SPU: 0 0 0 0 Spurious interrupts
PMI: 24 120 130 119 Performance monitoring interrupts
IWI: 0 0 3 0 IRQ work interrupts
RTR: 24 3 0 0 APIC ICR read retries
RES: 185530 146421 95420 45924 Rescheduling interrupts
CAL: 76456 74889 78143 76863 Function call interrupts
TLB: 75401 73603 76833 75025 TLB shootdowns
ERR: 0
MIS: 0
PIN: 0 0 0 0 Posted-interrupt notification event
PIW: 0 0 0 0 Posted-interrupt wakeup event
x7:/home/coop>
```

図2.8 : /proc/interruptsの内容

2.15 /sys

/sys

- もう一つの擬似ファイルシステム：ディスク上のどこにも保存されない。
- デバイスとドライバ、カーネルモジュール、システム構成構造に関する情報が含まれています。
- システムの動作やパラメータを調べたり、変更したりするのに使用できる。
- **proc**よりも制御性が高い

このディレクトリは、すべての情報がディスク上ではなく、メモリ上にのみ存在する**sysfs**擬似ファイルシステムのマウントポイントです。devや**proc**と同様に、**/sys**ディレクトリは、実行されていないシステムでは空です。

sysfsは、システムに関する情報を収集し、実行中のシステムの動作を変更するために使用されます。その意味ではしかし、**/proc**よりも若く、どのようなエントリーを含むことができるかについての厳しい基準を守ってきました。たとえば、**/sys**にあるほとんどすべての擬似ファイルには、1行または値しか含まれておらず、**/proc**にあるような長いエントリーはありません。

カーネル設定とシステム監視に関する次の章では、**/proc**と同様に、**/sys**のエントリーをこのコースを通して調べます。

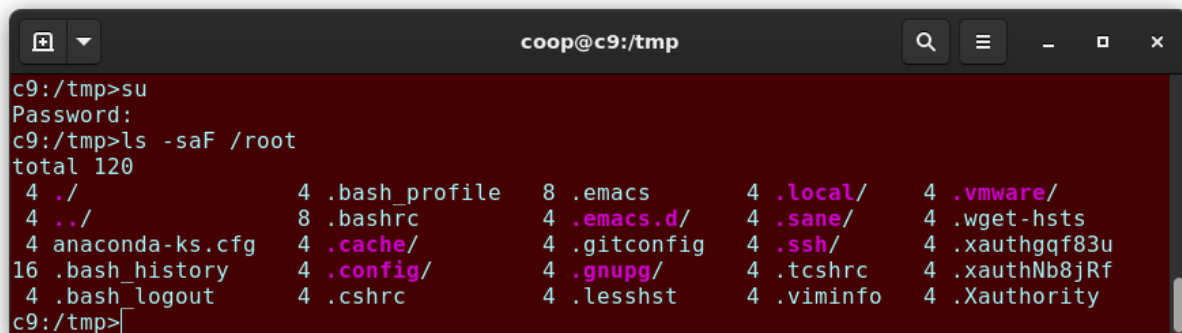
```
File Edit View Search Terminal Help
c7:/tmp>ls -lF /sys
total 0
drwxr-xr-x  2 root root 0 May 26 12:08 block/
drwxr-xr-x 23 root root 0 May 26 12:08 bus/
drwxr-xr-x 41 root root 0 May 26 12:08 class/
drwxr-xr-x  4 root root 0 May 26 12:08 dev/
drwxr-xr-x 23 root root 0 May 26 12:08 devices/
drwxr-xr-x  5 root root 0 May 26 12:08 firmware/
drwxr-xr-x  5 root root 0 May 26 12:08 fs/
drwxr-xr-x 11 root root 0 May 26 12:08 kernel/
drwxr-xr-x 88 root root 0 May 26 12:08 module/
drwxr-xr-x  2 root root 0 May 26 12:08 power/
c7:/tmp>
```

図 2.9 : /sys ディレクトリ

2.16 /root

/root

- rootユーザーのホームディレクトリとして使用



```
coop@c9:/tmp
c9:/tmp>su
Password:
c9:/tmp>ls -saF /root
total 120
4 ./
4 ../
4 anaconda-ks.cfg
16 .bash_history
4 .bash_logout
4 .bash_profile
8 .bashrc
4 .cache/
4 .config/
4 .cshrc
8 .emacs
4 .emacs.d/
4 .gitconfig
4 .gnupg/
4 .lessht
4 .local/
4 .sane/
4 .ssh/
4 .tcshrc
4 .viminfo
4 .vmware/
4 .wget-hsts
4 .xauthgqf83u
4 .xauthNb8jRf
4 .Xauthority
```

図2.10 : /rootディレクトリ

このディレクトリ（発音は「スラッシュ・ルート」）は、rootユーザーのホームディレクトリです。

このディレクトリを所有するrootアカウントは、スーパーユーザー権限を必要とする操作にのみ使用する必要があります。非特権ユーザーとして実行できる操作には、別のアカウントを使用してください。

2.17 /sbin

/sbin

- システムおよびメンテナンスに不可欠なユーティリティに使用されます。
- このディレクトリにあるプログラムは、一般的に特権ユーザが使用するものです。
- **usr**の起動とマウントに使用されるこのディレクトリの実行ファイル（**/binの実行ファイルに加えて**）。
- システムリカバリーを行うための実行ファイル（**/binにあるものに加え、このディレクトリにある実行ファイル**）

/sbinと/usr/sbinです。

前述したように、最近のディストリビューションでは、分離する戦略を放棄しているものもあります。
/sbinと/usr/sbin（および/binと/usr/bin）、そしてシンボリックリンクで1つのディレクトリを持つだけで、2つのディレクトリの表示を維持することができます。

このディレクトリには、起動、復元、回復、および/または修復に不可欠なバイナリが、以下のバイナリに加えて含まれています。
/bin ディレクトリにマウントする必要があります。また、起動時にルートファイルシステムが健全であることが分かれば、必要に応じて/usr、/homeなどの他のファイルシステムをマウントできるようにしなければならない。

このディレクトリには、以下のプログラムが含まれている必要があります（それらのサブシステムがインストールされている場合）：

fdisk、fsck、getty、halt、ifconfig、init、mkfs、mkswap、reboot、route、swapon、swapoff、update。

\$ ls -l /sbin

```
rw-rwxrwx 1 root 8 Apr 23 08:23 /sbin -> usr/sbin
```


/sbin の例

```

student@ubuntu:~$ ls /sbin
acpi_available  fscck.minix      lvmxchange      ntfsundelete    stop
agetty         fscck.msdos      lvm              on_ac_power     sulogin
alsa           fscck.nfs        lvmchange       osd_login       swaplabel
apm_available  fscck.vfat       lvmconf         pam_extrousers_chkpwd  swapoff
apparmor_parser fscck.xfs        lvmconfig       pam_extrousers_update  swapon
badblocks      fsfreeze         lvmdiskscan     pam_tally       switch_root
blkdiscard     fstab-decode     lvndump         pam_tally2      sysctl
blkid          fstrim          lvmetad         parted          tc
blockdev       gdisk           lvmpolld        partprobe       telinit
brctl          getcap          lvmsadc         pccardctl       tftp
bridge         getpcaps        lvmsar          pivot_root      tune2fs
brltty         getty           lvreduce        plipconfig      u-d-c-print-pci-ids
brltty-setup   halt            lvremove        plymouthd       udevadm
capsh          hdparm          lvrename        poweroff         umount.nfs
cfdisk         hwclock         lvresize        pvchange         umount.nfs4
cgdisk         ifconfig        lvs             pvck            umount.udisks2
cgmanager      ifdown          lvscan          pvcreate        unix_chkpwd
cgproxy        ifquery         MAKEDEV         pvdisplay       unix_update
chcpu          ifup            mdadm           pvmove          upstart
coldreboot     init            mdmon           pvremove        upstart-dbus-bridge
crda           initctl         mii-tool        pvresize        upstart-event-bridge
cryptdisks_start insmod          mkdosfs         pvs             upstart-file-bridge
cryptdisks_stop installkernel   mkfs             pvscan          upstart-local-bridge
cryptsetup     ip              mkfs.bfs        quotacheck      upstart-socket-bridge
cryptsetup-reencrypt ipset           mkfs.cramfs     quotaoff        upstart-udev-bridge
ctrlaltdel     ipset           mkfs.ext2       rarp            ureadahead
debugfs        ipset           mkfs.ext3       raw             veritysetup
depmod         ipset           mkfs.ext4       reboot          vgcfgbackup
dhclient       iptables        mkfs.ext4dev    regdbdump       vgcfgrestore
dhclient-script iptables-restore mkfs.fat         reload          vgchange
dmeventd       iptables-save  mkfs.minix      request-key     vgck
dmsetup        iptables       mkfs.msdos      resize2fs       vgconvert
dosfsck        iptunnel       mkfs.ntfs       resolvconf      vgcreate
dumpe2fs       isosize        mkfs.vfat       restart         vgdisplay
e2fsck         iwconfig       mkfs.xfs        rmmod           vgexport
               lvmxchange     ntfsundelete    stop            vgextend

```

図2.11 : /sbinディレクトリ : Ubuntu 18.04

RHEL、CentOS、Fedora、Ubuntuの最近のディストリビューションバージョンでは、/sbinと/usr/sbinがシンボリックリンクされているので、実際には同じです。

上記のスクリーンショットはUbuntu 18.04のもので、それ以降のバージョンでは、リンクされていない別のディレクトリが存在しなくなりました。

2.18 /srv

/srv

- サイトスペシフィックバリアブルデータ
- サーバー（http、ftpなど）用のデータ、スクリプト。
- バージョン管理システム用リポジトリ
- パッケージは、インストール時にそこにファイルを置くべきではありません

FHSによると：

/srv には、本システムが提供するサイト固有のデータが含まれます。

これは、ユーザーが特定のサービスのデータファイルの場所を見つけることができるようにするためと、読み取り専用データ、書き込み可能データ、スクリプト（cgiスクリプトなど）を1つのツリーにまとめる必要があるサービスを合理的に配置するためのものです。

srvのサブディレクトリに名前を付ける方法は、現在のところコンセンサスが得られていないため、特定されていない。srvの下のデータを構造化するための1つの方法は、例えば、ftp、rsync、www、cvsのようなプロトコルによるものです。

システム管理者（およびディストリビューション）の中には、**/srv**の使用を誓う人もいれば、無視する人もいます。**srvではなく、/var**に入れるべきものについては、しばしば混乱が生じます。

UbuntuやRed Hat系などのLinuxディストリビューションでは、デフォルトで**/srv**は空になっています。

2.19 /tmp

/tmp

- 一時ファイルの保存に使用
- ここに保存されたファイルが削除されることを想定している
 - **Ubuntu**はシステム再起動ごとに/tmpを消去します！
- メモリ上に/tmpを疑似ファイルシステムとしてマウントするシステム（**Fedora**など）には注意が必要です。
 - でストレージ上の実ファイルシステムに変換する：
`$ sudo systemctl mask tmp.mount`
および再起動

一時ファイルを保存するために使用され、どのユーザーやアプリケーションからもアクセスすることができます。ただし、/tmpにあるファイルは、長期間保存されることを期待することはできません：

- 一部のディストリビューションでは、パーシスクリプトがそれらを除外するように修正されていない限り、通常10日以上前のファイルを削除する自動クローンジョブを実行しています。
- ディストリビューションによっては、リブートするたびに/tmpの内容を削除するものがあります。これは**Ubuntu**の方針となっています。
- 最近のディストリビューションでは、/tmp ディレクトリを使用する仮想ファイルシステムを、**tmpfs** ファイルシステムを使用する**ラムディスク**のマウントポイントとしてのみ利用するものがあります。これは、**Fedora** システムのデフォルトのポリシーです。システムがリブートすると、すべての情報はそれによって失われます！

最後のケースでは、/tmpに大きなファイルを作成することは避けなければなりません。実際にディスクではなくメモリの領域を占有することになり、メモリの枯渇によってシステムに損害を与えたりクラッシュしたりすることが容易になります。アプリケーションは /tmp に大きなファイルを置かないようにするのがガイドラインですが、このポリシーに違反して /tmp に大きな一時ファイルを作成するアプリケーションはたくさんあります。

Fedoraなどの**systemd**を使用しているシステムでは、コマンドを実行することでこのポリシーを解除することができます：

```
$ sudo systemctl mask tmp.mount
```

その後、システムの再起動を行います。

2.20 /usr

/usr

表2.3 /usr以下のディレクトリー

/usr/bin	非必須コマンドバイナリ
/usr/etc	不要な設定ファイル（通常は空です。）
/usr/games	ゲームデータ
/usr/include	アプリケーションのコンパイルに使用するヘッダーファイルです。
/usr/lib	ライブラリファイル
/usr/lib64	64ビット用ライブラリファイル
/usr/local	第3階層（マシンローカルファイル用）
/usr/sbin	非本質的なシステムバイナリ
/usr/share	読み取り専用のアーキテクチャ独立ファイル
/usr/src	Linuxカーネルのソースコードとヘッダ。
/usr/tmp	セカンダリーテンポラリーディレクトリ

usrディレクトリは、**2次的な階層と考えることができ**、システムの起動に必要なファイルに使用されます。実際、/usrはルートディレクトリと同じパーティションに存在する必要はなく、同じシステムアーキテクチャを使用するホスト間でネットワークを介して共有することができます。

ソフトウェアパッケージは、/usrの直下にサブディレクトリを作成するべきではありません。互換性のために、他の場所へのシンボリックリンクが存在する場合があります。

このディレクトリは、通常、読み取り専用のデータです。シングルユーザーモードでは不要なバイナリが含まれています。このディレクトリには

ローカルバイナリなどが格納される/usr/localディレクトリ。マンページは/usr/share/manに格納されます。



非常に重要

最近のディストリビューションでは、/bin、/sbin、/lib、/lib64を/usrの下の特ナから分離し、シンボリックリンクを使って結合するという戦略を放棄しているものもあります。彼らは、/usrを別パーティションに置き、起動後にマウントできるようにするという、古くからあるコンセプトを時代遅れとみなしている。

2.21 /var

/var

- 可変データファイルが含まれます：リードオンリーでマウントすることはできません

- ログファイル
- スプールディレクトリとファイル
- 行政データファイル
- キャッシュコンテンツなどの一時的・一時的なファイル

表2.4 /var以下のディレクトリー

ディレクトリ	目的
/var/ftp	ftpサーバーのベースに使用します。
/var/lib	プログラムの実行に伴って変更される永続的なデータ。
/var/lock	リソースへの同時アクセスを制御するために使用されるロックファイル。
/var/log	ログファイルです。
/var/mail	ユーザーのメールボックスです。
/var/run	前回のブート以降のランニングシステムに関する情報。
/var/spool	プリントキューなど、スプールされたタスクや処理待ちのタスク。
/var/tmp	システム再起動時に保存される一時ファイル。 <code>tmp</code> にリンクされることもあります。
/var/www	ウェブサイトの階層をルート化する。

このディレクトリには、システム運用中に頻繁に変更される可変（または揮発）データファイルが含まれています。明らかに、`/var`は読み取り専用ファイルシステムとしてマウントすることはできません。セキュリティ上の理由から、`/var`は別のファイルシステムとしてマウントするのが良いとされることが多い。さらに、このディレクトリがいっぱいになっても、システムがロックされることはないはずです。

`/var/log`は、ほとんどのログファイルが配置されています。`/var/spool`は、メール、印刷、cronジョブなどの処理のためのローカルファイルが、処理待ちの間に保存される場所です。

```

student@debian: ~
File Edit View Search Terminal Help
student@debian:~$ ls -lF /var
total 48
drwxr-xr-x  2 root root  4096 May 26 12:23 backups/
drwxr-xr-x 14 root root  4096 Jan 16 2016 cache/
drwxr-xr-x  2 root root  4096 Nov 24 2015 crash/
drwxr-xr-x  2 root root  4096 Apr 26 2015 games/
drwxr-xr-x 62 root root  4096 May  1 11:21 lib/
drwxrwsr-x  2 root staff 4096 Nov 30 2014 local/
lrwxrwxrwx  1 root root    9 Apr 26 2015 lock -> /run/lock/
drwxr-xr-x 14 root root  4096 May 26 12:24 log/
drwxrwsr-x  2 root mail  4096 May 26 12:24 mail/
drwxr-xr-x  2 root root  4096 Apr 26 2015 opt/
lrwxrwxrwx  1 root root    4 Apr 26 2015 run -> /run/
drwxr-xr-x  7 root root  4096 Oct 17 2016 spool/
drwxrwxrwt 14 root root  4096 May 26 12:19 tmp/
drwxr-xr-x  3 root root  4096 Nov  2 2015 www/
student@debian:~$

```

図 2.12: /var ディレクトリ

2.22 /run

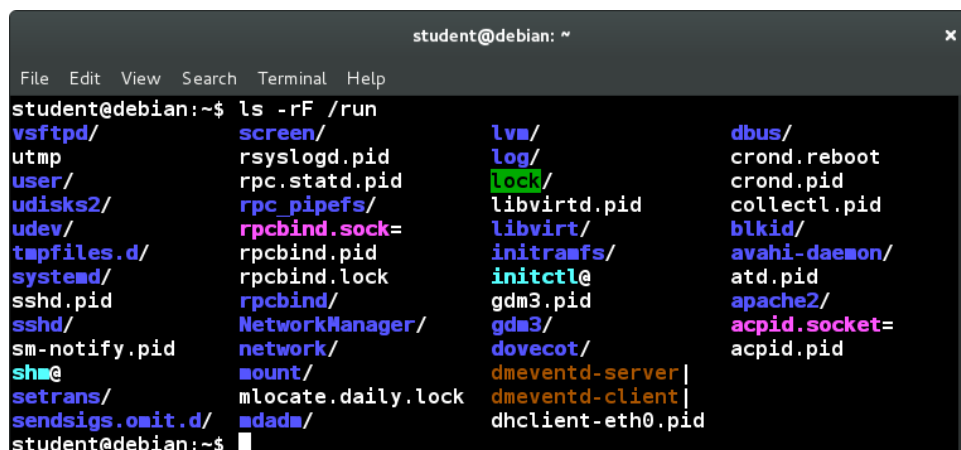
/run

- 疑似ファイルシステム、メモリ上にしか存在しない
- 一時的なランタイムファイルとして使用
- リムーバブルメディアのマウントによく使われる

`run`の目的は、一時的なファイルを保存することです。ランタイム情報を含み、システム起動の初期に書き込む必要があり、再起動時に保存する必要がないものです。

一般に`/run`は空のマウントポイントとして実装され、実行時にそこに`tmpfs`のラムディスク（`/dev/shm`など）がマウントされます。したがって、これはメモリ上にのみ存在する疑似ファイルシステムである。

`var/run`や`var/lock`のような既存の場所は、`/run`下のディレクトリへのシンボリックリンクになるだけです。その他の場所も、ディストリビューションの好みに応じて、`/run`以下の場所を指すだけかもしれません。



```
student@debian: ~
File Edit View Search Terminal Help
student@debian:~$ ls -rF /run
vsftpd/      screen/      lvm/         dbus/
utmp         rsyslogd.pid log/          crond.reboot
user/        rpc.statd.pid lock/         crond.pid
udisks2/     rpc_pipefs/  libvirtd.pid collectl.pid
udev/        rpcbind.sock libvirt/     blkid/
tmpfiles.d/  rpcbind.pid  initramfs/   avahi-daemon/
systemd/     rpcbind.lock initctl@     atd.pid
sshd.pid     rpcbind/     gdm3.pid     apache2/
sshd/        NetworkManager/ gdm3/        acpid.socket=
sm-notify.pid network/      dovecot/     acpid.pid
sh@          mount/       dmeventd-server|
setrans/     mlocate.daily.lock dmeventd-client|
sendsigs.omit.d/ dad/         dhclient-eth0.pid
```

図 2.13: `/run` ディレクトリ

2.23 研究室



ビデオデモンストレーションのリソース

[linux_distros_demo.mp4](#)
[using_disk_usage_demo.mp4](#)
[sing_disk_usage_demo.](#)

✍ 練習問題2.1 : Linuxのデフォルトのディレクトリの大きさ

duユーティリティを使用して、システムのトップレベル・ディレクトリの全体のサイズを計算します。コマンドを入力します：

```
$ du --help
```

は、この結果を効率よく取得し表示するためのヒントを提供します。

✓ ソリューション2.1

サイズとともに、/**の下にあるディレクトリ**の全リストを取得すること：

```
$ sudo du --max-depth=1 -hx /.
```

```
4.3M    /ホーム
16K     /ロスト+ファウンド
39M     /エトセトラ
4.0K    /srv
3.6M    /root
178M    /オプト
138M    /boot
6.1G    /usr
1.1G    /var
16K     /mnt
4.0K    /media
869M    /tmp
8.4G    /
```

オプションを使用したところ：

- `--maxdepth=1` : **/から**1つ下の階層に行き、ツリーで再帰的に下にあるものをすべて合計するだけです。
- `-h`: 人間が読める数字 (KB、MB、GB) を与える。
- `-x` 1つのファイルシステムに留まり、/**パーティション**にないディレクトリは見ない。この場合、無視することを意味します：

```
/dev /proc /run /sys
```

というのも、これらはメモリ上にのみ存在する擬似ファイルシステムであり、システムが動作していないときには、空のマウントポイントに過ぎないからです。これは**RHEL**システム上で行われたため、以下のマウントポイントもフォローされていません：

```
/bin /sbin /lib /lib64
```

というのも、これらは/**usr**下の対応するものとシンボリックにリンクされているだけだからです。

✍ 練習問題2.2: procファイルシステムの見学



ご注意ください

この演習で表示される内容は、カーネルのバージョンに依存するため、表示される出力と正確に一致しない場合があります。

1. rootで、`/proc`にcdして、ディレクトリのリストを作成します。これで、多くのファイルとディレクトリが表示されるはずです

:

```
$ cd /proc
```

```
$ ls -F
```

1/	128/	1510/	20/	2411/	30895/	53/	6925/	802/	951/	キロムスグ
10/	129/	1511/	2015/	2425/	31/	54/	7/	81/	952/	クパゲックグル
										ープ
1002/	13/	1512/	2022/	2436/	31449/	55/	70/	813/	957/	クパゲカウント
1007/	130/	1513/	2023/	2444/	32/	56/	702/	814/	97/	クページフラ
										ッグス
10540/	131/	1514/	20300/	2451/	33/	58/	709/	816/	9742/	ローダブリュ
										ーグ
10590/	13172/	152/	20354/	2457/	34/	585/	71/	817/	98/	錠前
10798/	132/	15552/	20380/	2489/	35/	59/	718/	82/	99/	メインフォ
10805/	133/	15663/	20388/	25/	36/	60/	719/	83/	9923/	ミスク
10806/	134/	15737/	20392/	2503/	37/	61/	72/	834/		モジュール
10809/	135/	159/	20396/	2504/	374/	6193/	721/	835/		マウントズアッ
										ト
10810/	136/	15981/	2086/	2531/	379/	62/	723/	84/		マートレル
10813/	137/	16/	2090/	2546/	38/	63/	725/	841/		ネットアット
10894/	138/	162/	211/	2549/	380/	634/	727/	842/		ページテーブ
										インフォ
10925/	1384/	1632/	22/	2562/	40/	64/	73/	85/		パーティシ
										ョン
10932/	1385/	1636/	2205/	25794/	41/	65/	7300/	857/		sched_debug (
										スケードデバッ
										グ
10934/	1387/	166/	2209/	26/	42/	662/	74/	86/		コンソール
10935/	139/	1670/	2212/	2610/	43/	663/	757/	864/		クプインフ
										ォ
10941/	1390/	17/	2232/	26108/	44/	665/	758/	867/		暗号
										スラブイン
10983/	1393/	17271/	2238/	2619/	4435/	666/	76/	87/		フォ
										ソフトアー
10998/	14/	17361/	2296/	2624/	45/	67/	761/	88/		ク
										スタット
11/	140/	1793/	2298/	2627/	46/	670/	762/	881/		ディスクス
11047/	1410/	18/	23/	2644/	468/	671/	765/	886/		タッツ
										ディーエムエ
1105/	1415/	1831/	23042/	2645/	47/	673/	766/	887/		スワップス
										システムズ
										ー/
										エクスキュー
										ブド
1121/	1429/	18880/	2344/	2679/	470/	674/	768/	888/		シスルック
1123/	1437/	18903/	2348/	27/	484/	678/	769/	889/		トリガー
1135/	1445/	19/	2353/	2706/	49/	679/	77/	89/		sysvipc/
11420/	146/	19392/	2354/	2762/	492/	68/	771/	9/		スレッドセル
										フ@
										timer_list
										タイマース
										タッツ
11499/	1463/	19488/	2365/	28/	493/	682/	78/	90/		アイオム
11515/	147/	1954/	23683/	2858/	5/	683/	79/	92/		アイオポー
										ツ
11530/	1476/	1963/	2370/	28730/	50/	686/	793/	921/		アイアー
										ルシー
1163/	148/	19727/	2372/	28734/	51/	687/	794/	928/		バージョン
1164/	1485/	19734/	2374/	29/	510/	69/	8/	930/		バイエムオー
										シーインフォ
12/	149/	19984/	24/	2973/	514/	690/	80/	931/		筋金入り
										バイエムスタ
										ット
										ゾーンインフ
										ォ

ディレクトリ名の多くが数字であることに注意してください。それぞれが実行中のプロセスに対応し、名前は**プロセスID**です。後ほど説明する重要なサブディレクトリは `/proc/sys` で、この下で多くのシステムパラメータを調べたり変更したりすることができます。

2. 以下のファイルを表示します：

- `/proc/cpuinfo` です：
- `/proc/meminfo` です：
- `/proc/mounts` です：
- `/proc/swaps` です：
- `/proc/version` です：
- `/proc/partitions` です：
- `/proc/interrupts` です：

名前を見れば、どんな情報を公開しているのかがよくわかる。

なお、これらの情報は常に更新されているわけではなく、見たいと思ったときにだけ得られるものです。

3. 任意のプロセスディレクトリを覗いてみてください（自分の所有するプロセスでない場合は、**sudo**を使用しない限り、情報の一部が制限される場合があります）：

\$ ls -F 4435

attr/	コレダンプフィルタ	gid_map	マウントインフォ	sessionid	システム
ントス	oom_score_adj	autogroup	咬合器	setgroups	コールタ
オーケーブ	ページマップ		イオ	smaps	スク/
ソナリティ	cwd@	際限	マウントスタツパー	stack stat	timerslack_ns
cggroup		エンバイロメント	ログインID	statm	uid_map
ネット		projid_map	clear_refs	status	
exe@map_files/	ns/	ルートアット			

cmdline、cwd、environ、mem、statusのようなフィールドを見てみましょう。

第3章

ユーザー環境



3.1	環境変数	58
3.2	キーショートカット	62
3.3	コマンド履歴	64
3.4	コマンドエイリアス	68
3.5	研究室	69

学習目標

このセッションが終わるころには、次のことができるようになっているはずです：

- **環境変数**の理解、使用、設定、変更、およびエクスポート。
- ショートカットキーを使って、画面の消去、プロセスのサスペンドやキルなどを行うことができます。
- 過去のコマンド履歴を調べ、以前のコマンドを取り出し、編集し、使用する。
- よく使うコマンドやオプションに**別名**を設定する。

57

3.1 環境変数

環境変数

- 名前と変数値の関連付け

ホー
ム
ス
ト
パス

- と異なるフォーマットで記載することができます：

\$ env
\$ エクスポート
\$ セット

- すべての変数の参照時には、先頭に\$が付きます：

\$ echo PATH = \$PATH

- ただし、定義されているときは別です：

MYCOLOR=blue

等号の周りにスペースがないことに注意してください！

`env`、`set`、`printenv`で環境変数を列挙すると、その数が多いことがわかる。多くのアプリケーションやプログラム言語が環境変数を利用し、それを期待し、存在しない場合は失敗します。Linuxでは、システム起動時に環境変数を設定し、様々なことに利用しています。自分の目的のために自分で作り、操作することができる。

`env | head -2`

```
XDG_VTNR=1
XDG_SESSION_ID=c259
```

`セット | ヘッド -2`

```
ABRT_DEBUG_LOG=/dev/null
BASH=/usr/bin/bash
```

`エクスポート | ヘッド -2`

```
declare -x BITS="64"
declare -x CCACHE_COMPRESS="1"
```

`$ echo $PATH`

```
/usr/local/texlive/2020/bin/x86_64-linux/:/sbin:/usr/lib64/ccache:/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/home/bjmoose/bin
```

`エコー $HOME`

```
/home/bjmoose
```

いくつかの重要な環境変数

- HOME
 - ユーザーのホームディレクトリ
 - **cd**でホームディレクトリに到達できるようにする
- PATH
 - 実行するプログラムを検索するためのディレクトリの順序付きリスト
 - コロンで区切られたディレクトリー
- ピーエスワン
 - コマンドラインプロンプト、カスタマイズが容易
- シェル
 - ユーザーのデフォルトシェル（**bash**、**csh**など）。
- エディター
 - ユーザーのデフォルトエディター（**emacs**、**vi**など）。

エコー \$HOME

```
/home/rjsquirrel
```

\$ echo \$PATH

```
/usr/lib64/ccache:/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/bin:/sbin:/home/rjsquirrel/bin
```

\$ echo \$PS1

```
\PWD>を使します。
```

エコー \$SHELL

```
/bin/bash
```

エコー \$EDITOR

```
/usr/bin/emacs
```

環境変数の設定

- 特定の変数をリストアップする：
`エコー $SHELL`
- 新しい変数値を設定する：
`$ VARIABLE=value`
- 新しい変数を恒久的に追加する：
 - `VARIABLE=value`を含むように `〜/.bashrc` を編集する。
 - 新しいシェルを起動する、またはログアウト/ログインする

`エコー $SHELL`

```
/bin/bash
```

`エコー $EDITOR`

```
/usr/bin/emacs
```

`$ export EDITOR=/usr/bin/nano`

`エコー $EDITOR`

```
/usr/bin/nano
```

`エコー $MYVAR`

`$ vi .bashrc` <-- "export MYVAR=myvalue "を追加。

`エコー $MYVAR`

`$ source .bashrc`

`エコー $MYVAR`

```
マイバリュー
```


環境変数の書き出し

- 環境変数は、デフォルトでは子プロセスに継承されない
- 変数をエクスポートすると、その動作が変更されます：
`VAR=value ;エクスポートVAR`
`$ export VAR=value`
- エクスポートされた変数を一覧表示することができます：
`$ エクスポート`

デフォルトでは、スクリプト内で作成された変数は、現在のシェルでのみ使用可能です：子プロセス（サブシェル）は、この変数の内容にアクセスすることはできません。子プロセス（サブシェル）はこの変数の内容にアクセスできません。変数を子プロセスから見えるようにするには、**export**コマンドを使用して変数をエクスポートする必要があります。変数のエクスポートは、1つのステップで行うことができます：

```
$ export VAR=value
```

または2ステップで行います：

```
VAR=value ;エクスポートVAR
```

子プロセスはエクスポートされた変数を変更することができますが、エクスポートされた変数は共有されずコピーされるだけなので、この場合の変更は親シェルに伝わらないことに留意してください。

```
$ export VERSION=$(uname-r)
```

```
$ エクスポート
```

```
declare -x BITS="64"
declare -x CCACHE_COMPRESS="1"
declare -x CCACHE_DIR="/mp/.ccache"
---
declare -x VERSION="4.0.3"
---
```

3.2 キーショートカット

キーショートカット

- CTRL-L
 - 画面をクリアにする
- シーティーエルディー
 - 現在のシェルを終了する
- シーティーエルゼット
 - 現在のプロセスをサスペンドバックグラウンドにする（**fg**を入力すると元に戻すことができる）
- CTRL-C
 - 現在のプロセスを殺す

CTRL-Lを入力すると、画面がクリアになります。**clear**と入力したのと同じ効果があります。試してみてください！上記の他のショートカットキーの例を紹介します：

\$ bash # 新しいサブシェルに入る

サブシェル：\$ # 実際にはCTRL-Dが入力され、"exit"が表示されまし

スリープ1000 # 1,000秒間スリープさせる

ゼット # 数秒後にCTRL-Zが入力された場合
[1]+停止中 sleep 1000

\$ fg # フォアグラウンドで "sleep 1000 "の実行を再開する。

スリープ1000

^C # 数秒後にCTRL-Cを入力する

\$ # "sleep "を殺し、シェルを与える

その他のショートカットキー

- CTRL-H
 - バックスペースと同じ
- シーティーエルエー
 - 行頭へ移動する
- CTRL-W
 - カーソルの前の単語を削除する
- タブ
 - ファイル、ディレクトリ、バイナリのオートコンプリート機能

3.3 コマンドヒストリー

沿革

- 以前の履歴を表示するコマンドです：
[ドル歴史](#)
- シェルは履歴を `~/.bash_history` に保存する。
- ヒストリーファイルの場所：HISTFILE
- ヒストリーファイルの最大行数：HISTFILESIZE
- 現在のセッションにおけるヒストリーリストの最大行数：HISTSIZE

履歴ファイルの場所を表示します。

```
$ echo $HISTFILE
```

```
/home/jbond/.bash_history です。
```

ヒストリーファイルの最大サイズを行単位で表示します。

```
エコー $HISTFILESIZE
```

```
2000
```

ヒストリーファイルの現在のサイズ（行数）を表示する。

```
$ echo $HISTSIZE
```

```
1000
```

ヒストリーファイルの最初の2行を表示します。

```
ヒストリー | ヘッド -2
```

```
40 cd RELEASE
41 ls -ltr
```

前のコマンドを呼び出す、編集する

- 前のコマンド/前のコマンドに移動する
 - 上／下矢印キー
- 前のコマンドを実行する：
 - !!
- 過去に使用したコマンドを検索する：
 - CTRL-R
 - 複数回押すと、前のコマンドを後方にスキャンします。

矢印キーは、履歴を上下に移動させるもので、最近実行したコマンドを優先します。

!!(!("バンバン"と発音されることが多い)は、前のコマンドを実行し

ます。以下は、CTRL-Rを使用して履歴を検索する例である：

これはすべて1行で行われる

CTRL-R git

(reverse-i-search)`git': git rm -f exported-variables.inc #

Enterを押して実行するか、もう一度押してください

:

\$ CTRL-R

(reverse-i-search)`git': git ls-files #

Enterを入力すると実行されます。

ヒストリーを使ったこれまでのコマンド

- 履歴の置き換えを開始する：

!

- 行の最後の引数を参照する：

!\$

- n番目のコマンドラインを参照する：

!n

- 文字列で始まる最も新しいコマンドを参照する

!"文字列"です。

履歴の置換はすべて「！」から始まります。したがって、コマンドの中で

```
$ ls -l /bin /etc /var
```

!`$` は、上の行の最後の引数である `/var` を指します。

ドル履歴

```
1 エコー $SHELL
2 エコー $HOME
3 エコー $PS1
4 ls -a
5 ls -l /etc/passwd
6 スリープ1000
7 歴史
```

`$ !1` # 上のコマンド#1を実行する

```
エコー $SHELL
/bin/bash
```

`$ !sl` # "sl"で始まるコマンドを実行する

```
スリープ1000
```

3.4 コマンドエイリアス

エイリアス

- エイリアスを作成し、カスタマイズされたコマンドを作成

```
$ alias name=command
```

等号の周りにスペースがないことに注意してください

- 永続化するには、`~/.bashrc`を編集します。
- のように、標準的なプログラムの動作を修正するのに優れています：

```
$ alias rm='rm -i' です。
```

- エイリアスを取り除く：

```
$ unalias name
```

- 現在有効なエイリアスをリストアップします：

```
$ エイリアス
```

エイリアスは、カスタム定義が可能です。引数なしで**alias**と入力すると、定義されたエイリアスのリストが表示されます。unaliasはエイリアスを削除します。

いくつかの例を挙げます：

```
$ alias l='ls -laF' です。
```

```
$ alias dir='ls -latF' です。
```

```
$ alias rm='rm -i' です。
```

```
$ alias mv='mv -i' とする。
```

```
$ alias cp='cp -ipdv' です。
```

```
$ alias df='df -T' です。
```

```
$ alias gitstat='ギットステータス -uno'
```

```
$ alias diffside='diff --side-by-side --ignore-all-space'
```

```
$ alias scapt='gnome-screenshot -i -w'
```


3.5 研究室

練習問題3.1：勘定科目

どのアカウントでログインしているのか？どうやって確認するのですか？

解決策3.1

whoamiユーティリティを実行します：

```
$ whoami
学生
```

練習3.2：/tmpディレクトリをパスに追加する

行だけを含む小さなファイル、/tmp/lsを作成します：

```
echo HELLO、こちらはニセモノのlsプログラムです。
```

して実行可能な状態にします：

```
$ chmod +x /tmp/ls
```

1. パスに/tmpを追加し、通常のパスが考慮された後にのみ検索されるようにします。lsと入力し、どちらのプログラムが実行されるかを確認します：lsと入力して、/bin/lsと/tmp/lsのどちらのプログラムが実行されるかを確認してください。
2. 通常のパスが考慮される前に、/tmpが検索されるように、/tmpをあらかじめパスに追加しておきます。もう一度、lsと入力して、どちらのプログラムが実行されるかを確認します：/bin/lsと/tmp/lsのどちらが実行されるかを確認してください。

このようにパスを変更する場合、セキュリティ上どのような配慮が必要でしょうか？

解答 3.2

まず、エディタを使って、あるいは単に実行して、ニセのlsプログラムを作成します：

```
$ echo "エコーHELLO、これはニセのlsプログラムです。" > /tmp/ls
$ chmod +x /tmp/ls
```

次の2つのステップでは、別のターミナルウィンドウで作業するか、新しいシェルを立ち上げるとよいでしょう。bashと入力すれば、新しいシェルを起動することができます。

1. \$ bash
\$ PATH=\$PATH:/tmpとする。

```
bin etc ゲーム include lib lib64 libexec ローカル sbin シェア src
```

```
テ
```

```
$ 出口
```

2. \$ bash
PATH=/tmp:\$PATH
\$ ls /usr

```
HELLO、これはニセモノのlsプログラムです。
```

\$ 出口

もし誰かが悪意のあるプログラムを/tmpに置くことができれば、あなたを騙して誤って実行させることができます。

演習3.3: コマンドヒストリー

あなたはLinuxワークステーションで忙しく働いていて、ある特定のbashコマンドシェルで約100個のコマンドを入力したことがあります。

以前、ある時点で新しいコマンドを使ったが、正確な名称は忘れてしまった。

あるいは、オプションや引数がたくさんあるかなり複雑なコマンドで、もう一度何を入力すればいいのか、エラーの起きやすいプロセスを経たくないのかもしれない。

コマンドの内容はどうやって確認するのですか？

履歴でコマンドを見つけたら、プロンプトで全部入力することなく、簡単にもう一度コマンドを発行するにはどうしたらいいのでしょうか？

解決策 3.3

コマンドの履歴です：

履歴コマンドは、自分が入力したコマンドを表示する方法です：

ドル歴史

```
1 cd /
2 ls
3 cd
4 プードル
5 エコー $SHELL
6 ls /var/
7 ls /usr/bin
8 ls /usr/local/bin
9 マニュアルファスタブ
10 ls
```

以前のコマンドを再実行するためには、いくつかの選択肢があります。例えば、最初にログインしたときに実行したmanコマンドを再実行したいとします。その場合、次のように入力します。

\$!9

と入力すると、#9として表示されているコマンドを再実行することができます。もし、入力したmanコマンドがこれだけだった場合は、次のように入力することもできます。

\$!man

のように、入力したコマンド名を覚えておくとよいでしょう。最後に、いくつかのmanコマンドを入力した場合、CTRL-Rを使って履歴を遡って検索し、再実行したい特定のmanコマンドを見つけ、Returnを押すだけで実行することができます。

練習問題3.4: コマンドエイリアス

長いコマンドやファイル名を何度も入力するのは面倒ですし、タイプミスなどの些細なミスも多くなります。エイリアスを導入することで、このような入力の手間を軽減するためのショートカットを定義することができます。

あなたがプロジェクトチームのメンバーで、プロジェクトのための共通の共有ディレクトリで作業しているとします。このディレクトリは、/home/staff/RandD/projects/projectX/srcに配置されています。

プロジェクトXの作業をしていると、このディレクトリにファイルを作成したり、修正したりする必要があることがよく

あります。入力するまでにあまり時間がかかりません：

```
$ cd /home/staff/RandD/projects/projectX/src
```

は面倒くさくなる。

上記の`cd`コマンドを実行するために、「projx」というエイリアスを定義して使用します。

✔ 解答 3.4

コマンドエイリアス：

エイリアスラインは次のようになります：

```
$ alias projx='cd /home/staff/RandD/projects/projectX/src' です。
```

なお、一重引用符の代わりに二重引用符を使用することもできますし、定義したエイリアスに空白がないため、引用符を全く使用しないことも可能です。

あとは、このディレクトリに変更するだけです。

```
$ projx
```

エイリアスを永続化するには、`$HOME/.bashrc` ファイルに配置するだけです。

第4章

ユーザーアカウント管理



4.1	ユーザーアカウント	74
4.2	シェルの起動ファイル	77
4.3	ユーザーアカウントの管理	80
4.4	ロックされたアカウント	82
4.5	パスワード	83
4.6	/etc/shadow	84
4.7	パスワード管理	86
4.8	パスワードエイジング	87
4.9	ルートアカウント	88
4.10	SSH	90
4.11	研究室	97

学習目標

このセッションが終わるころには、次のことができるようになっているはずです：

- 個々のユーザーアカウントの目的を説明し、その主な属性を列挙する。
- 新規ユーザーアカウントの作成、既存アカウントのプロパティの変更、アカウントの削除やロックが可能です。
- ユーザーパスワードの設定、暗号化、保存方法を理解し、セキュリティのためにパスワードの変更を長期的に要求する方法を理解している。
- rootアカウントの役割と使用するタイミングを理解する。
- Secure Shell (**ssh**) を使用し、ログインとコマンドを削除します。

73

4.1 ユーザーアカウント

ユーザーアカウントの目的

- ユーザー一人ひとりに、自分だけのプライベート空間を提供する。
- 特定の専用目的のために、特定のユーザーアカウントを作成すること。
- ユーザー間で権限を区別すること。

Linuxシステムは、マルチユーザー環境を提供し、人とプロセスが別々の同時作業環境を持つことを可能にします。

特別なユーザーアカウントは、システム上で**何でもできるルート**ユーザー用です。高価な間違いを避けるため、またセキュリティ上の理由から、rootアカウントは絶対に必要な場合にのみ使用する必要があります。

通常のユーザーアカウントは、システムで作業する人のためのものです。一部のユーザーアカウント（デーモンアカウントなど）は、プロセスをroot以外のユーザーで実行できるようにする目的で存在します。

また、システム上のユーザーのサブセットが共通の興味に応じてファイルや権限などを共有できる**グループ**管理についても説明します。

ユーザーアカウントの属性

etc/passwdから :

```
-----  
beav:x:1000:1000:セオドア・クリーバー:/home/beav:/bin/bash  
warden:x:1001:1001:Ward Cleaver:/home/warden:/bin/bash  
dobie:x:1002:1002:Dobie Gillis:/home/dobie:/bin/bash  
-----
```

- ユーザー名
- ユーザーパスワード
- ユーザー識別番号 (UID)
- グループ識別番号 (GID)
- コメントまたはGECOS情報
- ホームディレクトリ
- ログインシェル

システム上の各ユーザーは、基本的なアカウント属性を記述した `/etc/passwd` ファイルに対応する行を持っています (このファイルだけでなく、パスワードについても後ほど説明します)。ここでは、7つの要素があります：

1. ユーザー名
各ユーザーに割り当てられた固有の名前です。
2. ユーザーパスワード
各ユーザーに割り当てられたパスワードです。
3. ユーザー識別番号 (UID)
ユーザーアカウントに割り当てられた固有の番号。UIDは、ユーザー権限の決定やアクティビティの追跡など、さまざまな目的でシステムによって使用されます。
4. グループ識別番号 (GID)
ユーザーのプライマリグループ、プリンシパルグループ、またはデフォルトグループを示す。
5. コメントまたはGECOS情報
連絡先情報(フルネーム、Eメール、オフィス、連絡先番号)のためにコメントフィールドを使用するための定義された方法です。(GECOSの意味は気にしないでください。とても古い言葉です)。
6. ホームディレクトリ
ほとんどのユーザーにとって、これはユーザーのための作業領域を提供するユニークなディレクトリです。通常、このディレクトリはユーザーが所有し、**root 以外**はシステム上の `/home` のどこかにあります。
7. ログインシェル
通常、これは `/bin/bash` や `/bin/csh` などのシェルスクリプトプログラムである。しかし、特別な場合には、別のプログラムがここで参照されることもあります。一般に、このフィールドはあらゆる実行可能ファイルを受け入れる。

カレントユーザーの決定

- Linuxはマルチユーザシステム
- 現在ログインしているユーザーを検索します :
 - どちら様
- 現在のユーザーIDを調べる :
 - ふうきん

例えば、こんな感じです :

だれだれ

ビジュモース 0	2015-03-17 07:55 (:0)
ビジュモース	2015-03-17 07:55 (:0)
点ノ	2015-03-17 07:55 (:0)
0点	2015-03-17 07:55 (:0)
ビジュモース	2015-03-18 15:13 (:0)
プッ	2015-03-18 15:13 (:0)
ツ1	2015-03-18 16:54 (エフビーフ)

ドル

ビジュモース

4.2 シェルの起動ファイル

スタートアップファイル

- 各コマンドシェルは、複数のスタートアップファイルを使用して環境を構築します。
- インタラクティブな環境では、多少異なる影響がある
- `etc`のファイルは、グローバルな設定を提供します
- ユーザーのホームディレクトリにあるファイルは、グローバルな設定を上書きする

新しいシェル（コマンドウィンドウまたは実行するスクリプト）が実行を開始するたびに、適切な機能を確保するために採用される要素を含むファイルが含まれます。これには以下のようなものがあります：

- 多くのプログラムやスクリプトで使用される**環境変数**（`$PATH`を含む）を定義する。
- コマンドやオプションを指定するための省略記法として使用される**エイリアス**を定義する。
- 後続のスクリプトで使用可能な**関数**を定義する。

通常、`/etc`にあるシステム全体のグローバルな初期化ファイルがあり、個別のファイルが使用される前に、通常すべてのユーザーによって使用されます。

どのファイルをどのような順序で使用するかは、これから説明するように、文脈によって異なります。

スタートアップファイルのメリット

- ユーザーのプロンプトをカスタマイズする
- ユーザーの端末の種類を設定する
- コマンドラインのショートカットとエイリアスを設定する
- デフォルトのテキストエディターを設定する

スタートアップ（初期化）ファイルの処理がないと、コマンドやプログラムを実行するたびに、正しく機能するためのセットアップ作業が発生することがあります。

多くのプログラムは、実行開始時に特定の環境変数が設定されていることを評価し、それを利用して機能を制御しています。

例えば、テキストファイルを対話的に修正する必要があるプログラムは、EDITORがどのように設定されているかを確認することができます（おそらくvim、emacs、または

nano;ユーザーが使用するエディターのいずれかを使用します。

環境変数」の項でその例を紹介します。

スタートアップファイルの順番

- 全ユーザーの初期設定値
`/etc/profile`
- ログインシェルの設定
`~/.bash_profile`
- ログイン初期化
`~/.bash_login`
- `/etc/profile`をオーバーライドします。
`~/.profile`
- インタラクティブな非ログインシェルの設定
`~/.bashrc`
- 通常は、`~/.bashrc`をカスタマイズするだけでよいのですが。

Linuxにログインすると、常に`/etc/profile`が読み込まれ評価されます。

次に、以下のファイルがこの順序で検索されます：

`~/.bash_profile`

`~/.bash_login`

`~/.profile`

Linuxのログインシェルは、最初に来たファイルを見つけた後、その1つのスタートアップファイルの評価し、残りのすべてを無視します。

冗長に聞こえるかもしれませんが、様々なLinuxディストリビューションは、異なるスタートアップファイルを使用する傾向があります。

ログインしていない状態でサブシェルを作成すると、毎回、`~/.bashrc`だけが読み込まれ、評価されます。ログインシェルでは読み込まれず評価されませんが、ほとんどのディストリビューションとユーザは、ユーザが所有する3つのスタートアップファイルのうちの1つから`~/.bashrc`を呼び出すので、実際には`~/.bashrc`はログインシェルに使われています。

したがって、カスタマイズの大部分は、`~/.bashrc`に入れる必要があります。

4.3 ユーザーアカウントの管理

useraddでユーザーアカウントを作成する

- useraddでデフォルトの操作が可能
`sudo useradd bjmoose`
 - アカウント作成 bjmoose
 - ユーザーID、グループID、ホームディレクトリ、シェルのデフォルトを使用します。
- useraddのオプションを使用して、デフォルトを上書きする。
`$ sudo useradd -s /bin/csh -m -k /etc/skel -c "Bullwinkle J Moose" bmoose`
- オプションでシェル、skelディレクトリ、コメントフィールドを指定する。

`sudo useradd bjmoose`

は、以下のステップを実行させます：

- デフォルトでUID_MIN (`/etc/login.defs` で指定) より大きい次に利用可能なUIDが、bjmooseのUIDとして割り当てられます。
- GID=UIDのbjmooseというグループも作成され、bjmooseのプライマリグループとして割り当てられる。
- ホームディレクトリ `/home/bjmoose` が作成され、bjmooseが所有します。
- bjmooseのログインシェルは `/bin/bash` になります。
- `/etc/skel` の内容は、`/home/bjmoose` にコピーされます。デフォルトでは、`/etc/skel` には `bash` 用と `X Window` システム用のスタートアップ・ファイルが含まれています。
- `etc/shadow` ファイルのパスワード欄にbjmooseのエントリに対して「！」

デフォルトは、useraddのオプションで簡単に上書きすることができます：

```
$ sudo useradd -s /bin/csh -m -k /etc/skel -c "Bullwinkle J Moose" bmoose
```

ここで、ユーザー属性の一部には、明示的にデフォルト以外の値が与えられている。

ユーザーアカウントの変更と削除

- rootユーザーは`userdel`でユーザーアカウントを削除することができます。

```
$ sudo userdel rjsquirrel
```

rjsquirrelのユーザーアカウントを削除しますが、彼女のホームディレクトリは削除されません。

- ユーザーアカウントは`usermod`で変更可能

```
sudo usermod -L bjmoose
```

bjmooseのアカウントをロックし、ログインできないようにする。

```
$ sudo userdel rjsquirrel
```

`/etc/passwd`、`/etc/shadow`、`/etc/group`からユーザーrjsquirrelへの参照がすべて削除されます。これは、アカウントを削除する一方で、後にアカウントが再確立される可能性があるため、ホームディレクトリ（通常は`/home/rjsquirrel`）を削除しない。`userdel`に`-r`オプションが与えられている場合、ホームディレクトリも消去されます。ただし、削除されたユーザが所有するシステム上の他のファイルはすべて残ります。

`usermod`は、グループメンバーシップ、ホームディレクトリ、ログイン名、パスワード、デフォルトシェル、ユーザーIDなど、ユーザーアカウントの特性を変更するために使用することができます。使い方はとても簡単です。`usermod`は、必要に応じて`/etc`ディレクトリのファイルの変更も行います。

```
student@CentOS7:~$ sudo usermod --help
Usage: usermod [options] LOGIN

Options:
  -c, --comment COMMENT      new value of the GECOS field
  -d, --home HOME_DIR        new home directory for the user account
  -e, --expiredate EXPIRE_DATE set account expiration date to EXPIRE_DATE
  -f, --inactive INACTIVE    set password inactive after expiration
                              to INACTIVE
  -g, --gid GROUP             force use GROUP as new primary group
  -G, --groups GROUPS         new list of supplementary GROUPS
  -a, --append                append the user to the supplemental GROUPS
                              mentioned by the -G option without removing
                              him/her from other groups
  -h, --help                  display this help message and exit
  -l, --login NEW_LOGIN       new value of the login name
  -L, --lock                  lock the user account
  -m, --move-home             move contents of the home directory to the
                              new location (use only with -d)
  -o, --non-unique            allow using duplicate (non-unique) UID
  -p, --password PASSWORD    use encrypted password for the new password
  -R, --root CHROOT_DIR      directory to chroot into
  -s, --shell SHELL          new login shell for the user account
  -u, --uid UID              new UID for the user account
  -U, --unlock                unlock the user account
  -Z, --selinux-user SEUSER   new SELinux user mapping for the user account

[student@CentOS7 ~]$
```

図4.1 : usermodの使用

4.4 ロックされたアカウント

ロックされたアカウント

- **Linux**には、ロックされたシステムアカウントがいくつか同梱されています：
 - bin、daemon、sysなどのアカウント。
 - プログラムを実行することはあるが、ログインのために使用されることはない。
- 他の目的のために、他のロックされたアカウントを作成することができる
 - データベースなど主要なアプリケーションで使われるアカウント
- ロックされたアカウントに有効なパスワードがない
 - 通常、`/etc/shadow`では"!"で表されます。
- ロックされたすべてのアカウントが、無効なシェル（一般的に `/sbin/nologin`）

Linuxには、いくつかのロックされたアカウントが同梱されています。これは、プログラムを実行することはできますが、システムにログインすることはできず、有効なパスワードが関連付けられていないことを意味します。例えば、`/etc/passwd`には次のようなエントリーがあります：

```
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
```

nologinシェルは、ロックされたユーザーがシステムにログインしようとした場合、次のように返します：

```
このアカウントは現在ご利用いただけません。
```

または `/etc/nologin.txt` に保存されているメッセージのようなものです。このようなロックされたアカウントは、システムサービスやアプリケーションによって特別な目的で作成されます。**nologin**シェルで`/etc/passwd`をスキャンすると、システム上のユーザーを確認することができます。のように、特定のユーザーのアカウントをロックすることも可能です：

```
sudo usermod -L bjmoose
```

これは、アカウントはシステム上に残るが、ログインが不可能であることを意味します。ロック解除は、`-U`オプションで行うことができます。慣例として、ユーザーが組織を離れたり、長期休暇に入ったりするたびに、アカウントをロックすることがあります。アカウントをロックするもう一つの方法は、**chage**を使用してアカウントの有効期限を変更することです：

```
$ sudo chage -E 2001-09-11 rjsquirrel
```

実際の日付は、過去であれば関係ありません。

4.5 パスワード

ユーザーIDと/etc/passwdについて

```
beav:x:1000:1000:Theodore Cleaver:/home/beav:/bin/bash
rsquirrel:x:1001:1001:Rocket J Squirrel:/home/rsquirrel:/bin/bash
```

- `/etc/passwd` ファイルには、各ユーザーに1レコード（1行）ずつ、コロン（:）で区切られたフィールドのリストが含まれています：
 - ユーザー名：ユーザー固有の名前
 - password: ハッシュ化されたパスワード（`/etc/shadow`が使用されていない場合）またはプレースホルダー（`/etc/shadow`が使用されている場合は"x"）。
 - UID：ユーザー識別番号
 - GID：ユーザーの一次グループ識別番号
 - comment：コメント欄、通常はユーザーの本名
 - ホームディレクトリ ユーザーのホームディレクトリのパス名
 - shell: ログイン時に起動するシェルの絶対修飾名
- UIDはUID_MIN = 1000からスタートします。

ほとんどのLinuxディストリビューションが採用している慣例は、**UIDが1000未満のアカウントは特別なものとみなされ、システムに属するというもので、通常のユーザーアカウントは1000から始まります。**実際の値はUID_MINとして定義され、以下のように定義されています。

`/etc/login.defs`.

これまでRed Hat系のディストリビューションでは、UID_MIN=1000ではなく500を使用していましたが、**RHEL 7からはより一般的な値である1000が採用されています。**

useraddを使用する際に**UIDが指定されない場合**、システムはUID_MINから始まる使用IDを段階的に割り当てます。

さらに、各ユーザーは**プライマリーグループID**を取得し、デフォルトでは**UIDと同じ番号**になります。これらはUPG（User Private Groups）と呼ばれることもあります。

`etc/passwd`、`/etc/group`、`/etc/shadow`を直接編集するのは良くないので、**usermodなどの適切なユーティリティ**を使用してください。

4.6 /etc/shadow

なぜ/etc/shadowを使うのか？

- ユーザーごとにパスワードのエージングを有効にする
- ハッシュ化されたパスワードの安全性を高めることができます。

`etc/passwd`のデフォルトのパーミッションは644(-rw-r--r--)で、誰でもこのファイルを読むことができます。これは、システムプログラムやユーザーアプリケーションがこのファイルに含まれる情報を読む必要があるためで、残念ながら必要なものです。これらのシステムプログラムは、ユーザーrootとして実行されることはなく、いずれにせよ、rootのみがこのファイルを変更することができます。

特に懸念されるのは、ハッシュ化されたパスワードそのものである。ハッシュ化されたパスワードが`/etc/passwd`にあると、誰でもハッシュ化されたパスワードのコピーを作成し、CrackやJohn the Ripperなどのユーティリティを使って、ハッシュ化されたパスワードから元の平文のパスワードを推測することができます。これは、セキュリティ上のリスクです！

`/etc/shadow`のパーミッション設定は400(-r)です。---、つまりrootのみがこのファイルにアクセスできることを意味します。これにより、このファイルはハッシュ化されたパスワードの収集がより困難になります。

やむを得ない正当な理由がない限り、`/etc/shadow`ファイルを使用する必要があります。

/etc/shadow II

- `/etc/shadow` ファイルには、各ユーザーのレコードが1つ（1行）含まれています。
- 各レコードには、コロン（:）で区切られたフィールドが含まれています。
 - username : ユニークなユーザー名です。
 - パスワード : パスワードのハッシュ化（sha512）された値
 - lastchange: パスワードが最後に変更された1970年1月1日からの 日数
 - mindays : パスワードを変更できるようになるまでの最短日数
 - maxdays: パスワードを変更しなければならない最大日数
 - warn: パスワードの有効期限が切れる前にユーザーに警告を発する日数
 - 猶予 : パスワードの有効期限が切れた後、アカウントが無効化される日数
 - expire : アカウントが無効化される/される日付
 - reserved: 予約フィールド

`/etc/shadow`には、以下のように各ユーザーに1レコード（1行）ずつ入っています：



/etc/shadow

```
daemon*:16141:0:99999:7:::
```

```
-----
```

```
beav:$6$ICZyCnBJH9rmq7P.$RYNm10Jg3wrhAtUnahBZ/mTMg.RzQE6iBXyqaXHvxxbK
TYqj.d9wpoQFuRp7fPEE3hMK3W2gclYhiXa9MIA9w1:16316:0:9999:7:::
```

各レコードのユーザー名は、`/etc/passwd`にあるものと完全に一致しなければならず、また、同じ順序で表示されなければならない。すべての日付は、1970年1月1日（エポック日）からの日数で保存されます。

パスワードハッシュは、文字列「\$6\$」の後に8文字のソルト値が続き、その後に\$と88文字（sha512）のパスワードハッシュが続きます。

4.7 パスワード管理

パスワード管理

- パスワードは**passwd**で変更可能
- ユーザー自身がパスワードを変更できる
- Rootは任意のユーザーパスワードを変更できる

デフォルトでは、パスワードの選択は**pam cracklib.so**によって検査されるため、より良いパスワードの選択を行うことができます。通常のユーザーがパスワードを変更する様子：

`$ passwd`

```
bjmooseのパスワードを変更する
(現在の ) UNIXパスワード : <bjmooseのパスワード>
新しいUNIXパスワード : <bjmooseの新パスワード>
新しいUNIXパスワードを再入力する : <bjmoose's-new-password>
passwd: すべての認証トークンが正常に更新されました。
```

rootがユーザーのパスワードを変更する場合、rootは現在のパスワードの入力を要求されないことに注意してください：

`sudo passwd rjsquirrel`

```
新しいUNIXパスワード : <rjsquirrel's-new-password> 新
しいUNIXパスワードを再入力する : <rjsquirrel's-
password>。
passwd: すべての認証トークンが正常に更新されました。
```

なお、通常のユーザーは、短すぎるパスワードや辞書的な単語を使ったパスワードなど、悪いパスワードを設定することは許されません。しかし、rootにはそれが許されています。

4.8 パスワードエイジング

パスワードエイジング（チャージ）

- チャージの使用により管理される
`chage [-m mindays] [-M maxdays] [-d lastday] /`
`[I inactive] [-E expiredate] [-W warndays] ユーザー`
- 例を挙げます :
`$ sudo chage -I beaver`
`sudo chage -m 14 -M 30 wally`
`$ sudo chage -E 2012-4-1 eddie`
`$ sudo chage -d 0 june`

一般に、パスワードは定期的に変更することが重要だと考えられています。これにより、解読されたパスワードが侵入者にとって有用である時間が制限され、また、未使用のアカウントをロックするために使用することができます。しかし、ユーザーがこのポリシーを煩わしく感じ、常に変化するパスワードを書き留めてしまい、盗みやすくなってしまうという欠点があります。これを管理するユーティリティが**chage**です :

`chage [-m mindays] [-M maxdays] [-d lastday] [-I inactive] [-E expiredate] [-W warndays] user`

```
student@ubuntu: /etc/udev
student@ubuntu:/etc/udev$ chage -l student
Last password change           : Apr 14, 2017
Password expires                : never
Password inactive              : never
Account expires                : never
Minimum number of days between password change : 0
Maximum number of days between password change : 99999
Number of days of warning before password expires : 7
student@ubuntu:/etc/udev$
```

図4.2 : チャージの使い方

chageを使用できるのは、rootユーザーだけです。ただし、例外として、どのユーザーでも**-I**オプションを付けて実行することで、パスワードやアカウントの有効期限がいつになるかを判断することができます。

次回ログイン時に強制的にパスワードを変更させるため :

4.9 ルートアカウント

ルートアカウント

- Rootアカウントでシステムにフルアクセスできる
- ユーザーにルートアクセス権を与えない
- rootアカウントに変更する場合：
ス
- rootでコマンドを実行する場合：
sudo コマンド
- でsudoの設定を行います：`/etc/sudoers` と `/etc/sudoers.d` にあります。
- sudoはsuよりはるかに優先される

rootはあらゆるものにアクセスでき、あらゆることができる、非常に強力なアカウントです。

sudoは、通常のユーザーアカウントに一時的にroot権限を持たせることができます。sudoは、特定のアカウントだけにこの能力を持たせたり、特定のアカウントに特定のコマンドの昇格権限だけを持たせるように設定することができます。詳しくは、`man sudo` を参照してください。

su（エスユーと発音し、スイッチまたは代理ユーザーを意味する）は、ユーザーがそのシェルを終了するまで昇格した特権を許可するサブシェル環境を作成します。そのサブシェルで実行されるすべてのコマンドは、rootユーザーの昇格した権限で実行されます。

ミスが発生します！重要なファイルを削除してしまうような大きなミスを防ぐには、「Skype」を使わないのがベストです。su サブシェルで、昇格特権を必要としないコマンドを実行します。

ルートアカウント

rootアカウントの使用は稀な例外であるべきです

- どうしても必要なときだけrootでログインする
 - ミスをしないようにする
 - 潜在的なセキュリティの脆弱性を回避する
- root権限を取得するために**su**を使用することを義務付ける
 - ルートへの直接ログインを禁止する
 - 誰がrootになったか、監査証跡を設定する
- より安全性を高めるには、**sudo**を使用します。

rootアカウントは、絶対に必要な場合にのみ管理目的で使用し、通常のアカウントとしては決して使用しないでください。間違いは、完全性と安定性、そしてシステムセキュリティの両方において、非常に大きな代償を払うことになりかねません。

デフォルトでは、セキュリティ上の理由から、ネットワーク経由のルートログインは一般に禁止されています。ただし、[/etc/ssh/sshd_config](#) で設定する **ssh** と、[pam securetty.so](#) モジュールと関連する [/etc/securetty](#) ファイルを介した **PAM** (Pluggable Authentication Modules) による **Secure Shell** ログインは許可されます。に記載されているデバイスからのみ、ルートログインが許可されます。

[/etc/securetty](#).

一般に、すべてのルートアクセスは **su** または **sudo** を使用することが推奨されています（root アクセスのすべての監査証跡が、**su** または **sudo** を介して発生します）。

sudo）。**Ubuntu**など一部のディストリビューションでは、デフォルトでrootアカウントに直接ログインすることを禁止していることに注意してください。

PAMは、rootへの**su**を許可されるユーザを制限するために使用することも可能です。また、rootとして実行されたすべてのコマンドを記録するように**auditd**を設定することも価値があるかもしれません。

4.10 SSH

SSH

- **ssh** : リモートシステムにログインし、コマンドを実行する。
- **scp**を含む : リモートシステムとの間でファイルを転送する。
- セキュリティのために強力な暗号を使用 (Secure **S**hell という名称の由来) 。

```
$ ssh remote_computer.com
```

```
$ ssh some_user@remote_computer.com
```

```
$ ssh some_user@remote_computer.com apt update
```

```
$ scp file.txt remote_computer.com:/tmp
```

```
$ scp usr1@rem1.com:/tmp/f.txt usr2@rem2.com:/tmp/nf.txt
```

- 複数のシステムで同時にコマンドを実行する :

```
$ for machines in node1 node2 node3
```

```
do.
```

```
    (ssh $machines some_command &)
```

```
done
```

またはもっと簡単です :

```
$ nssh -vH node1 node2 node3 some_command
```

ネットワークを通じて、同じユーザー名または別のユーザー名で、リモートシステムにログインする必要があることがよくあります。また、リモートマシンとの間でファイルを転送する必要がある。いずれの場合も、傍受されることなく安全に行いたいものです。

SSH (Secure **S**hell) はこの目的のために存在します。強力なアルゴリズムに基づく暗号化を使用します。適切なsshパッケージがシステムにインストールされていれば、sshを使い始めるのに、それ以上の設定は必要ありません。リモートシステムにサインオンするには

```
$ ssh farflung.com
```

student@farflung.com のパスワード: (ここに入力)

farflung.comに学生アカウントがあることを前提に。別のユーザーでログインする場合

```
$ ssh root@farflung.com
```

```
$ ssh -l root farflung.com
```

あるシステムから別のシステムへファイルをコピーすること :

```
$ scp file.txt farflung.com:/tmp
```

```
$ scp file.tex student@farflung.com/home/student
```

```
$ scp -r some_dir farflung.com:/tmp/some_dir
```

(紙面の都合上、パスワードのご請求は省略させていただきました)

のように、**pssh** (**P**arallel **S**SH) ユーティリティを使えば、複数のシステムでコマンドを1回のフェールストロークで実行することができます :

```
$ pssh -viH machine1 machine2 machine3 do_something
```

pssh のすべてのオプションとパスワードの扱い方を理解するために、pssh の man ページを読む必要があるかもしれません。

ssh 設定ファイル

- ホームディレクトリで :
 - **id_rsa**: ユーザーの秘密暗号化キー。
 - **id_rsa.pub**: ユーザーの公開暗号化キー。
 - **authorized_keys** : ログインが許可される公開鍵。
 - **known_hosts** : 過去にログインが許可されたホスト。
 - **config** : 各種オプションを指定するためのファイルです。
- **ssh-keygen**で生成した公開鍵と秘密鍵。

SSHをより使いやすくするために、特にパスワードなしでログインできるように設定することができます。ユーザー固有の設定ファイルは、各ユーザーのホームディレクトリの下に隠された `.ssh` ディレクトリに作成されます：

まず、ユーザは `ssh-keygen` を使って秘密鍵と公開暗号鍵を生成する必要があります：

\$ `ssh-keygen`

公開/非公開のrsaキーペアを生成する。

...

秘密鍵は、決して誰とも共有してはならない。しかし、公開鍵は、パスワード不要のアクセスを許可したいすべてのマシンに与える必要があります。また、自分のマシンにアカウントを持っていて、そのアカウントにパスワード `ドレス` でアクセスすることを許可したい他のユーザーの公開鍵とともに、`authorized_keys` ファイルに追加する必要があります。`ssh`でログインしようとするユーザーの変化を検出すると、システムが警告を発し、アクセスを拒否する機会を提供します。

認可されたキーには、ユーザーとマシンに関する情報が含まれていることに注意してください：

\$ `cat authorized_keys`

ssh-rsa AAAAB3Nza. 学生@デビアン
ssh-rsa AAAAB3Nza. student@fedora

一方、`known_hosts`はコンピュータノードに関する情報のみを含んでいます：

\$ `cat known_hosts`

x7 ecdsa-sha2-nistp256 AAAAE2VjZHNhLXN. M=
192.168.1.200 ecdsa-sha2-nistp256 AAAA. bcs=

`man ssh_config` を実行すると、`ssh` 設定ファイルに入れることができるオプションが表示されます。

SSHコンフィグファイルの優先順位

設定ファイル进行处理する順番：

1. コマンドラインオプション
2. `~/.ssh/config`
3. `/etc/ssh/ssh_config`

ファイルやコンテンツの優先順位は、最初に使用されるマッチです。



`/home/student/.ssh/config`

ホストアップル

ホスト名 `192.168.0.196`

ユーザー `学生ポ`

ート `4242`

IdentityFile `/home/student/.ssh/custom`

ホストアワーズ

ホスト名 `ec2-34-238-135-25.compute-1.amazonaws.com`

ユーザー `ubuntu`

IdentityFile `/home/student/.ssh/cloud1.pem`

ForwardX11 `no`

PasswordAuthentication `no`

ホスト

上記の`~/.ssh/config`では、appleとawsのホストに対して特定の設定情報が記載されています。これらのどちらも一致しない場合は、一般的なパラメータが適用されます。この構成には、汎用パラメータはありません。コマンドの使用方法は次のとおりです：

`$ ssh apple`

または

\$ ssh aws

クラウドシステムでのSSH

- クラウドシステム上のSSHは、いくつかの自動化を適用した上で、同じように動作します。
- sshキーは、クラウドシステム作成時にデフォルトのユーザーで生成されます
- 公開鍵は、デフォルトユーザーの認可鍵ファイルにコピーされる
- パブリックとプライベートをローカルシステムにコピーするオプションがあらかじめ用意されている
- クラウドシステムのユーザーには、一般的にパスワードベースの認証はサポートされていません。

リモートシステムで公開鍵を置き忘れることはよくあることです。公開鍵の再作成は、**ssh-keygen** を使って行うことができます：

```
student@ubuntu:~/ssh$ cd ~/ssh$ ls -l
```

```
-rw----- 1 学生 学生 411 Jul 11 07:01 id_ed25519
-rw-r--r-- 1 学生 学生 96 Jul 11 07:01 id_ed25519.pub
```

公開鍵の名前を変更します：

```
student@ubuntu:~/ssh$ mv id_ed25519.pub id_ed25519.pub.lost student@ubuntu:~/ssh$ ls -l
```

```
-rw----- 1 学生 学生 411 Jul 11 07:01 id_ed25519
-rw-r--r-- 1 学生 学生 96 Jul 11 07:01 id_ed25519.pub.lost
```

```
student@ubuntu:~/ssh$ ssh-keygen -y -f ~/ssh/id_ed25519 > ~/ssh/id_ed25519.pub
student@ubuntu:~/ssh$ ls -l
```

```
-rw----- 1 学生 学生 411 Jul 11 07:01 id_ed25519
-rw-rw-r-- 1 学生 学生 96 Jul 11 07:04 id_ed25519.pub
-rw-r--r-- 1 学生 学生 96 Jul 11 07:01 id_ed25519.pub.lost
```

リモートグラフィカルログイン

- リモートマシンにフルグラフィカルデスクトップでログイン
- **VNC**（仮想ネットワークコンピューティング）を使うことが多い
- 一般的な実装は**tigervnc**
- というようにシンプルに：
 - リモートマシン（例：some_machine）にて：
`$ vncserver`
 - ローカルマシンにて：
`$ vncviewer -via server student@some_machine localhost:2`

このテストを行うには、まず**vnc**パッケージがインストールされていることを確認します：

\$ which vncserver vncviewer

```
/usr/bin/vncserver  
/usr/bin/vncviewer
```

これらのプログラムが見つからない場合は、次のような方法でインストールする必要があります：

\$ sudo [dnf|zypper|apt] tigervnc* をインストールする。

を使用して、正しいパッケージ管理システムコマンドを使用してください。(正確なパッケージ名はLinuxディストリビューションによって異なるため、正確なパッケージ名は記載していません。必要以上にインストールすることになるかもしれませんが、パッケージは大きくありません)。通常のユーザーとして、以下のコマンドでサーバーを起動します：

\$ vncserver

でテストすることができます。

\$ vncviewer ローカルホスト:2

(現在起動しているソフトやマシンの設定によって、2以外の数字(1, 3, 4, ...)で遊ぶ必要があるかもしれません)。リモートマシンから見る場合は、ちょっとだけ違います：

\$ vncviewer -via student@some_machine localhost:2

カラープロファイル」のために認証しなければならないというかなり奇妙なメッセージが表示され、パスワードが効かない場合は、次のようにサーバーマシンの**colord** daemonを終了させる必要があります。

\$ sudo systemctl stop colord

これはバグ（機能ではない）であり、不明確な理由により一部のディストリビューションと一部のシステムでのみ表示されます。

4.11 研究室



ビデオデモンストレーションのリソース

ユーザーアカウント使用デモ.mp4

練習問題4.1：ユーザーアカウントの操作

1. `etc/passwd`と`/etc/shadow`を調べ、各ファイルのフィールド、特に通常のユーザーアカウントのフィールドを比較します。何が同じで何が違うのか？
2. `useradd`を使用して`user1`アカウントを作成します。
3. `ssh`で`user1`としてログインします。でやればいいんです：
`$ ssh user1@localhost`
`user1`のパスワードが必要なため、失敗するはずです。
4. `user1`のパスワードを`user1pw`に設定し、`user1`で再ログインしてください。
5. `etc/passwd`に作成された新しいレコードを見てください、
`/etc/group`と`/etc/shadow`のファイルです。
6. `etc/default/useradd`ファイルを見て、現在のデフォルトが何に設定されているかを確認します。また、
`/etc/login.defs` ファイルも見てください。
7. Kornシェル（`ksh`）をデフォルトのシェルとして使用するユーザーアカウントを`user2`用に作成する。（`/bin/ksh`がない場合）をインストールするか、`/bin/csh`でCシェルを使用します）。パスワードを`user2pw`にする。
8. `etc/shadow`を見てください。`user1` アカウントの現在の有効期限は何ですか？
9. `chage`を使用して、`user1`のアカウントの有効期限を2013年12月1日に設定します。新しい有効期限を確認するために、`/etc/shadow` を見てください。
。
10. `usermod`を使用して、`user1`アカウントをロックします。
`etc/shadow`を見て、`user1`のパスワードについて何が変更されたかを確認する。この演習を完了するには、アカウントのパスワードを`userp1`にリセットしてください。

ソリューション4.1

1. `sudo grep student /etc/passwd /etc/shadow`

```
/etc/passwd:student:x:1000:100:LF Student:/home/student:/bin/bash
/etc/shadow:student:$6$jtoFVPICHhba$IGFFUU8ctrOG0istJ4/30DrNLI1FS66qnn0VbS6Mvm
luKI08SgbzT5.lcOHo5j/SOdCagZmF2RgzTvzLb11H0:16028:0:99999:7:::
```

(`student`の代わりに通常のユーザー名を使用することができます。) 一致するのはユーザー名フィールドだけです。

2. `$ sudo useradd user1`
3. `$ ssh user1@localhost`

`user1@localhost`のパスワードです：

のように、**sshd**サービスを最初に起動する必要があるかもしれないことに注意してください：


```
sudo service sshd restart
```

または

```
$ sudo systemctl restart sshd.service
```

4. `sudo passwd user1`

ユーザー `user1` のパスワードを変更します。新しいパスワードです :

5. `sudo grep user1 /etc/passwd /etc/shadow`

```
/etc/passwd:user1:x:1001:100::/home/user1:/bin/bash
/etc/shadow:user1:$6$OBE1mPMw$Clc7urbQ9ZSnyiniVOeJxKqLFu8fz4whfEexVem2
TFpucuWRN1CCHZ19XGhj4qVujsIRIS.P4aCXd/y1U4utv.:16372:0:99999:7:::
```

6. 例えば、RHELやopenSUSEのシステムで :

```
$ cat /etc/default/useradd
```

```
# useradd デフォルトファイル
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=。
SHELL=/bin/bash
SKEL=/etc/skel
CREATE_MAIL_SPOOL=yes
```

```
$ cat /etc/login.defs
```

```
----
```

2つ目のファイルは長いので再現しませんが、あなたのシステムで調べてみてください。

7. `$ sudo useradd -s /bin/ksh user2`

```
sudo passwd user2
```

ユーザー `user2` のパスワードを変更します。新しいパスワードです :

8. `sudo grep user1 /etc/shadow`

```
user1:$6$OBE1mPMw$Clc7urbQ9ZSnyiniVOeJxKqLFu8fz4whfEexVem2TFpucuWRN1CCHZ
19XGhj4qVujsIRIS.P4aCXd/y1U4utv.:16372:0:99999:7:::
```

賞味期限はないはずです。

9. `$ sudo chage -E 2013-12-1 user1`

```
sudo grep user1 /etc/shadow
```

```
user1:$6$OBE1mPMw$Clc7urbQ9ZSnyiniVOeJxKqLFu8fz4whfEexVem2TFpucuWRN1CCHZ
19XGhj4qVujsIRIS.P4aCXd/y1U4utv.:16372:0:99999:7::16040 :
```

10. `$ sudo usermod -L user1`

`sudo passwd user1`

第5章

グループ経営



5.1	グループ	102
5.2	所属団体	103
5.3	グループ経営	104
5.4	ユーザープライベートグループ	105
5.5	研究室	106

学習目標

このセッションが終わるころには、次のことができるようになっているはずです：

- Linuxユーザを1つ以上のグループに所属させることが有用である理由を説明する。
- グループメンバーシップの概念について説明する。
- **groupadd**、**groupdel**、**groupmod**、**usermod**を使用して、グループとそのメンバーを作成、削除、操作する。
- ユーザープライベートグループを説明する。

5.1 グループ

グループ

- 作業領域（ディレクトリ、ファイルなど）をユーザー間で共有できるようにする。
- グループメンバーへのアクセスは許可するが、全世界へのアクセスは許可しないファイルパーミッションの設定
- 特定の特定ユーザーが、他の方法では許可されないリソースにアクセスすることを許可する。

Linuxシステムでは、**グループ**と呼ばれるユーザーの集合体を形成し、そのメンバーは何らかの共通の目的を共有しています。その目的を達成するために、特定のファイルやディレクトリを共有し、いくつかの共通の特権を維持することで、システム上の他の人（時には総称して**ワールド**と呼ばれる）から分離されます。グループを利用することで、共同プロジェクトに大きな助けとなります。

ユーザーは1つまたは複数の**グループ**に属している

グループは**/etc/group**で定義され、**etc/passwd**がユーザーに対して持っているのと同じ役割をグループに対して持っています。ファイルの各行は次のようになっています：



/etc/group

```
....
グループ名 : パスワード : GID : user1,user2,...
....
```

- `groupname`はグループ名です。
- `password` は、パスワードのプレースホルダーです。グループパスワードは、**/etc/gshadow**が存在する場合のみ設定可能です。
- `GID`は、グループ識別子です。0から99の値は、システムグループ用です。100から**GID_MIN**（**/etc/login.defs**で定義されており、通常は**UID_MIN**と同じ）までの値は、特別なものとみなされます。**GID_MIN**を超える値は、**UPG**（User Private Groups）用です。
- `user1,user2,...` は、グループのメンバーであるユーザーのカンマ区切りのリストです。このグループがユーザーの主

5.2 所属団体

所属団体

Linuxユーザーです：

- 主要なグループが1つある
 - プライマリグループは`/etc/passwd`に記載されているグループです。
 - また、`/etc/group`にリストアップされます。
- 0～15個の二次グループに属する可能性がある
- のいずれかを実行することで、自分の所属するグループを特定することができます。
`$ groups [ユーザー1 ユーザー2 ...]`です。
`$ id -Gn [ユーザー1 ユーザー2 ...]`です。
- 引数なしで、どちらのコマンドも現在のユーザーについて報告します。

プライマリグループは、ユーザーがファイルやディレクトリを作成するときに使用されるGIDです。その他の二次的なグループに所属することで、ユーザーに追加の権限を与えることができます。

ユーザーは、`groups`コマンドまたは`id -Gn`コマンドを使用して、自分がどのグループのメンバーであるかを確認できます。デフォルトのグループは、ディストリビューションやインストールの仕様によって異なる場合があります。

CentOSの場合：

[学生@CentOS ~] グループ

スチューデント

Ubuntuの場合：

student@ubuntu ~\$ groups

student adm cdrom sudo dip plugdev lpadmin sambashare libvirt

5.3 グループ経営

グループ経営

グループアカウントで管理・維持することができる：

- **groupadd**：新しいグループを追加する
- **groupmod**：グループを修正し、新しいユーザーを追加する
- **groupdel**です：グループを削除する
- **usermod**です：ユーザーのグループメンバーシップを管理する

これらのグループ操作ユーティリティは、`/etc/group`と（存在する場合）`/etc/gshadow`を変更し、rootによってのみ実行される可能性があります。

例を挙げます：

```
$ sudo groupadd -r -g 215 staff
$ sudo groupmod -g 101 blah
sudo groupdel newgroup
$ sudo usermod -G 学生、グループ1、グループ2 学生
```

注意：**usermod -G**コマンドには十分注意してください；その後に表示されるグループリストは、変更点だけでなく、グループの完全なリストです。その後続くグループリストは、変更点だけでなく、グループの完全なリストです。抜け落ちた補足的なグループは消えてしまいます！破壊的でない使い方は、`-a`オプションを使用することで、新しいグループを追加する際に、既存のグループのメンバーシップを保持することができます。

5.4 ユーザープライベートグループ

ユーザープライベートグループ

- **Linux**はUPG (User Private Groups) を使用します。
- デフォルトでは、ユーザーのプライマリグループIDはユーザーIDと同じで、グループ名はユーザー名と同じです。
- UPGで作成されたすべてのユーザーには、**umask**が002に設定されます (**umask**については、後で詳しく説明します) 。
- ユーザーファイルはパーミッション664、ディレクトリは775で作成されます。

LinuxはUPG (User Private Groups) を採用しています。

UPGの考え方は、各ユーザーが自分だけのグループを持つというものです。ただし、UPGは非公開であることが**保証されているわけではなく**、**/etc/group**で誰かのプライベートグループにメンバーを追加することができます。

デフォルトでは、**useradd**でアカウントを作成したユーザーは：プライマリGID = UIDで、グループ名もユーザー名と同一です。

etc/profileで指定されているように、UPGで作成されたすべてのユーザーに対して**umask**が002に設定されます。このスキームでは、ユーザーファイルはパーミッション664 (rw-rw-r) で作成されます。)、ディレクトリは775(rwxrwxr-x)とします。(umaskについては後述します)。

5.5 研究室

練習5.1 : グループでの作業

1. 新しいユーザーアカウントを2つ作成し（下図のrockyとbullwinkle）、ホームディレクトリがあることを確認する。
2. friendsとbossesの2つの新しいグループを作成します（GIDは490です）。[etc/groupを見てください](#)。それぞれの新しいグループにどのようなGIDが与えられているかを確認する。
3. 両方の新しいグループにrockyを追加します。
bullwinkleさんをグループフレンドに追加します。
[etc/groupを見て](#)、どのように変化したかを確認します。
4. rockyでログインします。[somedirというディレクトリを作成し](#)、グループの所有者をbossesに設定する。（次回のセッションで説明するchgrpを使用する）
(rockyのホームディレクトリにallの実行権限を追加する必要があるかもしれません)。
5. bullwinkleでログインし、**touch**コマンドで/home/rocky/somedirにsomefileというファイルを作成してみてください。こんなことができるのでしょうか？いいえ、グループ所有権とディレクトリのchmod a+xがあるためです。
6. bullwinkleをbossesグループに追加して、もう一度試してみてください。新しいグループ・メンバーシップを有効にするために、ログアウトして再度ログインする必要があることに注意してください。次のことを行ってください：

ソリューション5.1

1. `sudo useradd -m rocky`
`$ sudo useradd -m bullwinkle`
`sudo passwd rocky`

新しいUNIXパスワードを入力します：
新しい UNIX パスワードを再入力します：
passwd: パスワードの更新に成功しました。

`sudo passwd bullwinkle`

新しいUNIXパスワードを入力します：
新しい UNIX パスワードを再入力します：
passwd: パスワードの更新に成功しました。

`$ ls -l /home`

```
計12
drwxr-xr-x 2 bullwinkle bullwinkle 4096 Oct 30 09:39 bullwinkle
drwxr-xr-x 2 ロッキー 岩石質 4096 10月30日 09:39 ロッキー
drwxr-xr-x 20 学生 スチューデント 4096 10月30日 09:18 学生
```

2. `sudo groupadd friends`
`$ sudo groupadd -g 490 bosses`
`$ grep -e friends -e bosses /etc/group`

friends:x:1003:bosses:x:4
90となります：

3. `$ sudo usermod -G friends,bosses rocky`
`$ sudo usermod -G friends bullwinkle`
`$ grep -e rocky -e bullwinkle /etc/group`

```

ロッキー : x:1001 : ブルウィ
ンクル : x:1002 : 友達 :
x:1003 : ロッキー、ブルウィ
ンクル ボス : x:490 : ロッキ
ー

```

ドルグループ ロッキー・ブルウィンクル

```

rocky : rocky friends bosses
bullwinkle : bullwinkle friends.

```

4. `$ ssh rocky@localhost`
`$ cd ~`
`$ mkdir somedir`
`$ chgrp bosses somedir`
`$ ls -l`

```

トータル16
-rw-r--r-- 1 rocky rocky 8980 Oct 4 2013 examples.desktop
drwxrwxr-x 2 rocky bosses 4096 Oct 30 09:53 somedir

```

`$ chmod a+x .`

5. `$ ssh bullwinkle@localhost`
`$ touch /home/rocky/somedir/somefile`

```

touch: /home/rocky/somedir/somefile に触れることができません : パーミッションが拒否されました

```

`$ 出口`

6. `$ sudo usermod -a -G bosses bullwinkle`
`$ ssh bullwinkle@localhost`
`$ touch /home/rocky/somedir/somefile`
`$ ls -al /home/rocky/somedir`

(ファイルの所有権に注意してください)

第6章

ファイルパーミッションとオーナーシップ



6.1	ファイルパーミッションとオーナーシップ	110
6.2	ファイルアクセス権	111
6.3	chmod、chown、chgrp	112
6.4	ウマスク	115
6.5	ファイルシステムACL	117
6.6	研究室	118

学習目標

このセッションが終わるころには、次のことができるようになっているはずです：

- オーナー、グループ、ワールドの概念を説明する。
- カテゴリーごとにファイルのアクセス権（読み取り、書き込み、実行）を設定する。
- ファイルへのアクセス要求を認証し、適切なパーミッションを尊重する。
- ファイルのパーミッションを変更するには**chmod**、ユーザーの所有権を変更するには**chown**、グループの所有権を変更するには**chgrp**を使用します。
- 新しく作成されたファイルに希望するパーミッションを設定する際の**umask**の役割を理解する。

109

6.1 ファイルパーミッションとオーナーシップ

オーナー、グループ、ワールド

- **owner**: ファイルを所有するユーザー（**user**とも呼ばれる）。
- **group**: アクセス権を持つユーザーのグループです。
- **other** : 世界の他の地域（**世界ともいう**）。

\$ ls -l

```
-rw-rw-r-- 1 学生 学生      1601 3月9日 15:04 mfile
-rw-r--r-- 1 student aproject 3280 4月15日  5:09 gfile
-rwxr-xr-x 1 student bproject 803280 Dec 9 11:42 doit
```

するとき :

\$ ls -l a_file

```
-rw-rw-r-- 1 coop aproject 1601 Mar 9 15:04 a_file
```

最初の文字（ファイルオブジェクトの種類を示す）の後に、潜在的なファイルユーザーに付与される**アクセス権**を示す9つの文字があります。これらは、**オーナー**、**グループ**、**その他**（**ワールドと同じ**）に対応する3つのグループに分かれて配置されている。

上記のリストでは、ユーザーはcoop、グループはaprojectです。

6.2 ファイルアクセス権

ファイルアクセス権

- とうように、ファイルの長いリストを行う場合：

```
$ ls -l /usr/bin/vi
```

```
-rwxr-xr-x.1 root root 910200 Jan 30 2014 /usr/bin/vi
```

3連符（文字2～10）のそれぞれは、以下の各セットを持つことができる：

- r: 読み取りアクセスが許可されています。
- w: 書き込みアクセスが許可されています。
- x: 実行アクセスが許可されている。

- の先攻後攻で認証が付与される：

- ファイルの所有権
- グループのメンバー
- ワールドパーミッション

許可されていない場合は、これらの文字の代わりに「-」が表示されます。

さらに、**setuid/setgid**パーミッションのように、各カテゴリーに特化したパーミッションも存在します。

これらの**ファイルアクセス許可**は、Linuxのセキュリティシステムの重要な部分である。ファイルへのアクセス要求には、要求するユーザーの資格情報と身元を、ファイルの所有者のものと比較する必要があります。

この認証は、次の順序で、これら3つの権限セットのいずれかに応じて付与されます：

1. 依頼者がファイル所有者の場合、ファイル所有者のパーミッションが使用されます。
2. それ以外の場合、要求者がファイルを所有するグループに属していれば、グループのパーミッションが調べられる；
3. それが成功しない場合は、ワールドパーミッションを調べます。

6.3 chmod、chown、chgrp

chmod

- ファイルのパーミッションの変更は、**chmod**で行います。
- 例えば、オーナーとワールドに実行権限を与え、グループに書き込み権限を再移動させるなどです：

```
$ ls -l a_file
```

```
-rw-rw-r-- 1 coop coop 1601 Mar 9 15:04 a_file
```

```
$ chmod uo+x,g-w a_file
```

```
$ ls -l a_file
```

```
-rwxr--r-x 1 coop coop 1601 Mar 9 15:04 a_file
```

ここで、uはユーザー（所有者）、oはその他（世界）、gはグループを表します。

- でない限り、自分が所有するファイルのパーミッションを変更することしかできません。

許可-cPanebremreispsreisoenntsedaeritheor alseanbgnvape,nusiunallny uwmritteenriicn 2bmr n,
asusycmhboalicsfo6rm44Octal ビットマップは通常0755に見える一方、記号表現は u+rwx,g+rwx,o+rx と見える。

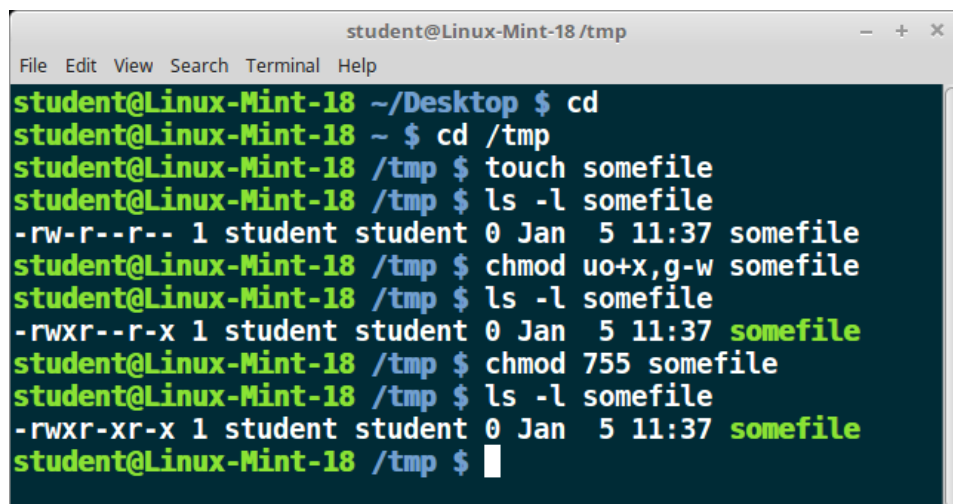
オクタル・デジット

- 8進数表現は、各桁の和を表す：
 - 読み出し許可が必要な場合は、4。
 - 書き込み許可が必要な場合は、2。
 - 実行許可を希望する場合は、1。

記号構文は入力や記憶が難しいので、8進数の省略記法を使うことが多く、1ステップですべてのパーミッションを設定することができます。これは簡単なアルゴリズムで行われ、各エンティティの3つのパーミッションビットをすべて指定するには、1桁の数字で十分です。

したがって、7はリード/ライト/エグゼキュート、6はリード/ライト、5はリード/エグゼキュートを意味する。

これを**chmod**で適用する場合、以下の例のように3桁の数字にそれぞれ値を与える必要があります：

A terminal window titled 'student@Linux-Mint-18 /tmp' showing a series of commands and their outputs. The commands are: 'cd', 'cd /tmp', 'touch somefile', 'ls -l somefile' (showing permissions -rw-r--r--), 'chmod uo+x,g-w somefile', 'ls -l somefile' (showing permissions -rwxr--r-x), 'chmod 755 somefile', and 'ls -l somefile' (showing permissions -rwxr-xr-x).

```
student@Linux-Mint-18 /tmp
File Edit View Search Terminal Help
student@Linux-Mint-18 ~/Desktop $ cd
student@Linux-Mint-18 ~ $ cd /tmp
student@Linux-Mint-18 /tmp $ touch somefile
student@Linux-Mint-18 /tmp $ ls -l somefile
-rw-r--r-- 1 student student 0 Jan  5 11:37 somefile
student@Linux-Mint-18 /tmp $ chmod uo+x,g-w somefile
student@Linux-Mint-18 /tmp $ ls -l somefile
-rwxr--r-x 1 student student 0 Jan  5 11:37 somefile
student@Linux-Mint-18 /tmp $ chmod 755 somefile
student@Linux-Mint-18 /tmp $ ls -l somefile
-rwxr-xr-x 1 student student 0 Jan  5 11:37 somefile
student@Linux-Mint-18 /tmp $
```

図6.1 : chmodの使用

chownとchgrp

- **chown**でファイルの所有者を変更する（superuserのみ可能）
`$ sudo chown wally somefile`
- **chgrp**でグループの所有権を変更する（自分が所属しているグループのみ）
`$ sudo chgrp cleavers サムファイル`
 - 両方を同時に変更する：
`$ sudo chown wally:cleavers somefile` です。
ここで、オーナーとグループをコロン（またはピリオド）で区切ります。
- **Rオプション**で再帰的に使用することも可能です：
`$ sudo chown -R wally:cleavers ./`
`sudo chown -R wally:wally subdir`

ファイルの所有者の変更は**chown**で、グループの変更は**chgrp**で行います。

ファイルの所有権を変更できるのは、スーパーユーザーだけです。同様に、グループの所有権を変更できるのは、自分がメンバーであるグループのみです。

Rオプションを使用すると、引数の下にあるすべてのファイルとディレクトリの所有者が変更されるようになります。

6.4 ウマスク

ウマスク

- ファイルやディレクトリのデフォルトパーミッションを設定するために使用します。
- **umask**: umaskを設定するプログラム
 - 保留する権限を列挙する
- **etc/profile**で設定されたデフォルトのumask。
 - rootのumaskは022
- 任意のユーザーで設定可能

ファイルを作成するときに与えられるデフォルトのパーミッションは、オーナー、グループ、ワールドに対して読み取り/書き込み (0666)、ディレクトリに対しては、全員に対して読み取り/書き込み/実行 (0777) です。しかし、次のようにすると

タッチ エイファイル

```
$ mkdir adir
```

```
ls -l | grep -e afile -e adir
```

```
drwxrwxr-x 2 coop coop 4096 Sep 16 11:18 adir
-rw-rw-r-- 1 coop coop    0 9月16日 11:17 afile
```

実際のパーミッションは、ファイルが664、ディレクトリが775に変更されていることに気づくでしょう。これは、現在の **umask** によって変更されたもので、その目的は、どのパーミッションが拒否されるべきかを示すことにあります。現在の値は、次のようにして表示することができます：

```
$ umask
```

```
0002
```

これは、システム管理者がユーザーに対して設定する最も一般的な値である。この値をファイル作成権限と組み合わせて、実際の結果を得ることができます、

0666 & ~002 = 0664; すなわち、rw-rw-r--。

のように、**umask** コマンドでいつでも **umask** を変更することができます。

```
$ umask 0022
```


6.5 ファイルシステムACL

ファイルシステムACL

- **POSIX ACL (Access Control Lists)** は、よりシンプルなユーザー、グループ、ワールドシステムを拡張します。
- 777パーミッションを使用せずにファイル/ディレクトリを共有するために使用します。
- マウントオプション(acl)として実装
- インストール時に作成されるファイルシステムで設定されるデフォルト
- ACLの取得/設定にはgetfacl/setfaclを使用します。
- 例を挙げます :
 - \$ getfacl /home/stephane/file1
 - \$ setfacl -m u:isabelle:rx /home/stephane/file1
 - \$ setfacl -x u:isabelle /home/stephane/file

特定のオブジェクトやオブジェクトのクラスにアクセスする際に、特定のユーザーやユーザーグループに特定の権限を付与することができます。

Linuxカーネルは**ACL**の使用を可能にするが、それでも特定のファイルシステムにも同様に実装する必要がある。最近のLinuxディストリビューションで使用されている主要なファイルシステムはすべて**ACL**拡張を組み込んでおり、マウント時にオプション-aclを使用することができます。**ACL**のデフォルトセットは、システムのインストール時に作成されます。

新しいファイルは、そのファイルが存在するディレクトリのデフォルト**ACL** (設定されている場合) を継承することに注意してください。また、mvとcp -pは、**ACL**を保持することに注意してください。

ACLのことです。

ACLを削除するには、次のようにします :

```
$ setfacl -x u:isabelle /home/stephane/file1
```

ディレクトリにデフォルトを設定する場合 :

```
$ setfacl -m d:u:isabelle:rx somedir
```

6.6 研究室



ビデオデモンストレーションのリソース

ウマスクデモ.mp4

練習問題6.1 : chmodの使い方

chmodを使用する際、パーミッションの指定方法として、8進数方式と記号方式のどちらかを使用することができます。ここでは、シンボリック方式についてもう少し詳しく説明します。

パーミッションを直接与えることも、パーミッションを追加・削除すること

も可能です。次の例を試してみてください :

```
$ chmod u=r,g=w,o=x afile
$ chmod u+w,g-w,o+rw afile
chmod ug=rwx,o-rw アファイル
```

各ステップの後に行う :

```
$ ls -l afile
```

をクリックして、パーミッションがどのようにかかったかを確認し、いくつかのバリエーションを試してみてください。

練習問題6.2: umask

で空のファイルを作成します :

```
$ touch afile
$ ls -l afile
```

```
-rw-rw-r-- 1 coop coop 0 Jul 26 12:43 afile
```

を見ると、デフォルトで作成され、オーナーとグループには読み取りと書き込みの両方の権限が与えられているが、ワールドには読み取り権限しか与えられていないことがわかる。

実際、オペレーティングシステムレベルでは、ファイルやディレクトリを作成するときに与えられるデフォルトのパーミッションは、実際にはオーナー、グループ、ワールド (0666) に対する読み取り/書き込みであり、デフォルト値は実際には現在の`umask`によって変更されています。

`umask`と入力するだけで、現在の値が表示されます :

```
$ umask
```

```
0002
```

これは、システム管理者がユーザーに対して設定する最も一般的な値である。この値をファイル作成権限と組み合わせて、実際の結果を得ることができます、

`0666 & ~002 = 0664`; すなわち、`rw-rw-r--`。

のように、`umask`を変更して新しいファイルを作成し、その結果のパーミッションを確認してみてください :

```
$ umask 0022  
$ touch afile2  
$ umask 0666  
$ touch afile3  
$ ls -l afile*
```

演習6.3: アクセスコントロールリストの活用

1. いつものユーザー名でファイルを作成し、そのファイルに対して**getfacl**を実行し、プロパティを確認します。
2. デフォルトのプロパティで新しいユーザーアカウントを作成します（または、以前の演習で使用したものを再利用します）。
3. そのユーザーでログインし、最初のステップで作成したファイルに行を追加しようとします。これは失敗するはずです。
4. **setfacl**を使用して、新しいユーザーがファイルを書き込めるようにしてから、もう一度試してみてください。
5. **setfacl**で新しいユーザーには読めないファイルにして、もう一度試してみてください。
6. 必要に応じて清掃を行う。

解決策 6.3

ターミナルウィンドウを2つ開き、1つは通常のユーザーアカウントとして、もう1つはセカンダリーアカウントとして作業するのが最も簡単でしょう。

1. ウィンドウ1では

```
$ echo これはファイルです > /tmp/afile
$ getfacl /tmp/afile
```

```
getfacl: 絶対パス名から先頭の '/' を除去する # file:
tmp/afile
# owner: coop
# group: coop
user::rw-
group::rw-
other::r-- です。
```

2. ウィンドウ1では

```
$ sudo useradd fool
sudo passwd fool
...
```

3. ウィンドウ2にて :

```
$ sudo su - fool
$ echo another line > /tmp/afile
```

```
-bashです : /tmp/afile : パーミッションが拒否されました
```

4. ウィンドウ1では

```
$ setfacl -m u:fool:rw /tmp/afile
$ getfacl /tmp/afile
```

```
getfacl: 絶対パス名から先頭の '/' を除去する # file:
tmp/afile
# owner: coop
# group: coop
user::rw-
user:fool:rw-
group::rw-
mask::rwx
other::r--.
```

ウィンドウ2にて :

```
$ echo another line > /tmp/afile
```

5. ウィンドウ1では

```
$ setfacl -m u:fool:w /tmp/afile
```

ウィンドウ2にて :

```
$ echo another line > /tmp/afile
```

```
-bashです : /tmp/afile : パーミッションが拒否されました
```

6. 掃除することです :

```
$ rm /tmp/afile
```

```
$ sudo userdel -r fool
```

第7章

パッケージマネジメントシステム



7.1	なぜパッケージを使うのか？	122
7.2	ソフトウェアパッケージの概念	123
7.3	パッケージの種類	124
7.4	利用可能なパッケージマネジメントシステム	125
7.5	パッケージングツールのレベル・バリエーション	126
7.6	パッケージのソース	127
7.7	ソフトウェアパッケージの作成	128
7.8	リビジョンコントロールシステム	129
7.9	利用可能なソースコントロールシステム	130
7.10	Linuxカーネルとgit	131
7.11	研究室	133

学習目標

このセッションが終わるころには、次のことができるようになっているはずです：

- ソフトウェアパッケージ管理システムが必要な理由を説明する。
- バイナリパッケージとソースパッケージの両方の機能を理解する。
- 利用可能な主なパッケージマネジメントシステムを列挙する。
- 裸のパッケージだけを扱うユーティリティと、パッケージ間の依存関係を扱うユーティリティの2つのレベルが必要な理由を理解する。
- 独自のパッケージを作成することで、ソフトウェアの中身やインストール方法を正確にコントロールできるようになることを説明します。
- ソース管理システムの役割を理解し、特にgitを理解する。

7.1 なぜパッケージを使うのか？

なぜパッケージを使うのか？

- 自動化を実現：手動でのインストールやアップグレードが不要
- スケーラビリティがある：1つのシステムにパッケージをインストールすることも、10,000のシステムにパッケージをインストールすることも可能です。
- 再現性・予測可能性
- セキュリティと監査

ソフトウェアパッケージ管理システムは、Linux が企業の IT 環境にもたらした最大の進歩の 1 つとして広く認識されています。ファイルやメタデータを自動化された予測可能で信頼できる方法で追跡することにより、システム管理者はパッケージ管理システムを使用して、個々のシステムで手作業を必要とせずに、インストールプロセスを何千ものシステムに拡張することができます。

7.2 ソフトウェアパッケージの概念

ソフトウェアパッケージの概念

- 関連するソフトウェアファイルを1つのパッケージ（アーカイブ）に集め、圧縮する（他のパッケージを先にインストールする必要がある場合もある）
- ソフトウェアのインストールや取り外しが簡単にできるようにする
- 内部データベースによるファイルの完全性検証も可能
- パッケージの原産地認証が可能
- アップグレードを促進する
- 論理的な特徴でパッケージをグループ化する
- パッケージ間の依存関係を管理する

パッケージ管理システムは、システム管理者がソフトウェアパッケージのインストール、アップグレード、設定、削除を、既知の、予測可能な、一貫した方法で自動化できるようにするツールを提供します。

パッケージには、実行ファイル、データファイル、ドキュメント、インストールスクリプト、コンフィギュレーションファイルが含まれます。また、バージョン番号、チェックサム、ベンダー情報、依存関係、説明などの**メタデータ**属性も含まれます。

インストールすると、その情報はすべて内部のデータベースに保存され、バージョンやアップデート情報の照会が可能になります。

7.3 パッケージの種類

パッケージの種類

- バイナリパッケージには、実行可能なファイルやライブラリなど、展開に必要なファイルが含まれています。これらは、アーキテクチャに依存します。
- ソースパッケージはバイナリパッケージを生成するために使用されます。ソースパッケージからバイナリパッケージを常に再構築できる必要があります。1つのソースパッケージを複数のアーキテクチャに使用することができます。
- アーキテクチャに依存しないパッケージには、スクリプトインタプリタ下で動作するファイルやスクリプト、ドキュメントや設定ファイルなどが含まれています。
- メタパッケージとは、デスクトップ環境やオフィススイートなど、比較的大きなサブシステムをインストールするために必要なあらゆるものを集めた関連パッケージのグループです。

バイナリパッケージは、システム管理者が最も頻繁に扱うものです。32ビットプログラムを実行できる64ビットシステムでは、あるプログラムに対して、**x86_64**または**amd64**という名前のバイナリパッケージと、**i386**または**i686**という名前のバイナリパッケージの2つがインストールされていることがあります。

ソースパッケージは、バイナリパッケージを作成するために使用された変更とソースコードを追跡するのに役立ちます。通常、ソースパッケージはデフォルトではシステムにインストールされませんが、常にベンダーから取得することができます。

例えば、RPMベースのシステムでは、**p7zip**のバイナリパッケージをこうしてリビルドすることができるのです：

```
# rpmbuild --rebuild -rb p7zip-16.02-16.el8.src.rpm
```

で、結果は **/root/rpmbuild** に置かれます：

```
# root/rpmbuild>find .-名前 "**rpm"
```

```
./RPMS/x86_64/p7zip-plugins-16.02-16.el8.x86_64.rpm
./RPMS/x86_64/p7zip-debugsource-16.02-16.el8.x86_64.rpm
./RPMS/x86_64/p7zip-plugins-debuginfo-16.02-16.el8.x86_64.rpm
./RPMS/x86_64/p7zip-16.02-16.el8.x86_64.rpm
./RPMS/noarch/p7zip-doc-16.02-16.el8.noarch.rpm
```

となっており、正確な場所はLinuxのディストリビューションとバージョンに依存します。

7.4 利用可能なパッケージマネジメントシステム

利用可能なパッケージマネジメントシステム



図7.1 : RPMとAPT

非常に一般的な2つのパッケージ管理システム：

- **RPM (レッドハットパッケージマネージャー)**

Red Hat Enterprise Linux、Fedora、CentOSなど、すべての**Red Hat**派生ディストリビューション、および**SUSE**とその関連コミュニティ**openSUSE**ディストリビューションで使用されています。

- **dpkg (Debianパッケージ)**

Debian、Ubuntuを含むすべての**Debian**派生ディストリビューションで使用されています。

Linux Mintです。

パッケージ管理システムは他にもあり、**Gentoo**で使われている**portage/emerge**、**Arch**で使われている**pacman**、**Embedded Linux**システムや**Android**で使われている特殊なものなどがあります。

また、古くは、パッケージの管理や削除を一切行わず、ただ**tarball**として提供するという方法もありました。この方法は、最も古い**Linux**ディストリビューションの1つである**Slackware**に残っています。

しかし、ほとんどの場合、**RPM**か**dpkg**のどちらかであり、このコースではこれらだけを検討します。

7.5 パッケージングツールのレベル・バリエーション

パッケージングツールのレベル・バリエー

- ローレベルユーティリティ :

単一のパッケージ、または名前を付けたパッケージのリストをインストールまたは削除します。

- 依存関係が完全に処理されていない、警告されるだけ、またはエラーが発生する。
- 他のパッケージのインストールが必要な場合、最初のインストールは失敗します。
- パッケージが他のパッケージによって必要とされている場合、削除は失敗します

- 例 : rpm、dpkg

- ハイレベルなユーティリティ

依存関係の問題を解決する :

その他必要なパッケージをインストールします

パッケージングシステムには2つのレベルがあります :

1. **低レベルのユーティリティです :** このユーティリティは、単一のパッケージ、または各パッケージが個別に指定されたパッケージのリストをインストールしたり削除したりするだけです。依存関係は完全に処理されず、警告が表示されるか、エラーが発生するのみです。

- 他のパッケージのインストールが必要な場合、最初のインストールは失敗します。
- パッケージが他のパッケージによって必要とされている場合、削除は失敗します。

rpmとdpkgユーティリティは、それらを使用するパッケージングシステムでこの役割を果たします。

2. **ハイレベルなユーティリティです :** 依存関係の問題を解決するものです :

- ソフトウェアをインストールする前に、別のパッケージやパッケージ群をインストールする必要がある場合、そのようなニーズを満たすことができます。
- パッケージの削除がインストールされている他のパッケージと干渉する場合、管理者は、中止するか、影響を受けるすべてのソフトウェアを削除するかのいずれかを選択することができます。

dnfとzypperユーティリティ (と古いyum) はrpmシステムの依存関係の解決を行い、aptとdpkg系ではapt-cacheなどのユーティリティが対応しています。

このコースでは、パッケージング管理システムのコマンドラインインタフェースについてのみ説明します。各Linuxディストリビューションで使用されているグラフィカルなフロントエンドも便利ですが、特定のインタフェースに縛られず、より柔軟性を持たせたいと考えています。

7.6 パッケージのソース

パッケージのソース

- ディストリビューションに複数のリポジトリがある
- すべてのパッケージがうまく連動すること
- 外部リポジトリも存在する
 - 一部のインターフェイスをきれいに
 - その他、コンフリクトを起こす

どのディストリビューションにも1つ以上のパッケージリポジトリがあり、システムユーティリティがソフトウェアを入手したり、新しいバージョンに更新したりするために利用します。ディストリビューションは、リポジトリにあるすべてのパッケージが互いにうまく機能するようにするのが仕事です。

ディストリビューションがサポートする標準的なリストに追加できる外部リポジトリは常に存在します。例えば、**EPEL** (Extra Packages for Enterprise Linux)と呼ばれるバージョン依存のリポジトリは、ソースがFedoraでメンテナがRed Hatに近いため、**RHEL**によく適合しています。

しかし、外部リポジトリの中には、あまりうまく構築・維持されていないものもあります。例えば、メインリポジトリでパッケージが更新されても、依存するパッケージが外部リポジトリで更新されないことがあり、これが**依存性地獄**の一形態となることがある。

7.7 ソフトウェアパッケージの作成

ソフトウェアパッケージの作成

- カスタムソフトのインストールが簡単なパッケージの作成
- パッケージは、インストール時に追加のタスクを実行することができます。
 - 必要なシンボリックリンクを作成する
 - 必要に応じてディレクトリを作成する
 - パーミッションの設定
 - スクリプト化できるもの
- 多くのフォーマットを用意しています
 - **Red Hat**および**SUSE**ベースのシステム向け**RPM**
 - **Debian**用**DEB** **Debian**ベースのシステム用
 - **Slackware**用**TGZ**
 - **Android**用**APK**

独自のカスタムソフトウェアパッケージを構築することで、独自のソフトウェアを簡単に配布・インストールすることができます。**Linux**のほぼすべてのバージョンに、これを行うための何らかのメカニズムがあります。

独自のソフトウェアを構築することで、ソフトウェアの中身やインストール方法を正確にコントロールすることができます。パッケージをインストールすると、新しいソフトウェアのインストールや古いソフトウェアの削除に必要なすべてのタスクを実行するスクリプトが実行されるように作成することができます。

7.8 リビジョンコントロールシステム

なぜリビジョンコントロールシステムを使

- プロジェクトの複雑さは、貢献者、ユーザー、テスターの規模や数によって大きくなります。
- リビジョン管理システム（バージョン管理システム）：
 - 変更の履歴（ログ）を正確に残す
 - 以前のリリースへのバックアップが可能
 - 相反する貢献を解決できる
- 多くの方式が採用されている

ソフトウェアプロジェクトは、プロジェクトの規模が大きくなればなるほど、あるいは開発者の数が増えれば増えるほど、その管理は複雑になっていきます。

アップデートを整理し、協力を促進するために、ソース管理のためのさまざまなスキームが利用可能です。このようなプログラムの標準的な機能には、変更の正確な履歴（ログ）を保持する機能、以前のリリースにバックアップできる機能、複数の開発者による競合しそうなアップデートを調整する機能などがあります。

ソース管理システム（またはリビジョン管理システムとも呼ばれる）は、共同開発を調整する役割を担っています。

7.9 利用可能なソースコントロールシステム

利用可能なソースコントロールシス

表7.1 : 利用可能なソースコントロールシステ

製品	URL
RCS	https://www.gnu.org/software/rcs/
シーバス	https://www.nongnu.org/cvs
サブバージョン	https://subversion.apache.org
ギット	https://www.kernel.org/pub/software/scm/git/
GNU Arch	https://www.gnu.org/software/gnu-arch
モノトーン	https://www.monotone.ca
メルカリ	https://www.mercurial-scm.org

プロプライエタリ、オープンに関わらず、利用可能な製品に事欠きません。

のライセンスが必要です。

ここでは、Linuxカーネル開発コミュニティから生まれた、広く使われているgitにのみ焦点を当てます。gitは、オープンソースプロジェクトにおいて、驚くほど短期間で圧倒的な地位を確立し、クローズドソース環境でもよく使われるようになりました。

7.10 Linuxカーネルとgit

Linuxカーネルとgit

- **Linux**カーネル開発システムには、特別なニーズがあります：
 - 世界中に広く普及している
 - 数千人のデベロッパーが
 - **GPL**ライセンスのもと、非常に公開されています。
- **git**はそのようなニーズを満たすために生まれ、**Linux**の効率的な成長を支えられました。



gitとその使用方法については、後の章で詳しく説明します。ここでは、git がどのように動作し、どのように使用するかを、付属のラボ演習で感じてみましょう。

Linuxカーネル開発システムは、世界中に広く分布しており、文字通り何千人もの開発者が関わっているという点で特別なニーズがあります。さらに、**GPL**ライセンスのもと、非常に公に行われています。

長い間、本当のソースリビジョンコントロールシステムはありませんでした。その後、主要なカーネル開発者は、**Linux**カーネル開発のための制限付き使用ライセンスを付与する商業プロジェクトである**BitKeeper** (<https://www.bitkeeper.com>) を使用するようになりました。

しかし、2005年春にライセンス制限をめぐる公然の紛争が発生し、**BitKeeper**の無償利用は**Linux**カーネル開発では不可能となった。

その結果、Linus Torvalds が原作者である **git** が開発されました。**git** のソースコードは <https://www.kernel.org/pub/software/scm/git/>、完全なドキュメントは <https://www.kernel.org/pub/software/scm/git/docs/> で見ることができる。

gitのしくみ

- 従来のバージョン管理システムではない
- ファイルを基本単位とした動作はしない
- オブジェクトデータベースには
 - ブロブ
 - 樹木
 - コミット
- **Directory Cache**は、ディレクトリツリーの状態を取得します。
- グラフィカルなインターフェイスを持つ
- **github**などのサイトには数百万もの**git**リポジトリがある

技術的には、**git** は通常の意味でのソース管理システムではなく、扱う基本単位もファイルではありません。**オブジェクトデータベース**と**ディレクトリキャッシュ**という2つの重要なデータ構造を持っています。

オブジェクトデータベースには、3種類のオブジェクトが含まれています：

- **ブロブ**：ファイルの内容を含むバイナリデータの塊。
- **ツリー (Trees)**：ファイル名と属性を含むblobの集合で、ディレクトリ構造を示す。
- **コミットです**：ツリースナップショットを記述したチェンジセット。

ディレクトリキャッシュは、ディレクトリツリーの状態を把握します。

コントロールシステムをファイル単位のシステムから解放することで、多くのファイルを含むチェンジセットをより適切に処理できるようになります。

gitは常に急速な発展を遂げており、そのためのグラフィカル・インターフェースもまた、急速な発展を遂げています。例えば、<https://www.kernel.org/git/>。ソースツリーだけでなく、特定の変更点を簡単にブラウズすることができます。

<https://www.github.com>などのサイトでは、公開・非公開を問わず、文字通り何百万もの **git** リポジトリをホストしています。**git**を有益に使うための記事、書籍、オンラインチュートリアルなど、簡単に見つけることができるものがたくさんあります。

7.11 研究室



ビデオデモンストレーションのリソース

パッケージ管理デモ.mp4



練習問題7.1 : gitによるバージョン管理



gitの詳細については、後のセクションで説明します。ここでは、gitの使い方の基本的な感覚をつかむだけにしておきます

gitがインストールされていることを確認する

あなたのシステムにはすでにgitがインストールされているかもしれません。gitがすでにインストールされているかどうかは、gitを実行すると表示されるはずです。もしそうでなければ、ソースを入手してコンパイルしてインストールすることもできますが、通常はコンパイル済みの適切なバイナリパッケージをインストールする方が簡単です。正確なパッケージ名は異なりますが、お使いのディストリビューションに応じて、以下のいずれかが動作するはずです :

```
sudo apt install git*      # Debian /Ubuntu
sudo zypper install        git*# openSUSE
$ sudo dnf install         git*# Fedora / RHEL 8 / CentOS 8
$ sudo yum install        git*# RHEL 7 / CentOS 7
```

gitの仕組みと使いやすさを実感してみましょう。とりあえず、自分たちのローカルプロジェクトを作るだけにしておきます。

1. まず、作業ディレクトリを作成し、それを扱うためにgitを初期化します :

```
mkdir git-test
$ cd git-test
$ git init
```

2. プロジェクトを初期化すると、すべてのバージョン管理情報を含む.gitディレクトリが作成されますが、プロジェクトに含まれる主要なディレクトリはそのまま残ります。このディレクトリの初期コンテンツは次のようになります :

```
$ ls -l .git
```

```
計40
drwxrwxr-x 2 coop coop 4096 Dec 30 13:59 branches/
-rw-rw-r-- 1 coop coop   92 12月30日 13:59 コンフィグ
-rw-rw-r-- 1 coop coop   58 12月30日 13:59 説明
-rw-rw-r-- 1 coop coop   23 12月30日 13:59 HEAD
drwxrwxr-x 2 coop coop 4096 Dec 30 13:59 hooks/
drwxrwxr-x 2 coop coop 4096 Dec 30 13:59 info/
drwxr-xr-x 4 coop coop 4096 Dec 30 13:59 objects/
drwxrwxr-x 4 coop coop 4096 Dec 30 13:59 refs/
```

このディレクトリとそのサブディレクトリの内容については後ほど説明しますが、ほとんどの場合、最初は空っぽです。

3. 次にファイルを作成し、プロジェクトに追加します :

```
$ echo some junk > somejunkfile
git add somejunkfile
```

4. で現在のステータスを確認することができます :

\$ git status

```

ブランチ上のマ
スター 初期コミ
ット
コミットする変更 :
(ステージを解除するには "git rm --cached <file>..." を使用します)

新しいファイル : somejunkfile

```

ファイルがステージングされているが、まだコミットされていないことを伝えていることに注意してください。

5. このリポジトリの責任者をgitに教えてあげましょう :

```

$ git config user.name "もう一人の天才"
$ git config user.email "b_genius@linux.com"

```

これは、グローバル設定ファイルで事前に定義していない限り、新しいプロジェクトごとに行う必要があります。

6. では、ファイルを修正し、差分の履歴を見てみましょう :

```

$ echo 別の行 >> somejunkfile
$ git diff

```

```

diff --git a/somejunkfile b/somejunkfile
index 9638122...6023331 100644
--- a/somejunkfile
+++ b/somejunkfile
@@ -1 +1,2 @@
 屑物
+αのライン

```

7. 変更を実際にリポジトリにコミットするために、私たちはこうします :

```

$ git commit -m "私の初期コミット"

```

```

初期コミット作成 eafad66: 私の初期コミットを作成しました。
 1 ファイル変更、1 挿入(+), 0 削除(-) 作成モード
100644 somejunkfile

```

mオプションでコミットに付随する識別メッセージを指定しない場合、エディタに飛び込んで内容を入力することになります。これを実行しないと、コミットは拒否されます。選択されるエディタは、EDITOR環境変数で設定されたものです。これはGIT_EDITORを設定することで優先させることができます。

8. で履歴を見ることができます :

```

git log

```

```

commit eafad66304ebbcd6acfe69843d246de3d8f6b9cc
Author : ある天才 <a_genius@linux.com>
日付:   Wed Dec 30 11:07:19 2009 -0600

```

私の最初のコミット

で、そこに入っている情報を見ることができます。長い16進数の文字列がコミット番号であることに気づきでしょうか。これは160ビット、40桁のユニークな識別子です。

9. これで、すでに終了しているファイルを修正したり、git addで新しいファイルを追加したりすることが自由にできるようになりました。しかし、それらはあなたが別の git commit を行うまでステージされます。
10. 今のは、それほど悪くはなかった。しかし、私たちはまだ表面しか見ていません。

第8章

デップグ



8.1	DPKG (Debianパッケージ)	138
8.2	パッケージのファイル名とソース	139
8.3	DPKGクエリ	140
8.4	インストール/アップグレード/アンインストール	141
8.5	研究室	142

学習目標

このセッションが終わるころには、次のことができるようになっているはずです：

- Debian Packaging System (DPKG)の方法と使用方法について説明する。
- バイナリーファイルとソース.debファイルの両方で使用される命名規則について説明する。
- ソースパッケージがどのようなものか知っている。
- パッケージに対するクエリおよびベリファイ操作を使用します。
- Debianパッケージのインストール、アップグレード、アンインストールを行う。

8.1 DPKG (Debianパッケージ)

DPKGエッセンシャルズ

- **Ubuntu**を含むすべての**Debian**ベースの**Linux**ディストリビューションで使用されています。
Linux Mint
- **rpm**のように、**dpkg**ユーティリティはパッケージを回収したり、デペンデンシー障害を改善したりはしません。
- パッケージファイルの接尾辞は「.deb」です
- **DPKG**データベースは/var/lib/dpkgに存在します。

DPKG (Debian Package) は、**Debian Linux** およびそこから派生した他のディストリビューションで、ソフトウェアパッケージのインストール、削除、管理に使用されるパッケージシステムです。**RPM**のように、日常的に使用するパッケージを直接取得するのではなく、ローカルにインストールしたり削除したりするように設計されています。

パッケージファイルの接尾辞は.debで、**DPKG**データベースは/var/lib/dpkgディレクトリに常駐しています。

rpm と同様、**dpkg** プログラムは宇宙を部分的にしか見ていません。**dpkg** はシステムにインストールされているものと、コマンドラインで与えられたものだけを知っていますが、システム上の他のディレクトリやインターネット上にある他の利用可能なパッケージのことは何も知りません。そのため、依存関係が満たされていない場合や、インストールされている他のパッケージが必要とするパッケージを削除しようとした場合にも失敗することになります。

8.2 パッケージのファイル名とソース

パッケージのファイル名とソース

A

バイナリパッケージの標準的な命名形式：

```
<名前>_<バージョン>-<リビジョン_番号>_<アーキテクチャ>.deb
```

のように：

```
logrotate_3.14.0-4_amd64.deb
logrotate_3.14.0-4ubuntu3_amd64.deb
```

- ・ ソースパッケージは、少なくとも3つのファイルから構成されています：
 - アップストリームのtarball（末尾が.tar.gz）です。
 - 説明ファイル（末尾に.dscがつくもの）
 - 上流ソースへのパッチと、そのパッケージのために作成された追加ファイルを含む、2つ目のtarball（.diff.gz または .debian.tar.gz で終了します）です。

Debianのパッケージファイル名は、特定の情報を表すフィールドをベースにしています。

歴史的な理由から、64ビットx86プラットフォームはx86 64ではなくamd64と呼ばれ、Ubuntuなどのディストリビューターはパッケージ名にその名前をなんとか挿入しています。

Ubuntuシステムでは、ソースパッケージをダウンロードして、どのようなファイルがダウンロードされ、どのようなファイルが作成されたかを見ることができます：

```
student@ubuntu: /tmp
student@ubuntu:/tmp$ apt-get source logrotate
Reading package lists... Done
NOTICE: 'logrotate' packaging is maintained in the 'Svn' version control system at:
http://svn.fedorahosted.org/svn/logrotate/
Need to get 84.1 kB of source archives.
Get:1 http://archive.ubuntu.com/ubuntu zesty/main logrotate 3.8.7-2ubuntu3 (dsc) [1,924 B]
Get:2 http://archive.ubuntu.com/ubuntu zesty/main logrotate 3.8.7-2ubuntu3 (tar) [58.9 kB]
Get:3 http://archive.ubuntu.com/ubuntu zesty/main logrotate 3.8.7-2ubuntu3 (diff) [23.3 kB]
Fetched 84.1 kB in 0s (129 kB/s)
dpkg-source: info: extracting logrotate in logrotate-3.8.7
dpkg-source: info: unpacking logrotate_3.8.7.orig.tar.gz
....
student@ubuntu:/tmp$ du -shc logrotate*
1.2M  logrotate-3.8.7
24K   logrotate_3.8.7-2ubuntu3.debian.tar.xz
4.0K  logrotate_3.8.7-2ubuntu3.dsc
60K   logrotate_3.8.7.orig.tar.gz
1.3M  total
student@ubuntu:/tmp$
```

図8.1 : Ubuntuでのソースパッケージの取得

8.3 DPKGクエリ

DPKGクエリ

- インストールされているすべてのパッケージを一覧表示します :
`$ dpkg -l`
- wgetパッケージのファイルを一覧表示します :
`$ dpkg -L wget`
- インストールされているパッケージとパッケージファイルの情報を表示します :
`$ dpkg -s wget`
`$ dpkg -l webfs_1.21+ds1-8_amd64.deb`
- パッケージファイル内のファイルを一覧表示する :
`$ dpkg -c webfs_1.21+ds1-8_amd64.deb`
- どのパッケージが `/etc/init/networking.conf` を所有しているかを表示します :
`$ dpkg -S /etc/init/networking.conf`
- インストールされたパッケージの整合性を

特定のパッケージに関する情報を得たり、インストール後に変更されたファイルを探したりすることが重要な場合があります。例えば、特定のパッケージのどのバージョンがインストールされているかを確認するためです :

```
dpkg -s dpkg | grep -i バージョン
```

```
バージョン : 1.19.7ubuntu3.2
```

をUbuntu 20.04にインストールしました。

追加の引数がない場合、dpkg -Vはシステム上のすべてのパッケージを検証します :

```
$ sudo dpkg -V
```

```
?5??... c /etc/logrotate.conf
?5?????? c /etc/logrotate.d/apt
?5??... c /etc/logrotate.d/bootlog
?5??... c /etc/logrotate.d/rsyslog
?5??... c /etc/logrotate.d/ufw
?5??... c /etc/logrotate.d/apport
?5??... c /etc/logrotate.d/speech-dispatcher
?5????c /etc/logrotate.d/unattended-upgrades
----
```

出力の解釈については、**man**ページを参照してください。

8.4 インストール/アップグレード/アンインストール

インストール/アップグレード/アン

- **foobar** パッケージをインストールまたはアップグレードします：
`$ sudo dpkg -i foobar.deb`
- インストールされたパッケージのうち、設定ファイルを除くすべてを削除する：
`sudo dpkg -r パッケージ`
- インストールされているパッケージを、設定ファイルを含めてすべて削除します。
`sudo dpkg -P パッケージ`

iオプション（インストール用）を使用する場合：

- パッケージが現在インストールされていない場合は、インストールされます。
- もし、そのパッケージが現在インストールされているものよりも新しいもの

であれば、アップグレードされます。なお、-Pオプションは**purge**の略です。

8.5 研究室



ビデオデモンストレーションのリソース

`dpkg_demo.mp4`を使用しています。

演習8.0 : dpkg 必須



Debian、Ubuntu、Linux Mintの場合

これらのラボを行うには、Debian、Ubuntu、Linux Mint などの Debian ベースのシステムにアクセスする必要があります。

練習問題8.1 : dpkgの使用

ここでは、Debianパッケージの問い合わせと検証を行うためのいくつかの簡単な操作を行うだけです。

1. ファイル `/etc/logrotate.conf` がどのパッケージに属しているかを確認します。
2. パッケージに含まれるすべてのファイルを含む、パッケージの情報を一覧表示します。
3. パッケージのインストールを確認する。
4. パッケージを外してみてください。

✓ ソリューション8.1

1. `$ dpkg -S logrotate.conf`

```
logrotate です :
/usr/share/man/man5/logrotate.conf.5.gz logrotate :
/etc/logrotate.conf
rsync を実行します : /usr/share/doc/rsync/examples/logrotate.conf.rsync
```

2. `$ dpkg -L logrotate`

```
---
```

3. `$ dpkg -V logrotate`

4. `$ sudo dpkg -r logrotate`

```
dpkg: 依存関係の問題でlogrotateの削除ができない: ubuntu-
standardはlogrotateに依存しているが :
  libvirt-daemon-systemはlogrotateに依存し
  ています。
```

```
dpkg: logrotateパッケージの処理エラー (--remove): 依存
関係の問題 - 削除されない
処理中にエラーが発生しました : logrotate
```

第9章

じどうプログラムツール



9.1	じどうプログラムツール.....	144
9.2	APTユーティリティ	145
9.3	クエリ.....	146
9.4	パッケージのインストール/リムーブ/アップグレード.....	147
9.5	クリーンアップ	148
9.6	研究室.....	149

学習目標

このセッションが終わるころには、次のことができるようになっているはずです：

- APTとは何かを説明する。
- **APT**の主要なユーティリティプログラムを定義する。
- パッケージのクエリーを実行する。
- aptを使ったパッケージのインストール・削除・アップグレード
- パッケージに関するシステム情報をクリーンアップしてコンパクトにする。

9.1 じどうプログラムツール

APTとは？

- アドバンスド・パッケージング・ツールの略
- aptやapt-cacheなどのユーティリティ（下位のdpkgプログラムを呼び出すもの）を含む。
- Debianパッケージ(.debファイル拡張子)で動作します。
- 1つのリポジトリを複数のDebianベースのLinuxディストリビューションで使用することは珍しいことではありません。



apt vs apt-get

ほとんどすべての対話的な目的では、もはや**apt-get**を使う必要はなく、より短い名前の**apt**を使うことができます。しかし、習慣的に**apt-get**を使うこともありますし、スクリプトではapt-getの方がうまく機能します。このコースでは、参照するコマンドがコマンドラインでもスクリプトでも使えるので、**apt-get**を使うことがあります。

Debianベースのシステムで使用する場合、APT (Advanced Packaging Tool) プログラムのセットは、基盤となるdpkgプログラムを使用するためのより高いレベルのインテリジェントサービスを提供し、Red Hatベースのシステムでのdnfと同じ役割を果たします。主なユーティリティはaptとapt-cacheです。パッケージのインストール、アップデート、削除の際に依存関係を自動的に解決することができます。外部のソフトウェアリポジトリにアクセスし、同期をとり、必要に応じてソフトウェアを取得し、インストールすることができます。

もう一度言いますが、synapticやUbuntu Software Centerなどの（コンピュータ上の）グラフィカルインターフェースや、aptitudeなどのAPTの古いフロントエンドは無視するつもりです。

9.2 APTユーティリティ

apt, apt-get, apt-cache, など。

- **Debian Linux** 用のパッケージを扱うコマンドラインツール
 - 個々のパッケージのインストール/管理
 - アップグレードパッケージ
 - アップグレードディストリビューション
- **dpkg**のフロントエンド
- リポジトリからパッケージを取得することも可能です。
- パックエイジを検索、検討、ダウンロードするための優れたリソースです：

<https://www.debian.org/distrib/packages>

<https://packages.ubuntu.com>

これらのユーティリティは、パッケージ管理のための **APT** コマンドラインツールです。これらのユーティリティは、個々のパッケージまたはシステム全体のインストール、管理、アップグレードに使用でき、困難な作業であるディストリビューションを完全に新しいリリースにアップグレードすることも可能です。

aptが**rpm**ファイルを扱えるようにする（不完全な）拡張機能もある

。dnfやzypperのように、**APT**は複数のリモートリポジトリを扱うことができます。

9.3 クエリ

クエリ

- アプトキャッシュ

```
apt-cache search  
apt-cache show  
apt-cache showpkg  
$ apt-cache depends
```

- アプトファイル

```
$ apt-file search  
$ apt-file list
```

- のように、先に**apt-file**をインストールし、データベースを更新する必要があるかもしれません：

```
$ sudo apt install apt-file  
sudo apt-file update
```

```
$ apt-cache search apache2
```

リポジトリを検索して、**apache2**という名前のパッケージを探します。

```
$ apt-cache show apache2
```

apache2パッケージの基本情報を表示します。

```
$ apt-cache showpkg apache2
```

apache2パッケージの詳細情報を表示します。

```
$ apt-cache depends apache2
```

apache2の依存パッケージをすべてリストアップします。

```
$ apt-file search apache2.conf
```

リポジトリを検索して、**apache2.conf**という名前のファイルを探します。

```
$ apt-file list apache2
```

apache2パッケージの全ファイルをリストアップします。

9.4 パッケージのインストール／リムーブ／アップグレード

パッケージのインストール／リムーブ／アップグレード

- パッケージのインストールと削除を行います：

```
sudo apt install glibc
sudo apt remove glibc
sudo apt --purge remove glibc
```

- リポジトリデータベースを更新し、システムまたはパッケージをアップグレードします：

```
$ sudo apt update
sudo apt upgrade
sudo apt dist-upgrade
```



個別パッケージのアップグレードは不可

dnfやzypperとは異なり、個々のパッケージを更新/アップグレードすることではなく、パッケージリスト全体を単位として扱います。

`$ sudo apt install [パッケージ]`をインストールします。

新しいパッケージのインストールや、既にインストールされているパッケージのアップデートを行う際に使用します。

`$ sudo apt remove [パッケージ]`.

システムからパッケージを削除するために使用します。この場合、設定ファイルは削除されません。

`$ sudo apt --purge remove [パッケージ]`を削除します。

システムからパッケージとその設定ファイルを削除するために使用します。

`$ sudo apt update`

パッケージインデックスファイルとそのソースを同期させるために使用されます。利用可能なパッケージのインデックスは、`/etc/apt/sources.list` で指定された場所から取得されます。

`sudo apt upgrade`

`sudo apt dist-upgrade`

アップグレードは、すでにインストールされているパッケージに対して、利用可能なすべてのアップデートを適用するために使用されます。dist-upgradeは、一般的に誤解されているように、まったく新しいディストリビューションバージョンにアップデートしません。

dnfやzypperとは異なり、アップグレードの前にアップデートを行う必要があるため、これらのユーティリティの習慣的なユーザーには混乱が生じる可能性があることに注意してください。

9.5 クリーンアップ

クリーンアップ

- 古いLinuxなど、不要になったパッケージは処分する。
カーネルバージョン
`sudo apt autoremove`
- キャッシュファイルや、インストールしたパッケージファイルのアーカイブを削除します：
`$ sudo apt clean`
これにより、スペースを大幅に節約することができます。

9.6 研究室



ビデオデモンストレーションのリソース

パッケージの使い方 [_gui_ubuntu_demo.mp4](#)



演習9.0 : dpkg 必須



Debian、Ubuntu、Linux Mintの場合

これらのラボを行うには、Debian、Ubuntu、Linux Mint などの Debian ベースのシステムにアクセスする必要があります。



練習9.1 : APTの基本コマンド

1. お使いのシステムで利用可能なアップデートがあるかどうかを確認します。
2. インストールされているカーネル関連パッケージをすべてリストアップし、インストール済みまたは利用可能なパッケージをすべてリストアップします。
3. `apache2-dev` パッケージ、またはまだインストールしていないものをインストールしてください。簡単なことをする :

```
$ apt-cache pkgnames
```

は完全なリストを見ることができます。リストを絞り込むためにワイルドカードの引数を与えたい場合もあります。



解決策 9.1

1. まず、パッケージのインデックスファイルをリモートリポジトリと同期させます :

```
$ sudo apt update
```

実際にアップグレードするために :

```
sudo apt upgrade
```

```
sudo apt -u upgrade
```

(先に述べたようにdist-upgradeを使うこともできます。)最初のフォームだけがインストールを試みます。

2.

```
$ apt-cache search "kernel"
```



```
$ apt-cache search -n "kernel"
```



```
$ apt-cache pkgnames "kernel"
```

2番目と3番目のフォームは、名前にkernelを含むパッケージのみを検索します。

```
$ dpkg --get-selections "**kernel**"
```

を使うと、インストールされているパッケージだけを取得することができます。なお、Debianベースのシステムでは、カーネル関連のパッケージはkernelではなくlinuxを使用する必要があるようです。

3.

```
$ sudo apt install apache2-dev
```



練習9.2 APTを使ったパッケージの情報検索

`apt-cache`と`apt` (`dpkg`ではなく)を使って、findします :

1. 名前または説明文に**bash**への参照を含むすべてのパッケージ。
2. インストールされている、利用可能な**bash**パッケージ。

3. **bash**のパッケージ情報です。
4. **bash**パッケージの依存関係です。

上記で使ったコマンドを、rootと一般ユーザーの両方で試してみてください。何か違いを感じますか？

✓ 解決策 9.2

1. `$ apt-cache search bash`
2. `$ apt-cache search -n bash`
3. `$ apt-cache show bash`
4. `apt-cache depends bash`
`apt-cache rdepends bash`

✍ 演習9.3 APTでパッケージのグループを管理する

APTは、メタパッケージの使用により、dnfと同様にパッケージのグループを管理する機能を提供します。メタパッケージは、インストールと削除が必要な関連パッケージをグループとして集めた仮想パッケージと考えることができます。

利用可能なメタパッケージの一覧を取得します：

```
apt-cache search metapackage
```

```
bacula - ネットワークバックアップサービス - メタパッケージ
bacula-client - ネットワークバックアップサービス - クライ
アントメタパッケージ bacula-server - ネットワークバックア
ップサービス - サーバメタパッケージ
cloud-utils - アップストリーム cloud-utils ソースのインストール用メタパ
ッケージ compiz - OpenGL ウィンドウおよび合成マネージャ
emacs - GNU Emacs エディタ (メタパッケージ)
....
```

そして、通常の単一パッケージのように、簡単にインストールすることができます：

```
$ sudo apt install bacula-client
```

```
パッケージリストを読み込
む...完了 依存関係ツリーの構築
状態情報を読み込む...完了
以下の追加パッケージがインストールされます：
  bacula-common bacula-console bacula-fd bacula-traymonitor
推奨パッケージです：
  bacula-doc kde gnome-desktop-environment 以
下の新しいパッケージがインストールされます：
  bacula-client bacula-common bacula-console bacula-fd bacula-traymonitor
アップグレード0、新規インストール5、削除0、アップグレードなし0。
742kB のアーカイブを取得する必要がある。
この操作の後、1,965kB の追加ディスクスペースが使用されます。続行
しますか？[Y/n] と表示されます。]
```

アンインストールされているメタパッケージを選択し、削除します。

第10章

回転数



10.1	RPM (レッドハットパッケージマネージャー)	152
10.2	パッケージのファイル名	153
10.3	RPMデータベースとヘルパープログラム	154
10.4	クエリ	156
10.5	パッケージの検証	157
10.6	パッケージのインストールと削除	159
10.7	RPMパッケージの更新、アップグレード、フレッシュ化	161
10.8	Linuxカーネルのアップグレード	163
10.9	rpm2archive と rpm2cpio	164
10.10	研究室	165

学習目標

このセッションが終わるころには、次のことができるようになっているはずです：

- RPMシステムがどのように構成され、**RPM**プログラムがどのような主要オペレーションを実現できるかを理解する。
- バイナリーファイルとソース**rpm**ファイルの両方で使用される命名規則を説明する。
- パッケージの照会、検証、インストール、アンインストール、アップグレード、フレッシュアップを行います。
- 新しいカーネルをアップグレードするのではなく、インストールすべき理由を把握する。

10.1 RPM (レッドハットパッケージマネージャー)

RPMを使うメリット

- ソフトウェアパッケージの管理
 - 一つのタスクやサブシステムに関連するファイルの集合体
 - プログラム、データ、ドキュメント、コンフィギュレーション情報が含まれています。
 - また、依存情報を含む場合があります。
 - パッケージは取得しません
- 管理者側のメリット
 - パッケージのインストールやアンインストール（消去）が簡単にできる
 - パッケージが正しくインストールされたかどうかを簡単に確認できる
 - FTPまたはHTTPを使用してパッケージをインストールすることができます。
- デベロッパーにとってのメリット
 - オリジナルの "プリスタイン" ソースを使用することができる

RPMは、Red Hatが開発したパッケージ管理ユーティリティです。(特定のタスクに関連するすべてのファイルが1つのファイルにパッケージ化され、そのファイルをどこでどのようにインストールし、アンインストールするかについての情報が含まれています。開発者がプログラムの新バージョンを作成する場合、通常、新しいRPMパッケージをリリースします。これらのファイルは、他のLinuxディストリビューションでは使用できない可能性があることに注意してください。

システム管理者にとって、RPMはソフトウェアパッケージの管理を容易にする。特定のファイルがどのパッケージから来たのか、どのバージョンのパッケージがインストールされているのか、正しくインストールされたのかを簡単に判断することができます。また、ディスクスペースを確保するために、完全なパッケージを削除することも簡単です。また、RPMは、ドキュメントファイルをパッケージの残りの部分と区別し、システムにドキュメントをインストールするかどうかを選択できるようにします。

RPMは、ソフトウェア開発者の仕事を容易にします。多くの場合、開発者は、別のオペレーティング・システムで正しくコンパイルおよび実行できるようにするために、コードを変更しなければなりません。RPMを使用すると、Linuxでビルドするために必要な変更を、元のソースから切り離しておくことができます。この機能は、ビルドに関連する変更がすべて1つの場所にあるため、新しいバージョンのコードの組み込みを容易にします。また、異なるアーキテクチャ用のLinuxのバージョンを構築することも容易になります。



ネットワークからのインストール

特定の URL が与えられない限り、rpm はネットワーク経由でパッケージを取得しません。絶対パスまたは相対パスを使用して、ローカルマシンからのみインストールします。

しかし、HTTPなどのプロトコルを使って、遠隔地から直接インストールすることもできますし、非推奨のFTPです。

10.2 パッケージのファイル名

パッケージのファイル名

- 標準的なバイナリRPMパッケージの命名形式：

<名前>-<バージョン>-<リリース>-<ディストロ>-<アーキテクチャ>.rpm
sed-4.5-2.el8.x86_64.rpm

- 標準的なソースRPMパッケージの命名形式：

<名前>-<バージョン>-<リリース>-<ディストロ>.
>.src.rpm
sed-4.5-2.el8.src.rpm

RPMパッケージのファイル名は、RPM標準規格 ([http.org](http://www.rpm.org/)) に記載されている、特定の情報を表すフィールドをベースにしています：

[//www.rpm.org/](http://www.rpm.org/))

distroフィールドは、実際にはパッケージの出所となるリポジトリを指定することが多いことに注意してください。

10.3 RPMデータベースとヘルパープログラム

RPMデータベースとヘルパープログラム

- `var/lib/rpm` にある RPM データベース。
- **Berkeley DB**のハッシュファイルとして保存されるデータベースファイル
- このディレクトリの他のヘルパープログラム
- でデータベースの再構築が可能です :

```
sudo rpm --rebuilddb
```

`/var/lib/rpm` はデフォルトのシステムディレクトリで、**Berkeley DB** ハッシュファイルの形式で **RPM** データベースファイルを保持しています。データベースファイルを手動で変更することはできません。更新は**rpmプログラム**を使用してのみ行う必要があります。

rpmプログラムの`-dbpath`オプションで、別のデータベース・ディレクトリを指定することができます。例えば、他のシステムからコピーされた**RPM**データベースを調べるために、このようなことをすることがあります。

`rebuilddb`オプションを使用すると、インストールしたパッケージのヘッダーからデータベースのインデックスを再構築することができます。これはどちらかというところ修復であり、ゼロからの再構築ではありません。

RPMが使用するヘルパープログラムおよびスクリプトは、`/usr/lib/rpm`に存在します。例えば、**RHEL 8** システムの場合、かなりの数が存在します：

```
$ ls /usr/lib/rpm | wc -l
```

```
74
```

ここで、`wc`は出力の行数を報告するものである。

rpmのデフォルト設定を指定するために、`rpmrc`ファイルを作成することができます。デフォルトでは、**rpm**は検索します：

1. `/usr/lib/rpm/rpmrc`
2. `/etc/rpmrc`
3. `~/rpmrc`

を上記の順序で実行します。これらのファイルはすべて読み込まれることに注意してください。**rpm**は、1つのファイルが存在することを発見してもすぐに停止するわけではありません。代替の `rpmrc` ファイルは `--rcfile` オプションで指定することができます。

10.4 クエリ

クエリ

- すべてのrpmクエリには、-qオプションが含まれます。
- qオプションは、他の多くのクエリーオプションと組み合わせることができます：
 - f: ファイルがどのパッケージから来たかを判断することができます。
 - l: 特定のパッケージの内容を一覧表示します。
 - a: システムにインストールされているすべてのパッケージ
 - i: パッケージに関する情報
 - p: パッケージデータベースではなく、パッケージファイルに対してクエリを実行します。

表 10.1 : rpm Query コマンドの例

どのバージョンのパッケージがインストールされるのか？	rpm -q bash
このファイルはどのパッケージから来たのでしょうか？	rpm -qf /bin/bash
このパッケージによってインストールされたファイルは何ですか？	rpm -ql bash
このパッケージに関する情報を表示します。	rpm -qi bash
このパッケージに関する情報を、パッケージデータベースではなく、パッケージファイルから表示します。	rpm -qip foo-1.0.0.1.noarch.rpm
このシステムにインストールされているすべてのパッケージを一覧表示します。	rpm -qa

その他に便利なパラメーターとして、--requiresと--whatprovidesがあります。

requiresオプションは、あるパッケージの前提条件のリストを返し、--whatprovidesオプションは、特定の必要なパッケージを提供するインストール済みのパッケージを表示します。

```
rpm -q --requires bash
```

```
rpm -qp --requires foo-1.0.0-1.noarch.rpm
```

```
rpm -q --whatprovides libc.so.6
```

10.5 パッケージの検証

パッケージの検証

- rpm -V コマンドは、パッケージを検証するために使用します。
- 何も問題がないときは出力されない
 - S : ファイルサイズが異なる
 - M : モードが異なる (パーミッションとファイルタイプ)
 - 5 : MD5の合計が異なる
 - D : デバイスのメジャー/マイナー番号の不一致
 - L : readLinkのパスミスマッチ
 - U : ユーザーの所有権が異なる
 - G : グループ帰属が異なる
 - T : mTime differs
- すべてのパッケージを確認するため :
`$ sudo rpm -Va`

rpmの-Vオプションを使用すると、特定のパッケージのファイルがシステムのRPMデータベースと一致しているかどうかを検証することができます。システム上のすべてのパッケージを検証するには、**rpm -Va**を使用します。

出力（問題がある場合のみ出力されます）では、各文字が、ファイルの属性とデータベースに記録されている属性の値との比較結果を示します。ピリオド）はテストに合格したことを意味し、クエスチョンマーク）はテストが実行できなかったことを意味します（例：ファイルのパーミッションが読み取りを妨げている）。それ以外の場合、この文字は対応する -**verify** テストの失敗を意味する。

何も問題がないときは出力しない。

rpm -V bash

ファイルのサイズ、チェックサム、修正時刻が変更されたことを示す出力。

rpm -V logrotate

```
S.5....T. c /etc/logrotate.conf
```

ファイルが存在しないことを示す出力。

sudo mv /sbin/logrotate /sbin/logrotate_KEEP

rpm -V logrotate

```
S.5...T. c /etc/logrotate.conf  
missing    /usr/sbin/logrotate
```

10.6 パッケージのインストールと削除

パッケージのインストール

- RPMは、パッケージのインストール時にいくつかのタスクを実行します。
 - 依存性チェック
 - コンフリクトチェック
 - インストール前に必要なコマンド
 - 設定ファイルをインテリジェントに扱う
 - パッケージからファイルを解凍し、正しい属性でインストールする。
 - インストール後に必要なコマンド
 - システムRPMデータベースの更新
- rpm -i パッケージをインストールする
`$ sudo rpm -ivh bash-4.4.19-12.el8_0.x86_64`
ここで、-iはインストール、-vは冗長、-hは進捗状況を示すハッシュマークを表示することを意味します。

パッケージの中には、他のパッケージがインストールされていないと正常に動作しないものがあるため、依存関係のチェックが必要です。

競合には、すでにインストールされているパッケージをインストールしようとしたり、古いバージョンを新しいバージョンに重ねてインストールしようとしたりすることが含まれます。パッケージのビルドを行う人は、インストールの前後に特定のタスクを実行するよう指定することができます。

設定ファイルをインストールする際、そのファイルが存在し、前のバージョンのパッケージがインストールされた後に変更された場合、**RPM**は古いバージョンを接尾辞 `.rpm.save` を付けて保存します。これにより、古い設定ファイルに加えた変更を、新しいバージョンのファイルに統合することができます。この機能は、適切に作成された RPM パッケージに依存します。

RPMは、ファイルを適切な場所にインストールするだけでなく、パーミッション、所有権、変更（ビルド）時間などの属性も設定します。

RPMは、パッケージをインストールするたびに、システムデータベースの情報を更新します。競合をチェックする際には、この情報を使用します。

パッケージのアンインストール

- アンインストールに成功した場合、通常、何も出力されません

```
sudo rpm -e system-config-lvm
```

system-config-lvm パッケージがインストールされていません。

- 依存関係に起因するエラー

```
$ sudo rpm --test -e xz
```

エラーが発生しました：依存関係に失敗しました：

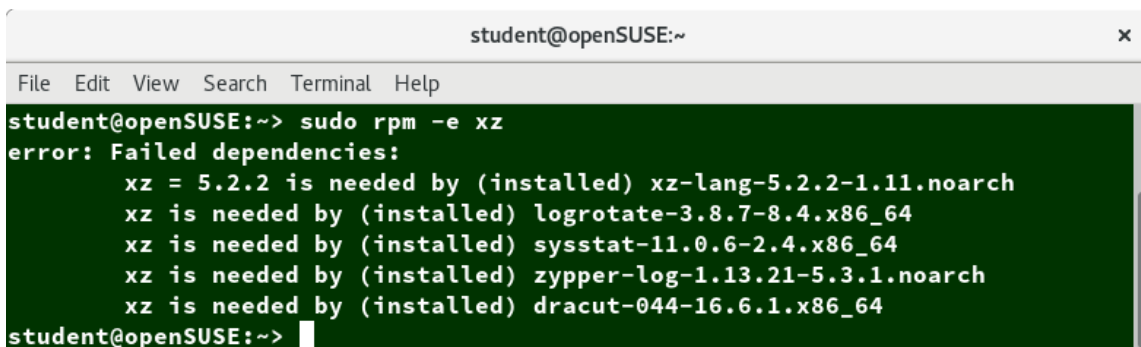
xz は (インストールされた) pcp-5.1.1-4.el8.x86_64
によって必要とされます xz は (インストールされた)
sos-3.9.1-6.el8.noarch によって必要とされます。
xz は (インストールされた) gettext-devel-0.19.8.1-17.el8.x86_64 に
よって必要とされます xz は (インストールされた) libreport-2.9.5-
15.el8.x86_64 によって必要とされます。
xz は (インストールされた) dracut-049-95.git20200804.el8.x86_64
によって必要とされます xz は (インストールされた)
↳ libvirt-daemon-driver-qemu-6.0.0-28.module+el8.3.0+7827+5e65edd7.x86_64 xz
は (インストールした) rpm-build-4.14.3-4.el8.x86_64 で必要です。
xzが必要なのは (インストールされている)
↳ libguestfs-1:1.40.2-25.module+el8.3.0+7421+642fe24f.x86_64 xz
は (インストール) sysstat-11.7.3-5.el8.x86_64 によって必要で

eオプションは、rpmにパッケージをアンインストール(消去)させるものです。通常、アンインストールしようとするパッケージがシステム上の他のパッケージによって必要とされている場合、rpm -eはエラーメッセージを表示して失敗します。

eと一緒に--testオプションを使用すると、実際にアンインストールを実行せずに、アンインストールが成功するか失敗するかを判断することができます。操作が成功した場合、rpmは何も出力しません。より多くの情報を得るには、-vv オプションを追加します。

消去のためのパッケージ引数は、rpmファイル名ではなく、パッケージ名であることを忘れないでください。

重要な注意： rpm/パッケージ自体を削除 (消去/アンインストール) することは絶対にしないでください。この問題を解決するには、OSを再インストールするか、レスキュー環境を使用するしかありません。



```
student@openSUSE:~  
File Edit View Search Terminal Help  
student@openSUSE:~> sudo rpm -e xz  
error: Failed dependencies:  
  xz = 5.2.2 is needed by (installed) xz-lang-5.2.2-1.11.noarch  
  xz is needed by (installed) logrotate-3.8.7-8.4.x86_64  
  xz is needed by (installed) sysstat-11.0.6-2.4.x86_64  
  xz is needed by (installed) zypper-log-1.13.21-5.3.1.noarch  
  xz is needed by (installed) dracut-044-16.6.1.x86_64  
student@openSUSE:~>
```

図 10.1 : RPM パッケージのアンインストール

スクリーンショットの例はopenSUSEシステムで行われたもので、スライドにあるものはRHELシステムで行われたものな

10.7 RPMパッケージの更新、アップグレード、フレッシュ化

パッケージのアップデート

- アップデートすることで、インストールされている場合、オリジナルパッケージを置き換える
`$ rpm -Uvh bash-4.4.19-12.el8.x86_64.rpm`
- 現在インストールされていない場合は、パッケージをインストールする
- 同じファイルを含む2つのパッケージをインストールすることができません。
- `oldpackage` オプションでダウングレードが可能です。

アップグレードの際、すでにインストールされているパッケージは、新しいバージョンのインストール後に削除されます。ただし、オリジナルのインストール時の設定ファイルは、`.rpm`という拡張子をつけて保存されます。

`U`オプションを使用し、パッケージがすでにインストールされていない場合は、単にインストールされ、エラーはありません。新しいRPMパッケージを古いRPMパッケージの上にインストールしようとする、既存のシステムファイルを上書きしようとするため、エラーメッセージが表示されて失敗します。

カーネルパッケージや他のアーキテクチャのライブラリパッケージは、一般的に複数回インストールされることが多い唯一のパッケージです。

`rpm -U`でダウングレードする場合（つまり、現在のバージョンを以前のバージョンに置き換える場合）には、以下のようにコマンドラインに `--oldpackage` オプションを追加します。

フラッシュニングパッケージ

- カレントディレクトリにあるすべてのパッケージをフラッシュ化する :
`$ sudo rpm -Fvh *.rpm`
 - 古いバージョンのパッケージがインストールされている場合、ディレクトリ内の新しいバージョンにアップグレードされます。
 - システム上のバージョンとディレクトリ内のバージョンが同じであれば、何も起こりません。
 - あるパッケージのバージョンがインストールされていない場合、ディレクトリ内のパッケージは無視されます。
- フラッシュニングは、一度に多くのパッチ（アップグレードパッケージ）を適用する場合に有効です。

フラッシュニングは、1つの例外を除いてアップグレードと似ています。アップグレードの際、パッケージがまだ存在しない場合、そのパッケージはロードされます。フラッシュニングの場合、パッケージがまだロードされていなければ、それは単に無視されます。アップグレードもフラッシュニングも、元のパッケージがすでにロードされている場合は、新しいパッケージをインストールします。

Fオプションは、いくつかの新しいパッチをダウンロードしたときに、すでにインストールされているパッケージをアップグレードし、新しいパッケージはインストールしないようにしたい場合に便利です。

10.8 Linuxカーネルのアップグレード

Linuxカーネルのアップグレード

- 常に新しいカーネルをインストールする（アップグレードしない）Red Hatベースのシステムで：

```
$ sudo rpm -ivh kernel-{version}.{arch}.rpm
```

- 問題があれば古いカーネルに戻すことができる
- 新しいカーネルがテストされた後に古いカーネルを削除することができますが、動作が確認されている古いカーネルを少なくとも1つ保持しておくことは常に良いアイデアです。



旧バージョンのパッケージを削除する

もし本当に古いバージョンのカーネルやその他のパッケージを削除したいのであれば、Red Hatベースのシステムではそれが可能です：

```
$ sudo dnf remove --oldinstallonly
```

このコマンドを使用するときには、注意してください。

新しいカーネルをシステムにインストールすると、効果を発揮するために再起動が必要になります（数少ないアップデートの1つです）。カーネルのアップグレード（-U）を行うべきではありません。アップグレードは、現在実行中の古いカーネルを削除してしまうからです。

これ自体はシステムを停止させませんが、再起動後に何らかの問題が発生した場合、古いカーネルはシステムから削除されているため、もはや古いカーネルで再起動することはできません。しかし、(-i)をインストールすると、両方のカーネルが共存し、どちらか一方に起動することを選択できます。

この操作を行うと、**GRUB**設定ファイルは自動的に新しいバージョンに更新され、システムを再設定しない限り、起動時のデフォルトの選択肢となります。

新しいカーネル・バージョンのテストが終わったら、必要でなければ古いバージョンを削除してもかまいませんが、これは必須ではありません。スペースがない場合を除き、1つ以上の古いカーネルを利用可能にしておくことが推奨されます。

10.9 rpm2archive と rpm2cpio

rpm2archiveとrpm2cpioを使用する

- RPMパッケージファイルをアーカイブに変換する :
`$ rpm2archive bash-XXXX.rpm`
は、`bash-XXXX.rpm.tgz`を作成します。
- ワンステップで抽出する :
`$ cat bash-XXXX.rpm | rpm2archive - | tar -xvz`
- RPM パッケージファイルを cpio アーカイブに変換する :
`$ rpm2cpio bash-XXXX.rpm > bash.cpio`
- 1つまたは複数のファイルを抽出します :
`rpm2cpio bash-XXXX.rpm | cpio -ivd bin/bash`
`$ rpm2cpio logrotate-XXXX.rpm | cpio --extract --make-directories`
- パッケージ内のファイルを一覧表示する ;
`$ rpm -qilp bash-XXXX.rpm`

rpm2archiveは、RPMパッケージファイルを**tar**アーカイブに変換するために使用します。引数として `-` が与えられると、入出力は `stdin`と `stdout`です。

これは、古い **rpm2cpio** ユーティリティよりも直接的かつ効率的で、パッケージファイルを **cpio** アーカイブに変換したり、パッケージファイルからファイルを抽出するために使用することができます。その際、すべてのファイルはカレントディレクトリからの相対パスで抽出されます。したがって、ユーザーが `/home/student`ディレクトリにいるときに、上記の例のように `bin/bash`ファイルを抽出すると、`/home/student/ bin/bash`に格納されます。

パッケージ内のファイルをリストアップするだけなら、**rpm**コマンドそのものを使うのが最も簡単な方法です：

```
$ rpm -qilp package.rpm
```


10.10 研究室



ビデオデモンストレーションのリソース

[rpmを使ったデモ.mp4](#)



演習10.0 : 回転数必須



Red Hat、Centos、Fedora、またはopenSUSEの場合

これらのラボを行うには、RHEL、CentOS、Fedora、SUSEなど、RPMベースのシステムにアクセスする必要があります。



練習問題 10.1 : RPMの使用

ここでは、rpmパッケージの問い合わせと検証を行うためのいくつかの簡単な操作を行うだけです。このラボは、Red Hat と SUSE ベースのシステムで同様に機能します。

1. ファイル `/etc/logrotate.conf` がどのパッケージに属しているかを確認します。
2. パッケージに含まれるすべてのファイルを含むパッケージの情報を一覧表示します。
3. パッケージのインストールを確認する。
4. パッケージを外してみてください。



ソリューション 10.1

1. `$ rpm -qf /etc/logrotate.conf`

```
logrotate-3.14.0-3.el8.x86_64
```

2. `rpm -qil logrotate`

```
---
```

なお、この2つのステップを組み合わせた、よりファンシーな形はこうなります：

```
$ rpm -qil $(rpm -qf /etc/logrotate.conf)
```

3. `rpm -V logrotate`

```
..?. ...../etc/cron.daily/logrotate
S.5....T. c /etc/logrotate.conf
```



レッドハットについて

4. RHEL 8の場合：

V.1.3 `sudo rpm -e logrotate`



のエラーが発生しました：依存関係に失敗しました：
logrotate は vsftpd-3.0.3-28.el8.x86_64 によって必要とされる（インストールされる）。
logrotate >= 3.5.2 は rsyslog-8.37.0-13.el8.x86_64 によって必要とされます（インストール済み）。



openSUSEの場合

openSUSE-Leap 15.1にて：

`sudo rpm -e logrotate`

のエラーが発生しました：依存関係に失敗しました：
logrotate is needed by (installed) xdm-1.1.11-lp151.13.2.x86_64
logrotate is needed by (installed) wpa_supplicant-2.6-lp151.4.4.x86_64
logrotate is needed by (installed) chrony-3.2-lp151.8.6.x86_64 logrotate
は (installed) net-snmp-5.7.3-lp151.7.5.x86_64 logrotate は (installed)
syslog-service-2.0-lp151.3.noarch によって必要とされています。
logrotate は (installed) vsftpd-3.0.3-lp151.6.3.x86_64 によって必要と
されています。
logrotate は (installed) libvirt-daemon-5.1.0-lp151.7.6.1.x86_64 によって
必要とされます logrotate は (installed) iscsiui-0.7.8.2-
lp151.13.6.1.x86_64 によって必要とされます mcelog-1.60-
lp151.2.3.1.x86_64 にはインストールされます(インストール)

パッケージの正確な依存関係ツリーは、ディストリビューションとインストールされたソフトウェアの選択の両方に依存することに注意してください。

練習問題10.2: RPMデータベースの再構築

`/var/lib/rpm`に保存されているRPMデータベースが破損する可能性がある条件があります。この演習では、新しいものを構築し、その完全性を検証します。

このラボは、Red HatとSUSEベースのシステムで同様に機能します。

1. リビルド処理で内容が上書きされるため、`/var/lib/rpm`の内容をバックアップしてください。これを怠り、何か問題が発生した場合、大変なことになります。
2. データ基盤を再構築する。
3. ディレクトリの新しい内容とバックアップされた内容を比較します。実際のファイルの内容はバイナリデータなので調べないで、ファイルの数と名前を記録します。
4. システム上のすべての回転数のリストを取得します。このリストを、実際に再構築手順を実行する前に生成されたリストと比較するとよいでしょう。クエリーコマンドが機能したのであれば、新しいデータベースファイルは問題ないはずです。
5. 2つのディレクトリの中身をもう一度比べてみてください。同じファイルがあるか？
6. バックアップ（おそらく100MB程度のサイズ）を削除することもできますが、ゴミ箱に入れる前に、システムが適切に動作していることを確認するために、しばらくの間、バックアップを残しておくといよいでしょう。

ソリューション 10.2

1. `$ cd /var/lib`
`$ sudo cp -a rpm rpm_BACKUP`

2. `sudo rpm --rebuilddb`
3. `$ ls -l rpm rpm_BACKUP`

4. `rpm -qa | tee /tmp/rpm-qa.output`

5. `$ ls -l rpm rpm_BACKUP`

6. おそらく、システムに問題がないことを確認するまでは、このようなことはしないほうがよいでしょう！

`$ sudo rm -rf rpm_BACKUP`

第11章

dnfとyum



11.1	ダーンフ	170
11.2	うん	171
11.3	クエリ	172
11.4	パッケージのインストール/リムーブ/アップグレード	173
11.5	dnfの追加コマンド	174
11.6	研究室	175

学習目標

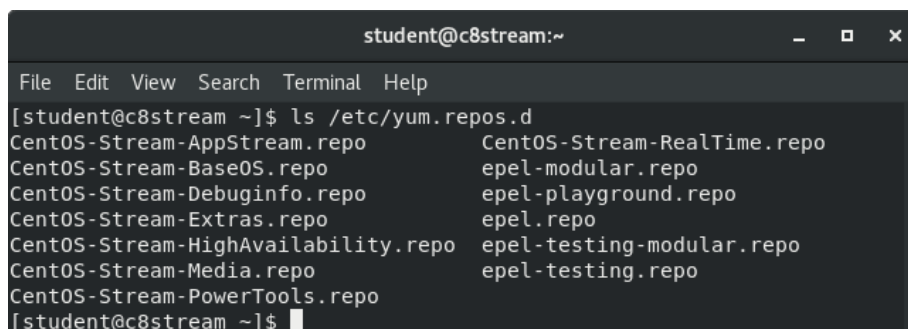
このセッションが終わるころには、次のことができるようになっているはずです：

- パッケージインストーラーとその特徴について説明する。
- dnfが高レベルのパッケージ管理システムとしてどのように機能するかを説明する。
- **dnf**がリポジトリを使用するように設定する。
- dnfが使用できるクエリについて説明する。
- **dnf**を使用してパッケージの検証、インストール、削除、アップグレードを行う。
- 追加コマンドと新しいリポジトリのインストール方法について説明します。

11.1 ダウンフ

dnfとは何ですか？

- RPMのフロントエンドとして使用
- リモートリポジトリからパッケージを取得することができます。
 - 複数の独立したレポジトリに依存することができる
 - [etc/yum.repos.d/](#)にあるリポジトリ設定ファイル。
- 依存関係を解消できる
- RHEL、CentOS、Fedoraなどで使用されています。



```

student@c8stream:~
File Edit View Search Terminal Help
[student@c8stream ~]$ ls /etc/yum.repos.d
CentOS-Stream-AppStream.repo      CentOS-Stream-RealTime.repo
CentOS-Stream-BaseOS.repo        epel-modular.repo
CentOS-Stream-Debuginfo.repo     epel-playground.repo
CentOS-Stream-Extras.repo        epel.repo
CentOS-Stream-HighAvailability.repo epel-testing-modular.repo
CentOS-Stream-Media.repo         epel-testing.repo
CentOS-Stream-PowerTools.repo
[student@c8stream ~]$
  
```

図 11.1 : rpm リポジトリ

dnfは、パッケージ管理に便利な機能を数多く備えています。RPMのフロントエンドであると同時に、1つまたは複数のリモートリポジトリからパッケージを取得する機能を備えています。dnfの最も優れた機能の1つは、依存関係を解決する機能です。

リポジトリ用の設定ファイルは [/etc/yum.repos.d](#) ディレクトリにあり、.repo という拡張子を持っています。非常にシンプルなrepoファイルは次のようなものです：

サンプルレポ

```

[レポ名] です。 ]
name=Description of the repository
baseurl=http://somesystem.com/path/to/repo
enabled=1
gpgcheck=1
  
```

dnfは、パフォーマンスを向上させるために、情報とデータベースをキャッシュします。キャッシュされた情報の一部または全部を削除するには、コマンドを実行することができます：

```
$ dnf clean [ packages | metadata | expire-cache | rpmdb | plugins | all ]。
```

enabledの値を変更するか、-disablerepo somerepoを使用することで、特定のレポの使用をオンまたはオフに切り替えることができます。

と --enablerepo somerepo のオプションがあります。

ここでは、**dnf**をコマンドラインで使用することに集中し、ディストリビューションが提供するグラフィカルなインターフェースについては考慮しないことにする。

11.2 うん

うん

- RHEL/CentOS 7から8への移行時にyumに代わってdnfが導入されました。
- Fedoraはさらに長く使っている
- dnfは後方互換性があります：
 - 一般的なyumコマンドはほぼ全て動作します
- 詳しいドキュメントは、
<https://docs.fedoraproject.org/en-US/quick-docs/dnf/>
<https://dnf.readthedocs.io/en/latest/> をご覧ください
-

yumコマンドを実行しようとする、dnfのいくつかのバージョンでは、これらが非推奨であることを警告し、正しいコマンドを指示する。

もしあなたが経験豊富なyumユーザーなら、dnfは日々のタスクの大部分をケアするyumコマンドのサブセットを受け入れるので、徐々に使い方を学ぶことができます。

11.3 クエリ

クエリ

- パッケージの検索に使用できる

```
$ dnf search
$ dnf info
$ dnf リスト
$ dnf grouplist
$ dnf groupinfo
```

- ファイル検索に使用可能

```
$ dnf provides
```

`$ dnf search` キーワード

キーワードを含むパッケージを検索します。

`dnf info` パッケージ名

パッケージに関する情報を表示する。

`$ dnf list` [インストール済み | アップデート | 利用可能]です。

インストールされているパッケージ、利用可能なパッケージ、またはアップデートされたパッケージを一覧表示します。

`$ dnf grouplist`

インストールされているパッケージグループ、利用可能なパッケージグループ、またはアップデートに関する情報を表示します。

`dnf groupinfo` パッケージグループ

パッケージグループに関する情報を表示します。

`$ dnf provides /path/to/file` を提供します。

ファイルに対するパッケージの所有者を表示します。(ファイル名に少なくとも1つの/を使用する必要があるため、混乱する可能性があることに注意してください)。

11.4 パッケージのインストール／リムーブ／アップグレード

パッケージのインストール／リムーブ／アッ

- コマンドを使用します：

```
$ sudo dnf install  
sudo dnf localinstall  
sudo dnf groupinstall  
$ sudo dnf remove  
$ sudo dnf update  
- .rpmsave  
- .rpmnew
```

`sudo dnf install パッケージ`

は、リポジトリからパッケージをインストールします。また、依存関係を解決してインストールする。

`sudo dnf localinstall パッケージファイル`

は、ローカルのrpmファイルからパッケージをインストールします。

`$ sudo dnf groupinstall 'グループ名'`

は、リポジトリから特定のソフトウェアグループをインストールし、グループ内の各パッケージの依存関係を解決してインストールします。

`sudo dnf remove package`

は、システムからパッケージを削除します。

`$ sudo dnf update [パッケージ]`です。

リポジトリからパッケージを更新します。リストにないパッケージは、すべてのパッケージを更新します。

インストール (またはアップデート) 中に、パッケージが更新された設定ファイルを持つ場合、古い設定ファイルの名前を `.rpmsave` という拡張子で変更します。古い設定ファイルが新しいソフトウェアでも動作する場合は、新しい設定ファイルの名前を `.rpmnew` という拡張子で変更します。これらのファイル名の拡張子を検索して、ファイルの調整を行う必要があるかどうかを確認することができます。

11.5 dnfの追加コマンド

dnfの追加コマンド

```
sudo dnf repolist
```

```
sudo dnf shell
```

```
$ sudo dnf shell [text-file].
```

```
$ sudo dnf install --downloadonly
```

```
sudo dnf history
```

```
$ sudo dnf clean [packages|metadata|expire-cache|rpmdb|plugins|all] です。
```

```
$ sudo dnf list "dnf-plugin*"です。
```

は、追加のdnfプラグインをリストアップします。

```
sudo dnf repolist
```

は、有効なリポジトリの一覧を表示します。

```
sudo dnf shell
```

```
$ sudo dnf shell file.txt
```

は、複数のdnfコマンドを実行するための対話型シェルを提供します。2番目の形式は、file.txtのコマンドを実行するものである。

```
$ sudo dnf install --downloadonly パッケージ
```

あなたのためにパッケージをダウンロードするだけです。パッケージは /var/cache/dnf ディレクトリに保存されます。

```
sudo dnf history
```

システム上のdnfコマンドの履歴を表示し、修正オプションを使用すると、以前のコマンドを元に戻したり、やり直したりすることも可能です。

```
$ sudo dnf clean [packages|metadata|expire-cache|rpmdb|plugins|all] です。
```

var/cache/dnfにローカルに保存されているファイルやメタデータをクリーンアップします。これは、スペースを節約し、古いデータのハウスクリーニングを行うものです。

11.6 研究室

演習11.0 : dnfまたはyum 必須



Red Hat、Centos、またはFedoraの場合

これらのラボを行うには、RHEL/CentOSやFedoraなど、dnfベースのシステムにアクセスする必要があります。RHEL/CentOS 7では、dnfは後方互換性があるため、以下のコマンドと解決策の提案において、yumをdnfに置き換えることができます。

練習問題 11.1 : dnfの基本コマンド

1. お使いのシステムで利用可能なアップデートがあるかどうかを確認します。
2. 特定のパッケージを更新する。
3. インストールされているカーネル関連パッケージをすべてリストアップし、インストール済みまたは利用可能なパッケージをすべてリストアップします。
4. httpd-develパッケージ、またはまだインストールされていないものをインストールします。簡単なことをする：

```
sudo dnf list
```

は完全なリストを見ることができます。リストを絞り込むためにワイルドカードの引数を与えたい場合もあります。

解決策 11.1

1.

```
$ sudo dnf update
```

```
$ sudo dnf check-update
```

```
sudo dnf list updates
```

最初のフォームだけが、インストールを行おうとします。
2.

```
sudo dnf update bash
```
3.

```
$ sudo dnf list installed "kernel*" 
```

です。

```
$ sudo dnf list "kernel*" 
```
4.

```
$ sudo dnf install httpd-devel
```

練習問題 11.2: パッケージの情報を探す

dnfを使い (rpmを直接使うのではなく)、findします：

1. 名前または説明文にbashへの参照を含むすべてのパッケージ。
2. インストールされている、利用可能なbashパッケージ。
3. bashのパッケージ情報です。
4. bashパッケージの依存関係です。

上記でを使用したコマンドを、rootと一般ユーザーの両方で試してみてください。何か違いを感じますか？

✔ 解決策 11.2

**ご注意ください**

ディストリビューションのバージョンによっては、情報を得るだけなのに、以下のコマンドで**sudo**を使わないとパーミッションエラーが出る場合があります。

```
sudo dnf search bash
```

```
sudo dnf list bash
```

```
sudo dnf info bash
```

```
$ sudo dnf deplist bash
```

演習11.3：パッケージのグループを管理する**ご注意ください**

RHEL/CentOSでは、以下のコマンドの一部で**sudo**を使用しないと、情報を取得するだけの場合でもパーミッションエラーが発生することがあります。

dnfは、パッケージのグループを管理する機能を提供します。

1. 次のコマンドを使用して、システムで利用可能なすべてのパッケージグループを一覧表示します：

```
$ dnf grouplist
```

2. Virtualization Hostグループを特定し、コマンドを使用してこのグループに関する情報を生成します。

```
$ dnf groupinfo "仮想化ホスト"
```

3. 使ってインストールします：

```
$ sudo dnf groupinstall "仮想化ホスト"
```

4. 現在システムにインストールされているパッケージグループのうち、不要なものを特定します。のようにdnf groupremoveを使って削除します：

```
$ sudo dnf groupremove "仮想化ホスト"
```

削除を確認するプロンプトが表示されるので、安全にコマンドを入力して動作を確認することができます。

groupremoveは、インストールされたものをすべて削除しないことがあります。これがバグか機能かについては、議論することができます。

練習問題11.4：新しいリポジトリを追加する

その著者によれば（ <http://www.webmin.com/index.htm> ）：

"Webminは、Unixのシステム管理のためのWebベースのインターフェイスです。最新のウェブブラウザを使用して、ユーザーアカウント、Apache、DNS、ファイル共有などを設定することができます。Webminは、以下のようなUnixの設定ファイルを手動で編集する必要性を排除しています。

/etc/passwdを使用し、コンソールやリモートからシステムを管理することができます。"

インストールとアップグレードのためのリポジトリを作成するつもりです。単にダウンロードページに行き、現在のrpmを取得することもできますが、それでは自動的にアップグレードが行われるわけではありません。

1. /etc/yum.repos.d ディレクトリに **webmin.repo** という新しいリポジトリファイルを作成します。それには、以下の内容が含まれているはずです：

**ウェブミンレポ**

【ウェブミン】の場合】

```
name=Webmin Distribution Neutral
```

```
baseurl=http://download.webmin.com/download/yum
```



```
mirrorlist=http://download.webmin.com/download/yum/mirrorlist
enabled=1
gpgcheck=0
```

2. webminパッケージをインストールします。

```
sudo dnf install webmin
```


第12章

ザッパー



12.1	ザッパー.....	180
12.2	クエリ.....	181
12.3	パッケージのインストール/リムーブ/アップグレード.....	182
12.4	その他のzypperコマンド.....	183
12.5	研究室.....	184

学習目標

このセッションが終わるころには、次のことができるようになっているはずです：

- zypperとは何かを説明する。
- zypper が使用できるクエリについて説明します。
- zypperを使ったパッケージのインストール、削除、アップグレード。
- 追加でより高度なzypperコマンドを学ぶことができます。

12.1 ザイパー

ザイパー

- **SUSE Linux**および**openSUSE**のパッケージをインストールおよび管理するためのコマンドラインツール。
- リポジトリからパッケージを取得する
- 依存関係を解消する
- **RPM**パッケージと連動
- コマンドや機能は**dnf**に似ています。

SUSEベースのシステムで使用する場合、**zypper**プログラムは、基盤となる**rpm**プログラムを使用するためのより高いレベルのインテリジェントサービスを提供し、**Red Hat**ベースのシステムで**dnf/yum**と同じ役割を果たす。パッケージのインストール、アップデート、削除の際に、依存関係を自動的に解決することができます。また、外部のソフトウェアリポジトリにアクセスし、同期をとり、必要に応じてソフトウェアの取得やインストールを行います。

zypperは、**SUSE Linux**および**openSUSE**でパッケージをインストールおよび管理するためのコマンドラインツールです。と非常によく似ています。

dnfの機能、そして基本的なコマンドの構文まで、**rpm**パッケージと連動しています。

12.2 クエリ

クエリ

- コマンド

`zypper list-updates`

`zypper repos`

`$ zypper search`

`$ zypper info`

`$ zypper search --provides /usr/bin/firefox`

`zypper list-updates`

は、利用可能なアップデートの一覧を表示します。

`zypper repos`

は、利用可能なリポジトリの一覧を表示します。

`$ zypper search <string>`

は、リポジトリから文字列を検索します。

`zypper info firefox`

は、パッケージに関する情報を一覧表示します。

`$ zypper search --provides /usr/bin/firefox`

は、リポジトリを検索して、あるファイルを提供するパッケージが何かを表示します。

12.3 パッケージのインストール／リムーブ／アップグレード

パッケージのインストール／リムーブ／アッ

- zypper install
sudo zypper install firefox
- zypper update
\$ sudo zypper --non-interactive update
sudo zypper update firefox
- zypper remove
sudo zypper remove firefox
- --non-interactive (または -n) は、確認を求めない。

sudo zypper install firefox

システムにパッケージをインストールまたはアップデートします。

sudo zypper --non-interactive install firefox

は、インストールやアップグレードの確認は求めません。これはスクリプトに便利です。

sudo zypper update

は、システム上のすべてのパッケージをリポジトリから更新します。

\$ sudo zypper --non-interactive update

システム上のすべてのパッケージをリポジトリから更新しますが、確認は求めません。これはスクリプトに便利です。

sudo zypper remove firefox

は、システムからパッケージを削除します。

dnfと同様に、削除されるパッケージを必要とするパッケージも削除されてしまうので、削除コマンドには注意が必要です。

12.4 その他のzypperコマンド

その他のzypperコマンド

- **zypper** コマンドラインシェルインタフェースを入力します :
`sudo zypper shell`
- 新しいリポジトリを追加します :
`sudo zypper addrepo`
- リポジトリを削除する :
`sudo zypper removerepo`
- `var/cache/zypp` をクリーンアップしてスペースを確保する :
`$ sudo zypper clean [--all]` です。

時には、いくつかの**zypper**コマンドを連続して実行する必要があります。各コマンドのすべてのデータベースを読み直すことを避けるために、次のようにシェルモードで**zypper**を実行することができます :

```
sudo zypper shell
> インストールバッシュ
...
> 出口
```

zypperは**readline**ライブラリをサポートしているため、**bash**シェルで利用可能な**zypper**シェルでのすべての同じコマンドライン編集機能を使用することができます。

新しいリポジトリを追加するには

`sudo zypper addrepo URIエイリアス`

は、指定された URI にあり、指定されたエイリアスを使用します

。リストからリポジトリを削除するには

`$ sudo zypper removerepo alias`

削除したいレポのエイリアスを使用します。

12.5 研究室



ビデオデモンストレーションのリソース

ユースドヤストデモ.mp4

演習12.0 : zypper必須



openSUSEの場合

これらのラボを行うには、SUSEやopenSUSEなど、zypperベースのシステムにアクセスする必要があります。

練習問題12.1 : 基本的なzypperコマンド

1. お使いのシステムで利用可能なアップデートがあるかどうかを確認します。
2. 特定のパッケージを更新する。
3. システムが認識している、有効または無効のすべてのリポジトリをリストアップします。
4. インストールされているカーネル関連パッケージをすべてリストアップし、インストール済みまたは利用可能なパッケージをすべてリストアップします。
5. **apache2-devel**パッケージ、またはまだインストールされていないものをインストールします。(注意 **httpd**は**SUSE**では**apache2**です。
のシステムです)シンプルにやること :
`sudo zypper search`
は完全なリストを見ることができます。リストを絞り込むためにワイルドカードの引数を与えたい場合もあります。

✓ ソリューション12.1

1. `zypper list-updates`
2. `sudo zypper update bash`
3. `zypper repos`
4. `$ zypper search -i kernel`
`zypper search kernel`
5. `$ sudo zypper install apache2-devel`

練習問題12.2: zypperを使用してパッケージに関する情報を検索する

zypperを使い (rpmを直接使うのではなく)、findする :

1. 名前または説明文に**bash**への参照を含むすべてのパッケージ。

2. インストールされている、利用可能な**bash**パッケージ。
3. **bash**のパッケージ情報です。
4. **bash**パッケージの依存関係です。

上記で使ったコマンドを、rootと一般ユーザーの両方で試してみてください。何か違いを感じますか？

✓ 解答 12.2

1. `$ zypper search -d bash`

dオプションがない場合は、実際の名前にbashが含まれるパッケージのみが報告されます。パッケージのzypper infoを実行して、bashがどこで言及されているかを確認する必要があるかもしれません。

2. `zypper search bash`

3. `zypper info bash`

4. `$ zypper info --requires bash`

を実行すると、**bash**が必要とするファイルのリストが表示されます。**bash**がインストールされているかどうか依存するものを確認する最も簡単な方法は、おそらく次のようにすることです。

`sudo zypper remove --dry-run bash`

この演習では、bashはシステムにとって不可欠であるため、不適切な選択です。

第13章

GITの紹介



リビジョンコントロールシステムの概念を説明し、**git**が歴史的にどのような進化を遂げてきたかを説明します。**git**で使用可能なグラフィカル・インターフェースについて説明します。重要なドキュメントの情報源も紹介します。

13.1	リビジョンコントロール	188
13.2	コードの出どころを知る : DCOとCLA.....	190
13.3	利用可能な改訂制御システム	192
13.4	グラフィカルなインターフェイス	193
13.5	ドキュメンテーション	195
13.6	研究室.....	197

学習目標

このセッションが終わるころには、次のことができるようになっているはずです：

- 改訂管理システムが重要である理由を説明する。
- Linuxカーネルでの**git**の使用など、分散型開発を行うプロジェクトで果たす役割を理解することができる。
- CLAや**DCO**の活用により、コード一行一行の出所を知ることがなぜ重要なのかを把握する。
- ソース管理システムの歴史を調査し、現在利用可能なオープンソースシステムをリストアップする。
- **git**のグラフィカルインターフェースがわかる。
- **git**の使用に関する詳細なドキュメントを見つけることができる。

187

13.1 リビジョンコントロール

なぜ改訂管理システムが必要なのか

私たちがリビジョンコントロールを必要とする理由は様々ですが、その中でも特に重要な要件は以下の通りです：

- 時間と共に変化が蓄積されていく：
 - バグとセキュリティの修正
 - 製品の強化
- 変更点を把握する必要がある：
 - と、必要に応じて元に戻すことができる
- 明確な改訂履歴が必要

プロジェクトに取り組んでいると、変化が蓄積されていきます。バグを修正し、新しい機能を追加し、最適化することで、コードベースは最初に作ったところから大きく乖離していきます。

時には、機能を追加したりバグを修正したりして、すべてが完璧に動作していると思っていても、ユーザー（おそらく自分自身でも）が予期しない方法でアプリケーションを使用したときに、そんな希望的観測が現実のものとなってしまいます。あるいは、より高速なハードウェアと複数のCPUを備えた新しいプラットフォームに移植する際に、あり得ないほど稀なレースコンディションに遭遇することもあります。

このような場合、問題が発生した時点からかなり進んでいる可能性があり、開発の中でうまくいかなかった箇所を特定し、問題を修正し、他のすべての変更を現在の状態まで進める必要があります。

特に、自分一人でプロジェクトやコードベースに取り組んでいる場合、開発者なら誰もがどこかしらでこの点に杜撰な態度をとっているはずですが、モジュラーコンポーネントを多用したクリーンでインテリジェントなデザイン、一度修正した部分を分解して組み立てるのが容易なデザインに代わるものではありませんが、効率的な開発サイクルを大幅に促進するために使用できるツールもあります。

分散型開発

- 多くの貢献者（Linuxカーネルでは数千人）がいると、問題が複雑になります。
- 中央官庁はあるのでしょうか？あるとすれば、誰ですか？
- 複数の開発者が同じコードに携わり、コンフリクトが発生した場合はどうするか？どのように解決するのでしょうか？
- プロジェクトの履歴は、どのように一行一行管理されているのですか？コードに対する責任は誰が負うのか？
 - メンテナンス用
 - 真正性宣言の場合
 - 法律やライセンスに関する考慮のため
- これまでにも多くのリビジョンコントロールシステムが存在してきました。ここでは、**git**を中心に解説します。

最もシンプルな方法は、作品のバックアップスナップショットを定期的に保存しておき、問題が発生したときにそれを元に戻すだけです。これは誰もがやったことがあるはずですが、これは手作業であり、プロジェクトの規模が大きくなるにつれて、ますます扱いにくくなる。現在のコードベースと以前のコードベースとの差分を取ることでありますが、完全なスナップショットではなく、変更履歴と差分を保存すればよいのではないのでしょうか。そうすれば、ディスクスペースを節約できます。さらに重要なことは、終点と始点から推測するのではなく、変更を直接追跡できることです。

また、他の開発者がプロジェクトに加わってくると、マニュアル方式では管理が難しくなってきます：

- 一人の開発者が、実際の変更に対応する中心的な存在であり続けなければなりません。
- 結果を混乱させたくなければ、他の開発者が直接変更をコミットすることはできません。

開発者が増えれば増えるほど、同時投稿を管理するのは難しくなります。

Linuxカーネルのような大規模な分散型プロジェクトでは、何千人ものコントリビューターが参加することがあり、すべての知恵を集めた中央の権威あるリポジトリという概念さえも障害になることがあります。

アップデートを整理し、協力を促進するために、ソース管理のためのさまざまなスキームが利用可能です。このようなプログラムの標準的な機能には、変更の正確な履歴（ログ）を保持する機能、以前のリリースにバックアップする機能、複数の開発者による競合する可能性のあるアップデートを調整する機能などがあります。

13.2 コードの出どころを知る：DCOとCLA

DCOとCLA

- コードの起源は、出所と保守の両方のために慎重に文書化する必要があります。
- **CLA**（コントリビューターライセンス契約）
旧来の方式で、貢献者全員が契約書にサインをする
- **DCO**（デベロッパー証明書）について
2004年にLinux Foundationによって導入された。すべてのコントリビューターは、以下の条件を満たす必要がある。
投稿にサインをする
- **CLA**は1回だけ、**DCO**はコミットごとに使用
- <https://wiki.linuxfoundation.org/dco> をご参照ください。

オープンソースプロジェクトでは、貢献されたコードがどこから来たのかを文書化して理解することが非常に重要です。これは、以下のような複数の理由から重要です：

- 質問や修正があった場合、一般的にはこの人たちが回答するか、その作業を行うか、少なくとも受け入れ前に確認する必要があります。
- コードの合法的な出所を知ること。誰かがうっかり、あるいは意図的に、別の文脈で使われていたかもしれないライセンスに違反するコードを使った場合、大きな問題になることがあります。この場合、誰に責任を負わせるべきかを知ることが重要です。

これらの問題を処理するための2つの一般的な方法は、**貢献者ライセンス契約 (CLA)** または **開発者起源証明書 (DCO)** のいずれかを使用することです。あるプロジェクトにとってどちらがより適切かについては、議論があるところです。しかし、多くの確立されたプロジェクトは、CLAからDCOに移行し、運営上の完全性文書としています。

CLA (旧来の方法) の場合、貢献者はプロジェクトごとに異なる契約書に署名しなければなりません。例えば、著作権や特許のライセンスを付与する場合としない場合があります。

DCOは2004年にLinux Foundationによって導入されたもので、よりシンプルで軽量なものとなっています。すべてのgitコミットはこの方式でサインオフしなければならない。

どちらの方法でも、ソフトウェアリポジトリ (GitHubやGitLabなど) をホストするサーバーは、すべての貢献がCLAを備えているか、DCOの要求に従ってサインオフされているかを確認するためのチェックを行うことができます。

一般的に**CLA**は一回きりの操作であるのに対し、**DCO**は貢献のたびにサインオフが必要ですが、これは些細なことです (つまり、各コミット操作に-sオプションを追加する)。

DCOの使い方については、<https://wiki.linuxfoundation.org/dco> に詳しくまとめられています。sオプションを使用すると、各コミットには次のようなメッセージが埋め込まれます：

```
サインオフバイ：あるデベロッパー
<some-developer@example.com>
```

13.3 利用可能な改訂制御システム

利用可能なソースコントロールシス

- リビジョン/ソース管理システムの長い歴史。最も古いものは以下の通りです：
 - エスシーシーエス
 - RCS
 - シーバス
- よりモダンなオープンソースシステムを：
 - サブバージョン
 - バザール
 - モノトーン
 - メルカリ
 - PRCS

リビジョン管理システムは、開発者が仕事をしている限り、利用可能でした。Unix系プラットフォームで広く利用できる最初のシステムは、おそらく1970年代初頭に遡る**SCCS**（Source Code Control **S**ystem）でしょう。

プロプライエタリ、オープンに関わらず、利用可能な製品には事欠きません。GPLライセンスでリリースされている製品の簡単なリストには、以下のものがあります：

表13.1：利用可能なソースコントロールシステム

製品	URL
RCS	http://gnu.org/software/rcs/
シーバス	http://savannah.nongnu.org/projects/cvs
サブバージョン	https://subversion.apache.org/
ギット	https://kernel.org/pub/software/scm/git/
バザール	http://bazaar.canonical.com
モノトーン	https://monotone.ca
メルカリ	https://mercurial-scm.org
PRCS	http://prcs.sourceforge.net

また、長いリストが存在する独自製品については、説明を省略します。

13.4 グラフィカルなインターフェイス

利用可能なGUI

- 複数のグラフィカルインターフェースが存在します：

インターフェース	商品説明
ギットギ	これを使うには、メインのリポジトリディレクトリから <code>git gui</code> または <code>git-gui</code> と入力して起動する必要があります。これにより、履歴の閲覧、変更、コミットを行うことができます、リモートリポジトリでの作業、ブランチの比較など。
ギットク	このリポジトリブラウザは、実際には <code>git-gui</code> の中から呼び出すことができ、多少はプロジェクトの履歴を見ることに特化した、より古いインターフェイスです。
クジット	Webサーバーと連携して設置することで、非常に効率的なリポジトリ閲覧が可能です。その能力の良い例として、 https://kernel.org を試してみてください。
ギットウェブ	<code>cgit</code> よりも古いインターフェイスで、似たような方法で動作する。

- また、GitHubやGitLabなどが提供するWebインターフェイスで作業するユーザーも

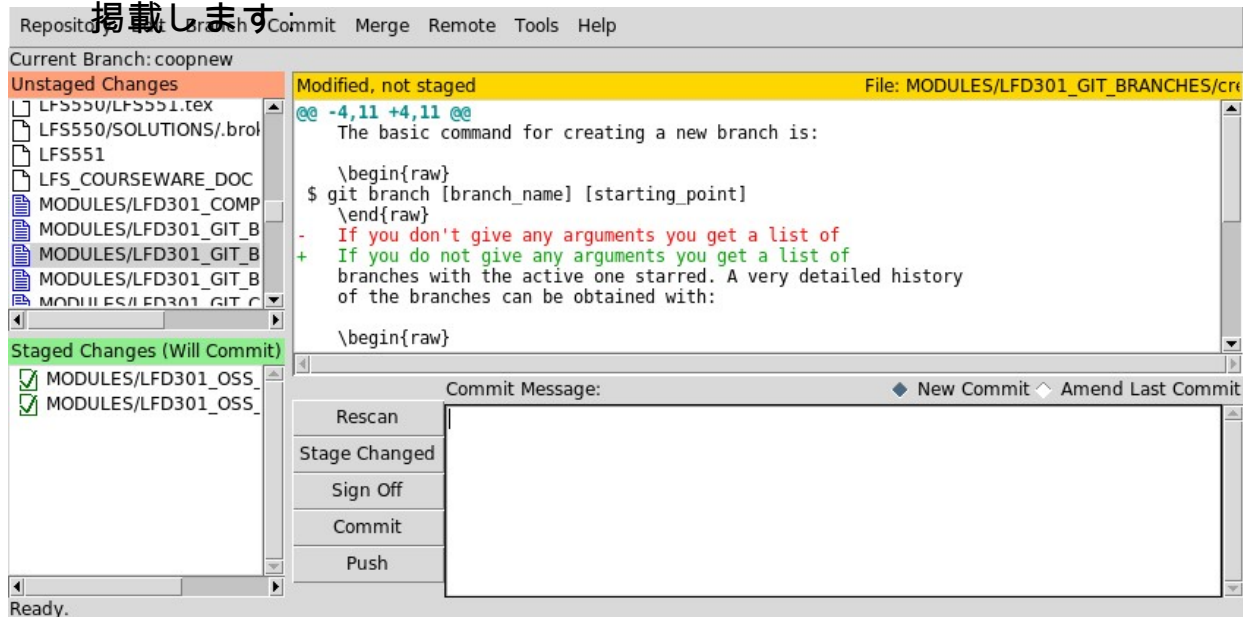
`git` リポジトリを閲覧・管理するために使用できるグラフィカルインターフェースは数多くあります。このクラスでは、これらを集中的に扱うことはしませんが、使用を控えるわけではありません。

このように強調する理由は、GUIが登場しては消え、ユーザーインターフェイスを変えるのに対し、コマンドラインベースのツールは大きな改訂を受ける可能性が低いからです。さらに、GUIを使用している場合でも、Gitの内部で何が起きているのかを理解すればするほど、`git`をより有効に活用することができます。

グラフィカル・インターフェースのデモンストレーションを行うこともありますが、それを中心に授業をデザインすることはありません。

ギットギ

- 例として、**git gui** (**git gui** と入力して実行) のスナップショットを掲載します。



- **git-gui**は、あなたのローカルマシン上のコードに焦点を当てたグラフィカル・ユーザー・インターフェースです。
- 変更のコミット、ブランチの作成、マージ、フェッチ、リモートレポへのプッシュに使用できます。
- **git-gui**はよく知られた**Tcl/Tk**フレームワークで書かれているため、ほとんどすべてのオペレーティングシステムで利用可能です。
- **man git-gui**というコマンドで、その操作の詳細を知ることができます。

13.5 ドキュメンテーション

ローカルドキュメント

- すべてのコマンドまたは特定のコマンドに関するヘルプを表示します :
`$ git help`
`git help コマンド`
- 特定のコマンドに関するヘルプは、この2つのコマンドが同等です。
`git help commit`
`$ man git-commit`

`git`をインストールすると、ローカルマシンに多くのドキュメントがもたらされます。ただ入力するだけです :

```
$ git help
```

は、基本的なコマンドを提供します。そのうちのどれか1つに関する詳細なヘルプを得ることができます :

```
git help コマンド
```

特定のコマンドに関するヘルプを得るには、2つの方法があります :

```
git help commit
```

```
$ man git-commit
```

を実行した結果、従来の`man`ページが出来上がりました。

その他の資料

- <https://kernel.org/pub/software/scm/git/docs/user-manual.html> : 公式Gitユーザズマニュアル。
- <https://git-scm.com> **git** のトップページ
- **Gitによるバージョン管理**、Jon Loeliger著、pub.O'Reilly, 2009.
- **Gitによるバージョン管理** : Jon Loeliger and Matthew Mc- Cullough 著「Gitによるバージョン管理 : 共同作業型ソフトウェア開発のための強力なツールとテクニック」、pub.O'Reilly, 2012.
- **Git入門** (James Bottomley著)
<https://training.linuxfoundation.org/resources/webinars/introduction-to-git/>
- **GIT Started with Community Software** by Jerry Cooperstein,
<https://training.linuxfoundation.org/cm/LFS307/ESC.pdf>.

ドキュメントの2大ソースは

- <https://kernel.org/pub/software/scm/git/docs/user-manual.html> : 公式のGitユーザズ・マニュアルです。他にも多くの有用な文書が、同じウェブサイトにあるか、そこからリンクされています。
この文書は簡潔ですが、権威があり、非常に有用です。
- **Gitによるバージョン管理**、Jon Loeliger著、pub.O'Reilly, 2009.
Gitによるバージョン管理 : 共同ソフトウェア開発のための強力なツールとテクニック、Jon LoeligerとMatthew McCullough著、pub.O'Reilly, 2012.
本書（およびその第2版）は、**git**の使用に関するいくつかの本のうちの1つですが、最も最新で、最も徹底したものです。このクラスでは、時間的な制約から、ざっとしか扱われないトピックが多くあります。この本は、実際の問題や経験を前面に押し出した多くの例で、非常に詳しく説明しています。

ドキュメントの全体的なソースとしては、**git**のホームページ（<https://git-scm.com>）に、豊富なドキュメントとサンプルが掲載されています。

Jerry Cooperstein氏によるスライドプレゼンテーション「**GIT Started with Community Software** presented at Chicago Embedded Systems Conference in 2010」については、ダウンロードしてください :

<https://training.linuxfoundation.org/cm/LFS307/ESC.pdf>。

James Bottomleyによる無料オンラインウェビナー「**Git入門**」については、以下をご覧ください :

<https://training.linuxfoundation.org/resources/webinars/introduction-to-git/>。

13.6 研究室



ビデオデモンストレーションのリソース

[git_guiを使う.mp4](#)



練習13.1 : git.kernel.orgでのcgitの例

- **cgit**リポジトリブラウザの包括的な実装を見るには、<https://git.kernel.org>。**git**そのものを含め、**Linux**カーネルコミュニティから発展した多くのプロジェクトのホストとして機能しています。
- プロジェクトまでスクロールして、サマリーをクリックして、特定のリポジトリを見ます。メインとなる**Linux**のカーネルリポジトリです。履歴やパッチなどを見てください。
- これは、同じく**cgit**を搭載している<https://kernel.org> を見るのとは違います。

第14章

Gitの使用法：一例



14.1	基本コマンド.....	200
14.2	簡単な例.....	203
14.3	コミットメントへのサインオフ	209
14.4	しゅとけん.....	210
14.5	研究室.....	212

学習目標

このセッションが終わるころには、次のことができるようになっているはずです：

- **clone**、**add**、**commit**、**merge**、**pull**、**push**など、**git**で使用する主要なコマンドを熟知していること。
- **git help**で特定のコマンドに関するシステム上のヘルプを得ることができます。
- リポジトリの作成と初期化、リポジトリへの変更、共同作業への公開など、シンプルながら完全な例で作業します。
- 責任者を確定する貢献への署名の方法を知っている。
- プライマリオーソリティブランチの名前を**master**ではなく**main**（または類似のもの）にする理由を説明してくだ

14.1 基本コマンド

いくつかの必須コマンド

```
$ git --バージョン  
$ git help [サブコマンド]です。  
$ git clone  
git branch  
$ git status  
$ git add  
$ git commit  
$ git merge  
$ git pull  
$ git push
```

一緒にインストールされているgitのバージョンを確認することができます：

```
$ git --バージョン
```

```
git バージョン 2.27.0
```

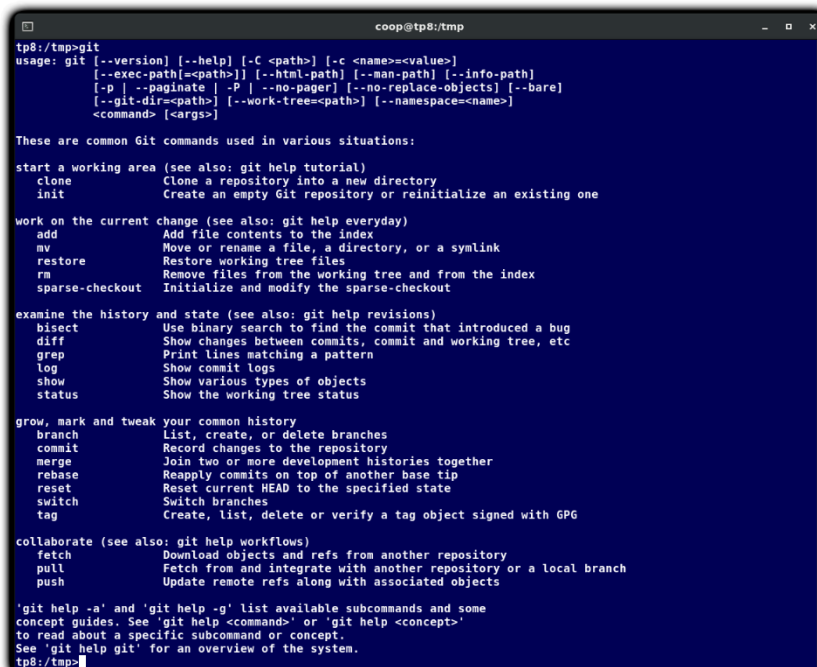
どのサブコマンドについても、**man**ページという形で詳細なヘルプ情報を得ることができます：

```
$ git help [サブコマンド]です。
```

例えば、次の2つの文は同じ結果をもたらします：

```
git help status  
man git-status
```

基本的なgitコマンド



```
coop@tp8:/tmp
tp8:/tmp>git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
       [--exec-path<path>] [--html-path] [--man-path] [--info-path]
       [-p] [--paginate] [-P] [--no-pager] [--no-replace-objects] [--bare]
       [--git-dir<path>] [--work-tree<path>] [--namespace=<name>]
       <command> [<args>]

These are common Git commands used in various situations:


start a working area (see also: git help tutorial)
   clone      Clone a repository into a new directory
   init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add        Add file contents to the index
   mv         Move or rename a file, a directory, or a symlink
   restore    Restore working tree files
   rm         Remove files from the working tree and from the index
   sparse-checkout  Initialize and modify the sparse-checkout

examine the history and state (see also: git help revisions)
   bisect     Use binary search to find the commit that introduced a bug
   diff       Show changes between commits, commit and working tree, etc
   grep       Print lines matching a pattern
   log        Show commit logs
   show       Show various types of objects
   status     Show the working tree status

grow, mark and tweak your common history
   branch     List, create, or delete branches
   commit     Record changes to the repository
   merge      Join two or more development histories together
   rebase     Reapply commits on top of another base tip
   reset      Reset current HEAD to the specified state
   switch     Switch branches
   tag        Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
   fetch      Download objects and refs from another repository
   pull       Fetch from and integrate with another repository or a local branch
   push       Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.
tp8:/tmp>
```

図14.1：基本的なgitコマンド

gitと入力すれば、**git**コマンドの基本的なリストが表示されます（上の写真のようなリスト）。

適用されるグローバル・オプションは、上記のリストで先頭に -- が付いているものだけです。サブコマンドの多くには独自のオプションがあり、それらは上記の[ARGS]に含まれています。

その他のヘルプ

もし、より完全なコマンドのセットを見ることに抵抗があるのであれば、そうしてください：

```
$ git help --all
```

特定のサブコマンドについて読むには、`'git help <command>'`

~~を~~参照してください。Porcelain のメインコマンド

付ける	ファイルの内容をインデックスに追加する
アム	メールボックスから一連のパッチを適用する
アーカイブ	名前付きツリーからファイルのアーカイブを作成する
両分	バイナリサーチでバグブランチを導入したコミットを探す
	ブランチの一覧表示、作成、削除
ひとまとめ	アーカイブによるオブジェクトとRefの移動
チェックアウト	ブランチを切り替えたり、作業ツリーファイルを復元する
さくらんぼ狩り	既存のコミットによってもたらされた変更を適用する
citool	git-commit のグラフィカルな代替品
クリーン	作業ツリーから未追跡のファイルを削除する
クローン	リポジトリを新しいディレクトリにクローンする
コミット	リポジトリに変更を記録する
描く	利用可能な参照差分に基づき、オブジェクトに人間が読みやす
い名前を付ける。	コミット間の変更、コミットと作業ツリーなどの表示
フェッチ	他のリポジトリからオブジェクトや参照元をダウン
ロードする format-patch	電子メール送信用のパッチを準備する
ジーシー	不要なファイルのクリーンアップとローカルリポジトリの最適化
ギットク	Gitリポジトリブラウザ
グレップ	パターンに一致する行を印刷する
グイ	Gitのポータブルなグラフィカル・インターフェース
イニット	空のGitリポジトリを作成したり、既存のリポジトリを再初期化したりします。
	log コミットログを表示する
	コミット間の参照関係を結合する

この長いリストは、かなり威圧的に見えるかもしれませんが、私たちは、これらのいくつかは、本当に専門家の使用のためのものであり、ほとんど使用されていない、または短縮された組み合わせコマンドを使用してより効率的に呼び出すことができます。

さらに、gitにはいくつかのグラフィカルなインターフェースがあり、すべての配管設備に名前をつけることができるようにする必要があります。

14.2 簡単な例

リポジトリの作成と初期化

ローカルプロジェクトを作成します。

```
mkdir git-test  
$ cd git-test  
$ git init
```

すべてのバージョン管理情報を格納する `.git` ディレクトリを作成します。

gitの仕組みと使いやすさを実感してみましょう。今のところ、私たちは自分のローカルプロジェクトを作るだけです。まず、作業ディレクトリを作成し、そのディレクトリで作業するために**git**を初期化します：

```
mkdir git-test
```

```
$ cd git-test
```

```
$ git init
```

プロジェクトを初期化すると、すべてのバージョン管理情報を含む**.git**ディレクトリが作成されますが、プロジェクトに含まれる主要なディレクトリはそのまま残ります。このディレクトリの初期コンテンツは次のようになります：

```
$ ls -l .git
```

```
計40
drwxrwxr-x 7 こう こう 4096 12 30 13:59 ./
              じょ じょ
              うじ うじ
              っし っし
              ゆう ゆう
drwxrwxr-x 3 こう こう 4096 12 30 13:59 ../
              じょ じょ
              うじ うじ
              っし っし
              ゆう ゆう
drwxrwxr-x 2 こう こう 4096 12 30 13:59 ブランチ/
              じょ じょ
              うじ うじ
              っし っし
              ゆう ゆう
-rw-rw-r-- 1 こう こう 92 12 30 13:59 コンフィグ
              じょ じょ
              うじ うじ
              っし っし
              ゆう ゆう
-rw-rw-r-- 1 こう こう 58 12 30 13:59 記述
              じょ じょ
              うじ うじ
              っし っし
              ゆう ゆう
-rw-rw-r-- 1 こう こう 23 12 30 13:59 HEAD
              じょ じょ
              うじ うじ
              っし っし
              ゆう ゆう
drwxrwxr-x 2 こう こう 4096 12 30 13:59 フック類
              じょ じょ
              うじ うじ
              っし っし
              ゆう ゆう
drwxrwxr-x 2 こう こう 4096 12 30 13:59 インフォ
              じょ じょ                      メーショ
              うじ うじ                      ン/
              っし っし
              ゆう ゆう
drwxrwxr-x 4 こう こう 4096 12 30 13:59 オブジェク
              じょ じょ                      ト/
              うじ うじ
              っし っし
              ゆう ゆう
drwxrwxr-x 4 こう こう 4096 12 30 13:59 リファエ
              じょ じょ                      ス
              うじ うじ
              っし っし
              ゆう ゆう
```

このディレクトリとそのサブディレクトリの内容については後ほど説明しますが、ほとんどの場合、最初は空っぽです。

プロジェクトにファイルを追加する

```
$ echo some junk > somejunkfile
```

```
git add somejunkfile
```

```
$ git status
```

ブランチ上のマ

スター 初期コミ

ット

コミットする変更：

(ステージを解除するには "git rm --cached <file>..." を使用します)

新しいファイル：somejunkfile

ファイルがステージング（追加）されているが、まだコミットされていないことを伝えていることに注意してください。

誰のリポジトリなのか

このリポジトリの責任者をgitに伝えます：

```
$ git config user.name "もう一人の天才"  
$ git config user.email "a_genius@linux.com"
```

グローバル設定ファイルにあらかじめ定義されている場合を除き、新しいプロジェクトごとに行う必要があります。

- このリポジトリの責任者をgitに教えてあげましょう：

```
$ git config user.name "もう一人の天才"  
$ git config user.email "a_genius@linux.com"
```

これは、グローバル設定ファイルで事前に定義していない限り、新しいプロジェクトごとに行う必要があります。

修正内容

ファイルを修正する：

```
$ echo 別の行 >> somejunkfile
$ git diff
```

```
diff --git a/somejunkfile b/somejunkfile
index 9638122...6023331 100644
--- a/somejunkfile
+++ b/somejunkfile
@@ -1 +1,2 @@
 屑物
+αのライン
```

変化をコミットする：

```
$ git add somejunkfile
$ git commit -m "私の初期コミット"
```

```
初期コミット作成 eafad66: 私の初期コミットを作成しました。
1 ファイル変更、1 挿入(+), 0 削除(-) 作成モード
100644 somejunkfile
```

- では、ファイルを修正し、差分の履歴を見てみましょう：

```
$ echo 別の行 >> somejunkfile
$ git diff
```

```
diff --git a/somejunkfile b/somejunkfile
index 9638122...6023331 100644
--- a/somejunkfile
+++ b/somejunkfile
@@ -1 +1,2 @@
 屑物
+αのライン
```

- 変更を実際にリポジトリにコミットするために、私たちはこうします：

```
git add somejunkfile
$ git commit -m "私の初期コミット"
```

```
初期コミット作成 eafad66: 私の初期コミットを作成しました。
1 ファイル変更、1 挿入(+), 0 削除(-) 作成モード
100644 somejunkfile
```

mオプションでコミットに付随する識別メッセージを指定しない場合、エディタに飛び込んで内容を入力することになります。これを実行しないと、コミットは拒否されます。選択されるエディタは、EDITOR環境変数で設定されたものです。これはGIT_EDITORを設定することで優先させることができます。

沿革

git log

```
commit eafad66304ebbcd6acfe69843d246de3d8f6b9cc
Author : ある天才 <a_genius@linux.com>
日付:   Wed Dec 30 11:07:19 2009 -0600
```

私の最初のコミット

- で履歴を見ることができます：

git log

```
commit eafad66304ebbcd6acfe69843d246de3d8f6b9cc
Author : ある天才 <a_genius@linux.com>
日付:   Wed Dec 30 11:07:19 2009 -0600
```

私の最初のコミット

で、そこに情報が入っているのがわかると思います。長い16進数の文字列が**コミット番号**であることにお気づきでしょう。これは160ビット、40桁のユニークな識別子で、後ほど説明します。**git**はファイル名ではなく、このような獣を気にします。

- これで、すでに存在するファイルを変更したり、git add で新しいファイルを追加したりすることが自由にできるようになりました。しかし、実際にリポジトリに変更が反映されるのは、もう一度 git commit を行うまでです。
- 今のは、それほど悪くはなかった。しかし、まだ表面しか見ていないのです。

14.3 コミットメントへのサインオフ

コミットメントへのサインオフ

履歴を追跡し、適切なライセンスと所有権を確保するために、リポジトリへのすべての変更に関与した人を知る。

sオプションでコミットにSigned-off-by行を追加します：

\$ **git commit -s -m "私の初期コミット"**

```
commit eafad66304ebbcd6acfe69843d246de3d8f6b9cc
Author : ある天才 <a_genius@linux.com>
日付:    Wed Dec 30 11:07:19 2009 -0600
```

私の最初のコミット

サインオフバイ : もうひとりの天才 <a_genius@linux.com>

署名者は著者である必要はなく、メンテナやレビュアーであってもよい。

履歴を追跡し、含まれるコードが適切なライセンスと所有権で行われていることを誰が保証しているかを知るために、リポジトリへのすべての変更に関与した人を知ることは重要です。

これは、-sオプションでコミットにSigned-off-by行を追加することで最も簡単に行うことができます。つまり、次のようにすることができます：

\$ **git commit -s -m "私の初期コミット"**

を、得ることになる：

```
commit eafad66304ebbcd6acfe69843d246de3d8f6b9cc
Author : ある天才 <a_genius@linux.com>
日付:    Wed Dec 30 11:07:19 2009 -0600
```

私の最初のコミット

サインオフバイ : もうひとりの天才 <a_genius@linux.com>

サインをする人は、変更の作者である必要はないことに注意してください。さらに、ある変更に対して、複数の人が署名することもできます。

14.4 しゅとけん

しゅとけん

マスターという名前は非推奨です。ほとんどのプロジェクトでは、メインに置き換えています。Inclusive Naming Initiative が作成した <https://inclusivenaming.org/word-lists/> を参照してください。

また、ブラックリスト、ホワイトリスト、ワーカーなどの用語の使用も非推奨です。

スレーブ

新しいリポジトリの場合：

```
$ git init
```

```
git checkout -b main
```

今後は、この役割を担うブランチの名前がmainであり、必要な変更がなされているものとします。

- 歴史的に、新しいリポジトリは常にmasterと呼ばれる初期ブランチで作成されました。しかし、多くのプロジェクトがこの名前を非推奨とし、mainに置き換えています。
- この変更の理由と、ブラックリストという言葉の使用の非推奨など、用語の変更とwhitelistについては、Inclusive Naming Initiativeが作成した<https://inclusivenaming.org/word-lists/>。
- 技術的には、このメインブランチにはどんな名前でもつけることができます。しかし、git は新しいリポジトリにデフォルトの名前をつけなければなりませんし、ウェブホストによっては避けるのが難しい要件もあります。
- GitHubはこの命名規則の変更を強く推奨しており、<https://github.com/github/renaming>で、新しいリポジトリの作成方法と、古いリポジトリの名前を変更してmainをメインブランチとする方法を説明しています。
- 新しいリポジトリを作成する場合、最初からこれを行う最も簡単な方法は、次のようにすることです：

```
$ git init
```

```
git checkout -b main
```


既存のレポの名前を変更する

ローカルブランチとリモートブランチの両方の名前を変更します：

```
git checkout master
git branch -m master main      # ローカル名を変更する

# リモート名を変更する
git push -u origin main
git symbolic-ref refs/remotes/origin/HEAD refs/remotes/origin/main

# 名前を確認します！
$ git branch -a
```

より簡単にアプローチできる：

```
git checkout master
git branch main
git checkout main
git push -u origin main
```

と言って、今後はmasterを無視すればいいのです。

- 既存のリポジトリでは、サーバー上のローカルブランチとリモートブランチの両方の名前を、次のようにして変更できます。

```
git checkout master
# ローカル名を変更する
git branch -m master main

# リモート名を変更する
git push -u origin main
git symbolic-ref refs/remotes/origin/HEAD refs/remotes/origin/main

# 名前を確認します！
$ git branch -a
```

- リモートサーバー（GitHubなど）の設定によっては、これがうまくいく場合もあればいけない場合もあります。もっと簡単な方法は、masterブランチを削除せず、mainにコピーして、今後はそこから作業する、というような方法です：

```
git checkout master
git branch main
git checkout main
git push -u origin main
```

と言って、今後はmasterを無視すればいいのです。

14.5 研究室

練習問題14.1：リポジトリの設定と、変更・コミットの実行

すでにお伝えした簡単なコマンドのサブセットだけを使って、簡単なリポジトリをセットアップしてみます。

1. まず、git initでリポジトリを初期化します。次にgit configで著者とメールアドレスの情報を追加します。
2. 簡単なテキストファイルをいくつか作成し、git addとgit commitでリポジトリに追加してコミットします。
3. 次に、ファイルの1つを変更し、git diffを実行して、作業中のプロジェクトファイルとリポジトリにあるファイルとの違いを確認します。
4. 変更したファイルを再度リポジトリに追加し、再度git diffを実行します。
5. 最後にもう一度コミットし、git logを使って履歴を調べます。

これらすべてを行うさまざまな段階で、.gitディレクトリの内容を調べ、どのファイルが変更されているか、その内容はどのようなかなどを確認します。できる限り多くのことを学んでください。

解答 14.1

1.

これから作業するディレクトリを設定する

```
rm -rf git-test ; mkdir git-test ; cd git-test
```

リポジトリを初期化し、.config ファイルに名前とemailを入れる

```
echo -e "LLLL          リポジトリ作成とITの設定" git init
git config user.name "お利口さん"
git config user.email "asmartguy@linux.com"
```

2.

echo -e "LLLL CREATING A COUPLE OF FILES AND ADDING THEM TO THE PROJECT AND COMMITTINGn"

*# いくつかのファイルを作成し、プロジェクトに追加します。
2つのコミットとして実行しますが、1つのコミットとして実行することも可能です。*

```
echo file1 > file1
git add file1
git commit file1 -s -m "これは最初のコミットです"
```

```
echo file2 > file2
git add file2
git commit -s -m "これは2番目のコミットです"
```

3.

ファイルの1つを変更し、その違いをリポジトリで確認する。

```
echo -e "MMMM          ファイルの1つを変更し、その後ディフティングする"
```

```
echo これは file2 の別の行です >> file2 git
diff
```

```
4.
# now stage it and diff again
git add file2
git diff
```

```
5.
echo -e "MMMMM                変更されたファイルを追加して、再びディフティングする。"
echo -e "MMMMM                RECOMMITTING FOR THE THIRD TIME" です。
# now get it all in with another commit

git commit -s -m "これは3番目のコミットです"

echo -e "\n\n*****                PROJECTn の歴史に目を向ける "

# 歴史を見る

コミットログ
```

上記の手順が書かれたスクリプトは、以下からダウンロードできます。

[s-14/lab_gitexample.sh](#)

をソリューションファイルの中に入れてください。

[lab_gitexample.sh](#)

第15章

プロセス



15.1	プログラムおよびプロセス	216
15.2	プロセスリミット	220
15.3	プロセスの作成	223
15.4	プロセスコントロール	225
15.5	未来へのスタートラインに立つプロセス	228
15.6	プロセスステート	232
15.7	実行モード	233
15.8	デーモン	236
15.9	にこにこ	237
15.10	研究室	239

学習目標

このセッションが終わるころには、次のことができるようになっているはずです：

- プロセスおよびそれに関連するリソースを説明する。
- プロセス、プログラム、スレッドの区別をつける。
- プロセスの属性、権限、状態を理解し、制限を制御する方法を知っている。
- プロセスがどのように作られ、どのようにバックグラウンドに置かれ、どのようにフォアグラウンドに戻されるかを話してください。
- ジョブ（バックグラウンドプロセス）の管理、将来的にプロセスを開始するための`at`の使用、実行を一時停止するためのスリープ、そして`cron`で定期的な実行をスケジューリングします。
- 考えられるプロセス状態を説明する。
- ユーザーモードとカーネルモードで実行することの違いを説明する。
- デーモンプロセスについて説明する。

215

15.1 プログラムおよびプロセス

プログラムとは？

- プログラムは、実行される命令（およびそのために必要なデータ）のセットである。
- CPUが直接実行するマシンレベルの命令で構成されている場合がある。
- 他のプログラムによって解釈されるコマンドのリストで構成されている場合がある。
- プログラマは、様々な言語（C、C++、Perlなど）を使ってプログラムの命令をコーディングする

プログラムvsスクリプト

プログラムとは、コンパイルしてバイナリで実行するもの、スクリプトとは、**bash**、**Python**、**Perl**などのインタプリタで実行するものを区別する必要があります。

プログラムは、命令のセットであり、命令を実行する際に使用される内部データも含まれます。プログラムは、外部データを使用することもあります。内部データには、ユーザープロンプトを表示するために使用されるプログラム内部のテキスト文字列が含まれます。外部データには、データベースからのデータも含まれます。

ls、**cat**、**rm**などのユーザーコマンドの多くは、オペレーティングシステムのカーネルやシェルの外部にあるプログラムです（言い換えれば、ディスク上に独自の実行プログラムを持っています）。

プロセスとは？

- プロセスとは、実行されるプログラムのこと
- **Linux**は、プログラムが実行されるたびに、新しいプロセスを作成します。
- 複数のプロセスが同時に同じプログラムを実行することがある
- オペレーティングシステムの主な目的は、ユーザーに代わってプロセスの実行を管理することである



プログラムは複数の同時進行するスレッドで構成されることがあり、各スレッドはそれ自身のプロセスとしてみなされる。

プロセスとは、実行中のプログラムのインスタンスのことです。実行中やスリープ状態など、さまざまな状態になることがあります。

すべてのプロセスは、**pid**（プロセスID）、**ppid**（親プロセスID）、**pgid**（プロセスグループID）を持っています。また、各プロセスは、プログラムコード、データ、変数、ファイルディスクリプタ、環境を持っています。

プロセスは**スケジューリング**によって制御され、完全にプリエンプティブである。カーネルだけがプロセスを先取りする権利を持っており、互いに先取りすることはできません。

歴史的な理由から、最大の**pid**は16ビット数、つまり32768に制限されてきました。この値は、[/proc/sys/kernel/pid_max](#)を変更することで変更可能です。プロセスが作成されると、最終的にpid_maxに到達し、その時点でPID = 300で再スタートします。

プロセスの属性

- すべてのプロセスには一定の属性があります：
 - 実行されるプログラム
 - コンテキスト(状態)
 - パーミッション
 - 関連するリソース

すべてのプロセスは、何らかのプログラムを実行している。任意の瞬間に、プロセスは、CPUレジスタの状態、プログラムのどこで実行しているか、プロセスのメモリに何があるか、その他の情報をトラップすることによって、自分自身のスナップショットを取ることができます。これがプロセスのコンテキストである。

プロセスは、CPU時間を他と共有する際にスケジューリング・アウトすることができる（あるいは、ユーザが要求を出す、データが届くなど、何らかの条件が満たされるのを待つ間にスリープさせる必要がある）ため、プロセスを交換する際にコンテキスト全体を保存でき、実行再開時に復元できることは、カーネルがコンテキストスイッチを行う上で重要である。

すべてのプロセスは、どのユーザーが実行するよう呼び出したかに基づくパーミッションを持っています。また、そのプログラムファイルを誰が所有しているかに基づいたパーミッションも持っている場合があります。実行ビットに "s" が付いているプログラムは、"実際の"ユーザーIDとは異なる "実効"ユーザーIDを持っています。これらのプログラムは、**setuid**プログラムと呼ばれます。非セツイドプログラムがそれを起動したユーザーの権限で実行されるのに対して、プログラムを所有するユーザーのユーザーIDで実行されます。rootが所有するsetuidプログラムは、セキュリティ上問題となることがあります。

passwdプログラムは、setuidプログラムの一例です。どのユーザーでも実行可能です。ユーザーがこのプログラムを実行すると、ユーザーのパスワードが管理されている書き込み制限のあるファイル、`/etc/passwd`と`/etc/shadow`を更新できるように、プロセスはroot権限で実行されます。

なお、各プロセスは、割り当てられたメモリやファイルハンドルなどの資源を持っています。

プロセスリソースの分離

- カーネルがプロセスを分離する（デフォルトで）
 - セキュリティの確保を促進する
 - 安定性を高める
- カーネルは、プロセスが任意のハードウェアのリソースにアクセスすることを許可していません。
 - カーネルが管理するハードウェアへのアクセス
 - システムコールによるアクセス

プロセスが開始されると、他のプロセスから保護するために、独自のユーザースペースに隔離されます。これにより、セキュリティが向上し、より安定した動作を実現します。

プロセスはハードウェアに直接アクセスすることはできません。ハードウェアはカーネルによって管理されているため、プロセスはシステムコールを使用する必要があります。

を使用して、ハードウェアに間接的にアクセスすることができます。システムコールは、アプリケーションとカーネルとの間の基本的なインターフェースです。

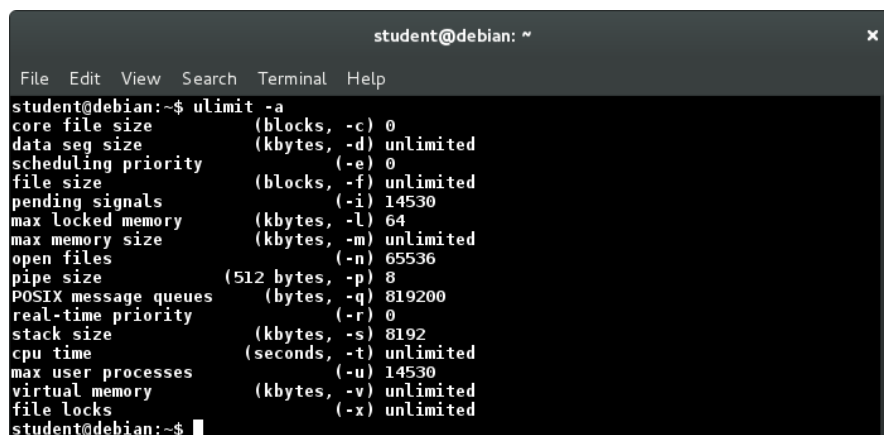
15.2 プロセスリミット

ulimitでプロセスを制御する

- **bash**コマンドを内蔵
- プロセスリソースの制限を表示またはリセットする
- 個々のユーザーやプロセスがシステムリソースを使い果たすことがないように、機能を制限するために使用します：
 - メモリ
 - CPU タイム
 - アクティブなプロセスまたはファイルの数
- システム管理者は、全ユーザーに対してシステム全体の制限を設定することができる

システム管理者は、これらの値の一部をどちらかに変更する必要がある場合があります：

- 個々のユーザーやプロセスが、メモリ、CPU時間、システム上の最大プロセス数などのシステムリソースを使い果たすことがないように、機能を制限すること。
- 例えば、多くのクライアントを扱うサーバーでは、デフォルトの1024個のファイルを開くと、その作業が不可能になることがあります。



```
student@debian: ~  
File Edit View Search Terminal Help  
student@debian:~$ ulimit -a  
core file size          (blocks, -c) 0  
data seg size           (kbytes, -d) unlimited  
scheduling priority     (-e) 0  
file size               (blocks, -f) unlimited  
pending signals         (-i) 14530  
max locked memory       (kbytes, -l) 64  
max memory size         (kbytes, -m) unlimited  
open files              (-n) 65536  
pipe size               (512 bytes, -p) 8  
POSIX message queues    (bytes, -q) 819200  
real-time priority      (-r) 0  
stack size              (kbytes, -s) 8192  
cpu time                (seconds, -t) unlimited  
max user processes      (-u) 14530  
virtual memory          (kbytes, -v) unlimited  
file locks              (-x) unlimited  
student@debian:~$
```

図15.1: ulimit

リミットの設定

- することで、特に制限を設けない :
\$ ulimit [オプション] [リミット]。
のように
\$ ulimit -n 1600
となり、ファイルディスクリプタの最大数が1600に増加します。
- [etc/security/limits.conf](#)を変更し、再ブートすることで永続化させる。

この変更は、現在のシェルにのみ影響することに注意してください。ログインしているすべてのユーザーに有効な変更を加えるには、[/etc/security/limits.conf](#)という非常にきれいにセルフドキュメント化されたファイルを修正し、再起動する必要があります。

ハードリミットとソフトリミット

限度額には2種類あります：

- **Hard**：ユーザーがリソース制限を引き上げることができる最大値で、rootユーザーのみが設定できる。
- **ソフト**：現在の制限値で、ユーザーが変更できるが、ハードの制限値を超えることはできない。

```
$ ulimit -H -n
```

```
4096
```

```
$ ulimit -S -n
```

```
1024
```

15.3 プロセスの作成

プロセスの作成

- 最初のユーザプロセスはinitと呼ばれ、pid = 1である。
- 後続のプロセスは、initや他の実行中のプロセスからフォークされる
- 親プロセスからフォークすると、子プロセスが生成され、あらゆる点で親と同等であり、ピアとなる。
- 親が死亡した場合、子供はinitの「養子」となる。
- カーネルが作成したプロセスもあります。これらのプロセスは、psの出力で[...]で囲まれた名前を持っています。



ご注意ください

systemdベースのシステムでは、kthreaddという特別なプロセスがpid=2で実行され、その仕事は孤児を養子にすることであり、その子供はppid=2を示すようになる。

平均的なLinuxシステムでは、常に新しいプロセスが作成されています。これはしばしばフォークと呼ばれ、元の親プロセスは実行され続け、新しい子プロセスが開始されます。

多くの場合、単なるフォークではなく、親プロセスが終了し、子プロセスが親のプロセスIDを継承するexecでフォローされます。フォークとエクゼキューションという言葉は頻繁に使われるため、一つの単語として考えられることもあります。

古いUNIXシステムでは、forkやexecと多くの点で似ているが、細部が異なるspawnというプログラムがよく使われていた。これはPOSIX標準の一部ではなく、Linuxの通常の部分でもありません。

新しいプロセスがどのように開始されるかを知るために、多くのクライアントを処理するウェブサーバーを考えてみましょう。クライアントとの新しい接続が行われるたびに、新しいプロセスを起動することができます。Linuxでは、完全なプロセスを作るか、新しいスレッドを作るかの技術的な差はあまりなく、それぞれのメカニズムがほぼ同じ時間、同じ量のリソースを使用します。

別の例として、sshd デーモンは、init プロセスが sshd init スクリプトを実行したときに起動し、その後 sshd デーモンを起動する役割を担います。このデーモンプロセスは、リモートユーザーからの ssh リクエストをリッスンします。

リクエストを受信すると、sshd はそのリクエストに対応するために自身の新しいコピーを作成します。各リモートユーザーは、リモートログインに対応するため、sshd デーモンの独自のコピーを実行させます。sshd プロセスは、リモートユーザーを認証するためにログインプログラムを開始します。認証に成功すると、ログインプロセスはシェル (bashなど) をフォークオフして、ユーザーコマンドを解釈します。

内部カーネルプロセスは、バッファがディスクにフラッシュされるようにする、異なるCPUの負荷が均等になるようにする、デバイスドライバがキューに入れられた仕事を処理する、などのメンテナンス作業を行います。これらのプロセスは、システムが稼働している間は、何かすることがあるとき以外は眠っていることが多い。

コマンドシェルでプロセスを作成する

などのコマンドシェルインタプリターでユーザーがコマンドを実行する場合、そのコマンドを実行します。

バッシュを使用しています：

- 新しいプロセスが作成される（ユーザーのログインシェルからフォークされる）。
- waitシステムコールで、親シェルプロセスをスリープさせる
- コマンドは、**exec**を介して子プロセスの空間にロードされます。
システムコール、**bash**に代わるもの
- コマンドの実行が完了し、子プロセスはexitシステムコールで終了します。
- 親シェルが子プロセスの死によって再起不能になり、新しいシェルプロンプトを発行する。
- その後、親シェルはユーザーからの次のコマンド要求を待ち、このサ

コマンドラインの末尾にアンパサンド（&）を付けてバックグラウンド処理用のコマンドを発行した場合、親シェルは待機要求をスキップして、すぐに新しいシェルプロンプトを発行してバックグラウンド処理を並行して実行できるようにすることができる。それ以外のフォアグラウンドの要求に対しては、シェルは子プロセスが完了するか、シグナルによって停止されるまで待ちます。

一部のシェルコマンド（echoやkillなど）は、シェル自体に組み込まれており、プログラムファイルの読み込みを伴わないものです。これらのコマンドでは、実行の際にforkもexecも発行されません。

15.4 プロセスコントロール

背景と前景の処理

- デフォルトでフォアグラウンドで実行されるプロセス
- コマンドの後にアンパサンド（&）を付けると、そのコマンドをバックグラウンドで実行します：

`$ sudo updatedb &`

- CTRL-Zでフォアグラウンドプロセスをサスペンドする
 - **bg**はバックグラウンドで動作させる
 - **fg**は、それを前面に押し出す
- CTRL-Cでフォアグラウンドプロセスを終了させる

フォアグラウンドジョブはシェルから実行され、ジョブが終了するまでシェルへのアクセスを遅らせることができます。これは問題ではないかもしれませんが、ジョブが完了するまでに長い時間がかかる場合、バックグラウンドに置くことでシェルを解放し、さらなる対話的な作業を行うことができます。バックグラウンドジョブは低い優先度で実行されるため、対話的な作業をスムーズに進めることができます。また、バックグラウンドジョブに影響を与えることなく、ターミナルウィンドウをログオフすることもできます。

```
$ updatedb &  
[1] 22209  
$ sleep 100  
^Z
```

```
[2]+停止中      スリープ
```

```
$ bg
```

```
[2]+ sleep 100 &
```

```
$ fg
```

```
スリープ  
100^Z  
[2]+停止中      スリープ
```

```
$ bg
```

```
[2]+ sleep 100 &
```

```
$ fg
```

```
スリープ  
100^C
```

求人情報の管理

- **jobs**は、現在の端末のバックグラウンドプロセスを表示します
- ジョブID、状態、コマンド名を表示します：

求人情報

1 -	Runningupdatedb &
2 +	ストップスリープ 10

- ジョブIDは**bg**と**fg**で使用可能です

\$ jobs -l

は、ジョブのPIDを提供します。

バックグラウンドジョブは、ログオフするとそのウィンドウから開始された**ジョブ**が表示されなくなるという点で、ターミナルウィンドウとまだ多少つながっています。

15.5 未来へのスタートラインに立つプロセス

アットを使って未来から始める

- 任意の非対話型コマンドを指定時刻に実行する
- 遅延後にコマンドを開始する：

現在 + 2日

```
at> mail < /var/log/messages admin@example.com at>  
<EOT> です。  
job 1 at 2013-01-16 13:24
```

- CTRL-Dを使用してEOT文字を挿入します。
- キューイングされたジョブの検査や削除のために**atq**を実行します。

未来の時間を指定して**at**を入力することで、**at**のインタラクティブな部分が開始されます。プロンプトで、実行するコマンドを入力し、エンターキーを押し、CTRL-Dで終了します。その後、**atq**を使用してジョブ情報を確認します。

クーロン

- cronは、特定の間隔でコマンドをスケジュールするために使用します。
- さまざまな方法で管理できる：
 - **crontab**は、ユーザーがジョブを指定することができます
 - **/etc/cron.d/** はフォーマットされたジョブファイルで拡張することができる
 - **/etc/cron.{hourly,daily,weekly,monthly}** には、任意のシステムスクリプトを含めることができます。
- cronは何十年も前からUNIX系OSで使われてきましたが、最近のLinuxディストリビューションでは、まもなく説明するように、**anacron**に移行しています。

cronは非常に便利で柔軟なツールです。バックアップなど、定期的に行う必要があるジョブに使用されます。

タスクは、1時間ごと、毎日、週1回、月1回、あるいは10秒ごとに実行するようにキューイングすることができます。ジョブが完了したときや失敗したときには、メールが送信されます。

\$ ls -l /etc/cron.d

トータル16

```
-rw-r--r-- 1 root root 128 Mar 29 2017 0hourly
-rw-r--r-- 1 root root 78 Dec 29 09:24 atop
-rw-r--r-- 1 root root 108 Jun 13 2017 raid-check
-rw----- 1 root root 235 Mar 29 2017 sysstat
```

\$ ls -l /etc/cron.daily

計68

```
-rwxr-xr-x 1 root root      302 Mar 7 08:55 docker-latest-logrotate
-rwxr-xr-x 1 root root 18812 Mar 20 00:01 google-chrome
-rwxr-xr-x 1 root ルート      1 2017年1月17日 グーグルアース
lrwxrwx 1 root ルート      45 2016/06/16 google-talkplugin→です。
→ /opt/google/talkplugin/cron/google-talkplugin
-rwx----- 1 root root      219 2017年1月24日 ログrotate
-rwxr-xr-x 1 root ルート      618 2014年03月17日 man-db.cron
-rwx----- 1 root root      208 2016年2月4日 mlocate
-rwxr-x--- 1 root root 2126 Aug 17 2016 prelink
-rwx----- 1 root root      256 2017年9月1日 rhsmd
-rwxrwxr-x 1 root root 18775 Mar 5 13:55 slack
```

アナクロン

- 古いLinuxシステムでは、**crontabは/etc/cron.***サブディレクトリにジョブを実行するタイミングに関する情報を含んでいました。
- しかし、暗黙の了解として、機械は常に動いているものとされてきました。
- マシンの電源が切れていると、スケジュールされたジョブは実行されない。
- anacronは最近のシステムで**cron**に取って代わられました。
- anacronは、システムが稼働しているときに、必要なジョブを制御された状態で、時間をずらして実行します。
- キーとなる設定ファイルは **/etc/anacrontab** です。

anacronの設定ファイルの例：

/etc/anacrontab

/etc/anacrontab: anacronの設定ファイル

詳細は anacron(8) と anacrontab(5) を参照して

SHELL=/bin/sh

PATH=/sbin:/bin:/usr/sbin:/usr/bin

MAILTO=root

ジョブの基本遅延に加えられる最大システム遅延時間

random_delay=45

以下の時間帯にのみジョブが開始される

START_HOURS_RANGE=3-22

#期間（日） 遅延時間（分） job-identifier めいれいけい

1 5 クロンデー ナイスランパーツ
 リー /etc/cron.daily

7 25 クロンウィ ナイスランパーツ
 ークリー /etc/cron.weekly

スリーピング

- sleepは、簡単にコマンドを遅延させるために使用することができます：
\$ (sleep 600; launchbackup.sh) &

sleepはシェルスクリプトでも使用でき、例えば、会議の時間が近づいていることを10分ごとにリマインダとして表示することがで

きます。

15.6 プロセスステート

プロセスステート

- プロセスは、いくつかの可能な状態のうちの1つである。
 - 実行中：CPUで実行中、またはO/Sが判断して実行可能な状態。
 - 待機すること：ディスクや外部からの情報を待つことで、スリーピングとも呼ばれます。
 - 停止しています：デバッガまたはCtrl-Zで停止することができます。
 - ゾンビ：消えたが、親プロセスで適切に削除されていない

Running状態のプロセスは、現在CPUから注目されているか（言い換えれば、実行中）、実行の準備ができてキューで待機しているかのいずれかです。キューに入れられたプロセスは、優先順位がキューの1番目に上がり、オペレーティングシステムにアイドルなCPUがあるときに、実行するように選択される。

Waiting（または**Sleeping**）状態のプロセスは、自身が行った要求（通常はI/O）を待っており、要求が完了するまでそれ以上進むことができない。要求が完了すると、イベントがO/Sに割り込み、CPUが待機中のプロセスに再割り当てされ、処理を続けることができるようになります。

プロセスは、命令の実行が中断されたことを意味する「**停止**」状態になることがあります。この状態は、プログラマーが実行中のプログラムのメモリ、CPUレジスタ、フラグ、その他の属性を調べるときによく経験します。これが終わると、プロセスは再開することができます。

ゾンビの状態は、プロセスが終了したときに入るものです。ゾンビプロセスは重要で、各ゾンビはO/Sのプロセステーブルのスペースを占有し続けるからです。プロセステーブルには、システム内の各アクティブプロセスのエントリがあります。ゾンビプロセスは、プロセステーブルのエントリを除くすべてのリソースを解放します。

スケジューラはすべてのプロセスを管理する。プロセスの状態は、プロセスリストによって報告される。

15.7 実行モード

2つの実行モード

- ユーザーモード
 - 制限された権限で実行する
- システム(またはカーネル)モード
 - 完全な無制限権限で実行する

プロセス（またはマルチスレッドプロセスの特定のスレッド）は、任意の時点で、**ユーザーモード**または**マルチスレッドモード**で実行することができます。

システムモード、カーネル開発者は通常**カーネルモード**と呼んでいます。

どのような命令を実行できるかはモードによって異なり、ソフトウェアではなくハードウェアのレベルで強制されます。

モードはシステムの状態ではなく、マルチコアやマルチCPUシステムでは各ユニットが個別の状態になることがあるため、プロセッサの状態であると言えます。

インテルの用語では、ユーザーモードは**リング3**、システムモードは**リング0**とも呼ばれる。

ユーザーモード

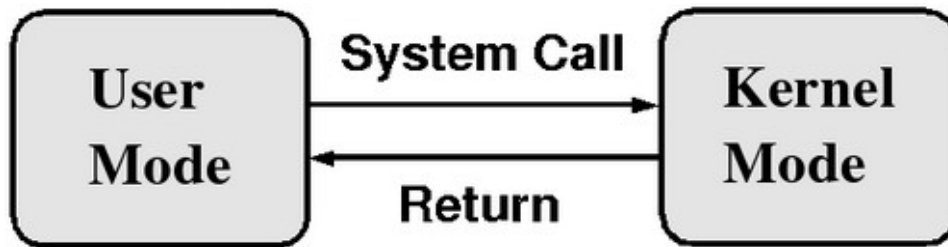


図15.2 : ユーザーモードとカーネルモードとシステムコール

- ユーザーモードはカーネルモードより権限が少ない
- 通常のプログラムコードは、このモードで実行されます

システムコール（次節で説明）を実行する場合を除き、プロセスはユーザーモードで実行され、カーネルモードよりも少ない特権を持つ。

プロセスが開始されると、他のプロセスから保護するために、独自のユーザースペースに隔離されます。これにより、セキュリティが促進され、より大きな安定性が生まれます。これは、**プロセスリソースの分離**と呼ばれることもあります。

ユーザーモードで実行される各プロセスは、独自のメモリ空間を持ち、その一部は他のプロセスと共有されることがあります。共有メモリセグメントを除き、ユーザープロセスは他のプロセスのメモリ空間に読み書きすることはできません。

rootユーザーやsetuidプログラムとして実行されるプロセスも、システムコールにジャンプするとき以外はユーザーモードで実行され、ハードウェアにアクセスする能力は限られています。

システム(カーネル)モード

- プログラムがシステムコールを行う際に入力される
- システムコールが終了するまで持続する
- ユーザーモードに比べ、より多くの権限を持つ
- システムモードのプロセスは、カーネルコードを実行しています。
 - ディスクアクセス（読み出し、書き込み）の取得
 - メモリ増設のお願い(brk, sbrk)
 - 現在の作業ディレクトリを変更する（cd）

システム（カーネル）モードでは、CPUは周辺機器、メモリ、ディスクなど、システム上のすべてのハードウェアにフルアクセスすることができます。アプリケーションがこれらのリソースにアクセスする必要がある場合は、システムコールを発行し、ユーザーモードからカーネルモードへのコンテキストスイッチを引き起こす必要があります。ファイルの読み書き、新しいプロセスの作成など、この手順に従わなければなりません。

アプリケーション・コードがカーネル・モードで実行されることはなく、カーネル・コードであるシステム・コールそのものだけが実行されます。システムコールが完了すると、戻り値が生成され、プロセスは逆コンテキストスイッチでユーザーモードに戻る。

その他、ハードウェア割り込みの処理や、スケジューリングルーチンなどシステムの管理タスクを実行するときなど、プロセスとは関係のないカーネルモードになっていることがあります。

15.8 デーモン

デーモン

- デーモンは、システムのユーザーに特定のサービスを提供することを唯一の目的とするバックグラウンドプロセスです。
- 必要なときだけ動作するため、かなり効率的な運用が可能です。
- 多くは起動時に起動します。
- 名前はしばしば（常にではありませんが）**d**で終わります：例えば、**httpd** や **systemd-udev** です。
- 外部イベント（**systemd-udev**）または経過時間（**crond**）に応答する場合があります。
- 一般に、制御用端子や標準入出力装置を備えていない。
- より良いセキュリティ制御を提供することもあります。

デーモン・プロセスは通常、システムのユーザーに特定のサービスを提供するバックグラウンド・プロセスである。システム上の唯一の目的は、その特定のサービスを提供することです。例としては、**xinetd**、**httpd**、**lpd**、**vsftpd**などがあります。

15.9 にここ

優先順位を決めるためにナイスを使

- 各プロセスには、プロセスの実行優先度に影響するナイス値がある
- **nice** は、その nice 値を調整することで、プロセスの優先度を上げたり下げたりすることができます。
- ナイス値が高いほど優先度が低い
- 低い素敵な値は優先度を上げる
- 非特権ユーザーはナイスバリューを上げることしかできない
- スーパーユーザーだけが素敵な値を下げるすることができます

nice 値は -20 (最低の nice、最高の優先度) から 19 (最高の nice、最低の優先度) までの値です。nice 値を下げるができるのはスーパーユーザーですが、どのユーザーでもプロセスの nice 値を上げることができることに注意してください。デフォルトでは、プロセスの nice 値は 0 であり、これはシェルから継承される。

nice は、特定の nice 値でプロセスを実行するために使用することができます：

```
$ nice -n 10 myprog
```

myprog を nice 値 10 で実行します。

```
$ nice -n 19 myprog
```

myprog を最も低い優先順位で実行し、19 の素敵な値を表示します。

```
$ sudo nice -n -20 myprog
```

myprog を最優先で実行、素敵な値 -20

nice の値を与えない場合、デフォルトは nice を 10 増加させる。引数を全く与えない場合は、現在の優しさを報告します。他に競合するものがなければ、そのプロセスがすべての CPU 時間を獲得することもあり得ます。

ナイスバリューを変更する

- **renice**は、実行中のプロセスのnice値を変更することができます。

`$ renice --help`

使用することです：

```
renice [-n] <priority> [-p|--pid] <pid>...  
renice [-n] <priority> -g|--pgrp <pgid>...  
renice [-n] <priority> -u|--user <user>...
```

- pid 20003 の nice value を 5 に上げる：
`$ renice +5 -p 20003`
- スーパーユーザーだけがプロセスの素敵な値を下げるができる

renice は、すでに実行中のプロセスの nice 値を上げたり下げたりするのに使われます。基本的には、その場で素敵な値を変更することができます。

デフォルトでは、スーパーユーザーだけがnicenessを下げる、つまり優先度を上げることができます。しかし、</etc/security/limits.conf>を編集することで、通常ユーザーにも所定の範囲内でnicenessを下げる能力を与えることが可能です。

非特権ユーザーが素敵な値を増やした後、スーパーユーザーだけがそれを元に戻すことができます。

既に実行中のプロセスの綺麗さを変えるには、次のように**renice**コマンドを使うのが簡単です：

`ドル レニス +3 13848`

とすると、**pid = 13848** のプロセスで niceness が 3 にリセットされます。複数のプロセスを同時に実行することができ、他にもいくつかのオプションがあるので、**man renice**を参照してください。

15.10 研究室



ビデオデモンストレーションのリソース

[renice_demo.mp4](#)を使用。

練習問題 15.1 : ulimitでプロセスを制御する

どうぞよろしくお願いします：

ヘルプ `ulimit`

を読み、[/etc/security/limits.conf](#)を読んでから、以下の手順を実行してください。

1. `bash`と入力して新しいシェルを起動し（または新しいターミナルを開き）、変更が新しいシェルでのみ有効であるようにします。開いているファイル数の現在の制限を表示し、ハードリミットとソフトリミットを明示的に表示します。
2. リミットをハードリミット値に設定し、うまくいったかどうか検証する。
3. ハードリミットを2048に設定し、動作確認。
4. リミットを前の値に戻してみてください。うまくいったか？

解決策 15.1

1. `$ bash`
`$ ulimit -n`

```
1024
```

```
$ ulimit -S -n
```

```
1024
```

```
$ ulimit -H -n
```

```
4096
```

2. `$ ulimit -n hard`
`$ ulimit -n`

```
4096
```

3. `$ ulimit -n 2048`
`$ ulimit -n`

```
2048
```

4. `$ ulimit -n 4096`

```
bash: ulimit: open files: cannot modify limit: 操作が許可されていない。
```

```
$ ulimit -n
```

```
2048
```

もうこれ以上は無理でしょう！

なお、スタックサイズ (-s) のような別の制限を選択した場合、ハードリミットがあるため、再び上昇させることができます。をアンリミテッドにしました。

演習15.2: システムVのIPCアクティビティを検証する

System V IPCは、UNIXの初期に遡る、かなり古いプロセス間通信の方法である。これには3つのメカニズムがあります：

1. 共有メモリセグメント
2. セマフォ
3. メッセージキュー

最近のプログラムでは、これら3つのメカニズムすべてに**POSIX IPC**メソッドを使用する傾向がありますが、**System V IPC**アプリケーションはまだ数多く存在しています。

システム上の**System V IPC**アクティビティの全体的なサマリーを取得するには、次のようにします：

```
$ ipcs
```

```
----- メッセージキュー -----
キー          エムエスアイディアー    オー      使用済みバイト  メッセ

----- 共有メモリセグメント -----。
キー          シミード    オーナー    パーマネ  バイト    搔っ捌    地位
               ント
0x01114703 0      根          600        1000      6
0x00000000 98305   こうじょ    600        4194304   2      デスト
                  うじっし
                  ゆう
0x00000000 196610   こうじょ    600        4194304   2      デスト
                  うじっし
                  ゆう
0x00000000 23068675   こうじょ    700        1138176   2      デスト
                  うじっし
                  ゆう
0x00000000 23101444   こうじょ    600        393216    2      デスト
                  うじっし
                  ゆう
0x00000000 23134213   こうじょ    600        524288    2      デスト
                  うじっし
                  ゆう
0x00000000 24051718   こうじょ    600        393216    2      デスト
                  うじっし
                  ゆう
0x00000000 23756807   こうじょ    600        524288    2      デスト
                  うじっし
                  ゆう
0x00000000 24018952   こうじょ    600        67108864  2      デスト
                  うじっし
                  ゆう
0x00000000 23363593   こうじょ    700        95408     2      デスト
                  うじっし
                  ゆう
0x00000000 1441811   こうじょ    600        2097152   2      デスト
```

うじっし
ゆう

----- セマフォ配列 -----。				
キー	微菌	オーナー	パーマネ ント	エヌエスエム
0x00000000	98304	アパッシ ユ	600	1
0x00000000	131073	アパッシ ユ	600	1
0x00000000	163842	アパッシ ユ	600	1
0x00000000	196611	アパッシ ユ	600	1
0x00000000	229380	アパッシ ユ	600	1

現在実行中の共有メモリセグメントは、ほとんどすべてキーが0（IPC_PRIVATE）であり、親子関係のあるプロセス間でのみ共有されることを意味しています。さらに、1つを除くすべての共有メモリセグメントは、これ以上アタッチメントがない場合、破棄されるようにマークされています。

また、セグメントを作成したプロセスや、セグメントに付随するラストの情報を得ることができます：

\$ ipcs -p

----- メッセージキューPID -----			
msqid	オーナー	エルピーダ	
	エルピーアイディー		
シミ= 共有メモリセグメント/プロセスPID -----			
0	根	1023	1023
98305	こうじょ うじっし ゆう	2265	18780
196610	こうじょ うじっし ゆう	2138	18775
23068675	こうじょ うじっし ゆう	989	1663
23101444	こうじょ うじっし ゆう	989	1663
23134213	こうじょ うじっし ゆう	989	1663
24051718	こうじょ うじっし ゆう	20573	1663
23756807	こうじょ うじっし ゆう	10735	1663
24018952	こうじょ うじっし ゆう	17875	1663
23363593	こうじょ うじっし ゆう	989	1663
1441811	こうじょ うじっし ゆう	2048	20573

このように、することで

\$ ps aux |grep -e 20573 -e 2048

こう	2048 5.3 3.7 1922996 305660 ?	アールエル	Oct27 77:07
こうじ	20573 1.9 1.7 807944 141688 ?	ス	09:56 0:01 /usr/lib64/thunderbird/thunderbird
ようじ		ル	
っしゅ			
う			
こうじ	20710 0.0 0.0 112652 2312 pts/0	S+	09:57 0:00 grep --color=auto -e 20573 -e 2048
ようじ			
っしゅ			
う			

thunderbirdはgnome-shellが作成した共有メモリセグメントを使用していることがわかります。
お使いのシステムでこの手順を実行し、誰が使用しているのか、さまざまなリソースを特定します。潜在的な情報漏えいはないか
(アクティブなプロセスによって使用されなくなった共有リソース)をシステム上で使用していますか？例えば、やること：

\$ ipcs

----- 共有メモリセグメント -----		キー
シミード		

0x00000000 622601		
0x0000001a 13303818		

オーナー	パーマネント	バイト	搔っ捌く	地位
こうじょうじっしゅう	600	2097152	2	デスト
ゆうこうじょうじっしゅう	666	8196	0	

は、アタッチメントがなく、破壊のマークもない共有メモリセグメントを示します。このため、後続のプロセスがアタッチメントを付けなければ、メモリリークを起こしながら永遠に存続する可能性があります。

練習問題15.3: アップタイムとロードアベレージの取得

システムの稼働時間を把握し、ロードアベレージも表示します。

解答 15.3

非常に簡単な方法は、アップタイムユーティリティを使うだけです：

アップタイム

```
10:26:40 up 3:19, 5 users, load average : 1.46, 1.40, 1.19
```

2つ目の方法は、**top**からの出力の1行目を見ることです：

トップ | ヘッド

```
top - 10:28:11 up 3:20, 5 users, load average : 1.93, 1.52, 1.25
タスクのことです：合計313個、          ランニング1本、スリーピング312本、          0が停止しました、
          ゾンビ0匹
CPU : 1.0 us、0.3 sy、0.0 ni、98.2 id、0.5 wa、0.0 hi、0.0 si、0.0
KiB Mem : 合計16284472、空き6556792、使用1029760、バッファ/キャッシュ8697920
KiB スワップ：合計8290300、空き8290300、          0 が使用されました。10364220空きメモリ
PID USER          PR NI      504536 186312 65488 S6      .7 1 .1    6:28.30 skype-b+
18248 コー          20  0      655804 50816 30884 S6      .7 0 .3    0:20.11 エマックス
          プ
          1ルート          20  0      204912          65083956 S0      .0    0:00.92 systemd
                                0 .0
```

第3の方法として、**w**を使用する方法があります：

```
10:30:51 up 3:23, 5 users, load average : 0.55, 1.11, 1.14
ユーザ   ティーテ   フロム          LOGIN@  IDLE   ジェイピーユー   PCPU WHAT
   ィー          ( ログイ
   ン
   こうじょうじっし :0          07:08   ?xdm?16:510   .19s gdm-session-(セ
   ゆう 0          ッション)
   こうじょうじっし :0          07:09   2:22m 0 .12s 0 .12s bash
   しゅう 点/0
   点
   こうじょうじっし :0          07:09   1:37m 0 .42s 0 .42s bash
   しゅう プッツ
   1
   こうじょうじっし :0          07:09   0.00s 51.09s 0 .00s w
   しゅう プッツ
   ツー
   こうじょうじっし :0          07:09   27:08   0.25s 0 .25s バッシュ
   しゅう 点/3
   点
```

練習15.4: 背景と前景のジョブ

ターミナルウィンドウからグラフィカルなプログラムを起動し、ウィンドウ内で文字を入力することができなくなるようにします。

1. のように、新しいファイルで**gedit**を開きます。

```
$ gedit somefile
```

ターミナルウィンドウで文字入力ができなくなりました。

2. ポインタがターミナルウィンドウの上にあるときに、CTRL-Zを押します。

ゼット

```
[3]+停止中          某ファイルの
```

geditのウィンドウで文字入力ができなくなりました。

3. **jobs -l**で、このターミナルウィンドウから起動したプロセスを確認することができます：

```
$ jobs -l
```

```
[1] 17705 実行中      evince *.pdf &  
[2]- 18248 走行中      emacs /tmp/hello.tex &  
[3]+ 19827 停止中      gedit somefile
```

4. ここで、直近のジョブ(**gedit somefile**)をバックグラウンドに置く：

```
$ bg
```



```
[3]+ gedit somefile & です。
```

これで、**gedit**ウィンドウに入力できるようになったはずです。

5. プロセスを再びフォアグラウンドにする：

```
$ fg
```

某ファイルの編集

ターミナルウィンドウで入力できなくなったことに注意してください。

6. クリーンアップするには、再度プロセスをサスペンドし、killで終了させます：

ゼット

```
[3]+停止中          某ファイルの
```

```
$ jobs -l
```

```
[1] 17705 実行中      evince *pdf &
[2]- 18248 走行中      emacs /tmp/hello.tex &
[3]+ 19827 停止中      gedit somefile
```

```
$ kill -9 19827
```

```
$ jobs -l
```

```
[1] 17705 実行中      evince *pdf &
[2]- 18248 走行中      emacs /tmp/hello.tex &
[3]+19827人死亡      gedit somefile
```

```
$ jobs -l
```

```
[1]- 17705 走行中      evince *pdf &
[2]- 18248 走行中      emacs /tmp/hello.tex & で
```

練習15.5: 将来的にバッチ処理にatを使用する場合

今から未来の時刻に実行される非常に簡単なタスクをスケジュールします。これは、lsやdateを実行し、その出力を保存するような簡単なものです。(1分後のような短い時間を使うこともできます)。

コマンドは**at**でスケジュールしたディレクトリで実行されることに注意してください。こうしてください：

1. 短い**bash**スクリプトから。
2. インタラクティブ

ソリューション 15.5

1. を含むファイル**testat.sh**を作成します：

```
#!/bin/bash ( バッシュ
日付 > /tmp/datestamp
```

で実行可能にして、**at**でキューに入れる：

```
$ chmod +x testat.sh
$ at now + 1 minute -f testat.sh
```

ジョブの実行がキューイングされているかどうかは、**atq**で確認することができます：

アットキュー

```
17      Wed Apr 22 08:55:00 2015 学生あるあ
```

実際にジョブが実行されたことを確認します：

```
$ cat /tmp/datestamp
```

```
水曜日 Apr 22 08:55:00 CDT 2015
```

コマンドから `>/tmp/datestamp` を取り除くとどうなるのでしょうか？(ヒント: メールと入力するよう促されていない場合は、メールと入力してください!)

- インタラクティブには、基本的に同じ手順です。でジョブをキューに入れるだけです：

```
現在 + 1分で $
at> date > /tmp/datestamp
CTRL-D
アットキュー
```

練習15.6 cronによる定期的なタスクのスケジューリング

毎日午前10時に何か簡単なタスクを実行する**cron**ジョブを設定する。

解答 15.6

以下の内容で**mycrontab**というファイルを作成します：

```
0 10 * * * /tmp/myjob.sh
```

を作成し、**/tmp/myjob.sh**が含まれるようにします：

```
SH /tmp/myjob.sh

#!/bin/bash ( バッシュ )
echo こんにちは、私は$(date)で$0を実行しています。
```

を作成し、実行可能な状態にします：

```
$ chmod +x /tmp/myjob.sh
```

で**crontab**系に入れる：

```
$ crontab mycrontab
```

と搭載されたことを確認する：

```
$ crontab -l
```

```
0 10 * * * /tmp/myjob.sh
```

```
$ sudo ls -l /var/spool/cron/student
```

```
-rw----- 1 学生 学生 25 Apr 22 09:59 /var/spool/cron/student
```

```
sudo cat /var/spool/cron/student
```

```
0 10 * * * /tmp/myjob.sh
```

もし、あなたが本当に毎日これを実行したくないのであれば、次のようなメッセージをプリントアウトすることに注意してください：

```
こんにちは私は、Wed Apr 22 10:03:48 CDT 2015で /tmp/myjob.sh を実行しています。
```

と郵送すると、一緒に削除することができます：

```
$ crontab -r
```

ある日の午前10時にマシンが起動していない場合、anacronは適切な時間にジョブを実行します。

第16章

プロセスモニタリング



16.1	プロセスモニタリング	248
16.2	トラブルシューティング	249
16.3	ps	251
16.4	pstree	253
16.5	トップ	255
16.6	研究室	257

学習目標

このセッションが終わるころには、次のことができるようになっているはずです：

- コマンドラインのプロセス監視ツールを特定し、使用する。
- システムトラブルのトラブルシューティングを開始する。
- **ps** を使用して、プロセスに関連する特性や統計情報を表示する方法について説明します。
- 異なる **ps** 出力フィールドを識別し、ps 出力をカスタマイズする。
- プロセスの祖先とマルチスレッドアプリケーションの視覚的な説明を得るには、**pstree**を使用します。
- システム負荷をインタラクティブに表示するには、**top**を使用します。

247

16.1 プロセスモニタリング

プロセス監視ツール

- トップ
- アップタイム
- ps
- プスツリー
- エムピーエスタット
- アイオスタット
- サール
- ノルマスタット
- ストレッチャー

Linuxの管理者は、プロセスを監視するために、**ps**、**pstree**、**top**など、UNIX系OSで長い歴史を持つ多くのユーティリティを利用しています。

ここでは、プロセス監視のための主なツールを紹介します：

表16.1：プロセス・負荷監視ツール

ツール	目的
トップ	プロセスアクティビティ、ダイナミックに更新される
アップタイム	システムの稼働時間と平均負荷
ps	プロセスに関する詳細情報
プスツリー	プロセスとその接続を示すツリー
エムピーエスタット	複数プロセッサの使用
アイオスタット	CPU使用率およびI/O統計
サール	システムアクティビティに関する情報を表示・収集する
ノルマスタット	NUMA (Non-Uniform Memory Architecture) に関する情報。
ストレッチャー	プロセスが行うすべてのシステムコールに関する情報

また、**/proc**ファイルシステムは、システム上のプロセスやその他の項目を監視するのに便利です。

16.2 トラブルシューティング

トラブルシューティング・レベル

トラブルシューティングの3つのレベル：

- 初心者の方：非常に早く教えることができる。
- 経験豊富な方：数年の練習を経て登場する。
- ウィザード生まれつきのものだと思っている人もいますが、それはナンセンスです。

どんなに管理されたLinuxシステムでも、問題は発生します。

トラブルシューティングでは、問題がソフトウェアに起因するのかハードウェアに起因するのか、また、問題がシステムのローカルなものなのか、ローカルネットワーク内やインターネットから来るものなのかを切り分けることができます。

トラブルシューティングを適切に行うには、判断力と経験が必要であり、常に芸術の域に達していますが、優れた方法論に従うことで、再現性のある方法で問題の原因を切り分けることができます。

トラブルシューティングの基本テクニック

- 問題の特徴づける。
- 問題を再現する。
- いつも簡単なものから試してみる。
- 考えられる原因を一つずつ排除していく。
- 一度に1つだけ変えてみて、それで問題が解決しない場合は、元に戻す。
- システムログ（`/var/log/messages`、`/var/log/secure`など）を確認し、詳細な情報を確認してください。

直感に基づく飛躍は禁物であり、確立された手順に従う必要がある。チェックリストと統一された手順を使用する動機は、魔法使いに頼ることを避け、どんなシステム管理者でも、よく知られた手順に従えば、最終的に問題を解決できるようにするためである。そうでなければ、魔法使いが組織を去った場合、困難な問題を解決するのに十分なスキルを持った人がいなくなってしまうからです。

一方、直感を尊重し、カンをチェックすることを選択した場合、直感的な道を継続するか放棄するか、生産性が高そうかどうかで判断するために、十分なデータを迅速に取得できるようにする必要があります。

直感を無視すると問題解決に時間がかかることがあります。トラブルシューターのこれまでの実績は、この方法でリソースを投入するかどうかを評価するための重要なベンチマークとなります。つまり、「直感」は魔法ではなく、「経験」なのです。

16.3 ps

psでプロセスの状態を表示する

- **proc**から情報を取得する
- 選択と演出の自由度が極めて高い
- 一般的なオプションの選択もあります：

```
$ ps aux
$ ps -elf
$ ps -eL
$ ps -C "bash"
```

psは、**/proc**ディレクトリから取得したプロセスの特性や統計情報を表示するためのワークホースです。

このコマンド・ユーティリティは、すべての**UNIX系OS**に存在し、その多様性は、**Linux版のps**が受け付ける複雑なオプションの数々にも反映されています：

1. **UNIX**のオプションで、先頭に-を付ける**必要**があり、グループ化されている場合があります。
2. **BSD**のオプションで、-を前につけてはいけ**ない**もので、グループ化されていてもよい。
3. **GNU**ロングオプション。各オプションの前に--を付ける**必要**があります。

このような可能性のあるオプションがあると、生活がかなり混乱することになります。ほとんどのシステム管理者は、日常的に1つか2つの標準的な組み合わせを使用する傾向があります。

```
c8:/tmp>ps auxf
USER      PID  %CPU  %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         2   0.0   0.0      0     0 ?        S    06:43   0:00 [kthreadd]
root         3   0.0   0.0      0     0 ?        I<   06:43   0:00 \_ [rcu_gp]
....
coop    3068   0.0   0.2 846052 38012 ?        SsL   06:53   0:11 \_ /usr/libexec/gnome-terminal-server
coop    3103   0.0   0.0 237040  5648 pts/0    Ss   06:53   0:00 | \_ bash
coop    51808  0.0   0.0 248280  6272 pts/0    S+   13:41   0:00 | | \_ git log
coop    51809  0.0   0.0 219456  2496 pts/0    S+   13:41   0:00 | | \_ /usr/bin/less
coop    3158   0.0   0.0 237512  5980 pts/1    Ss+  06:53   0:00 | \_ bash
c8:/tmp>
```

図16.1：BSDオプションでpsを使う

ps 出力のカスタマイズ

- オプションの後にカンマで区切られたフィールド識別子のリストを付けると、psフィールドの選択されたリストをプリントアウトすることができます。
 - pid: プロセスID
 - uid: プロセス所有者のユーザーID
 - cmd: すべての引数を含むコマンド
 - cputime: 累積CPU時間
 - pmem: マシンの物理メモリに対するプロセスの常駐セットサイズの比率、パーセントで表される
- その他多くの出力オプションについては、**ps**のマニュアルページを参照してください。

```
c8:/tmp>ps -o pid,user,uid,priority,cputime,pmem,size,command
  PID USER      UID PRI    TIME %MEM  SIZE COMMAND
  47619 coop      1000 20 00:00:00  0.0  2656 bash
  49543 coop      1000 20 00:00:00  0.0  1100 ps -o pid,user,uid,priority,cputime,pmem,size,command
c8:/tmp>ps -o pid,user,uid,priority,cputime,pmem,size,command
```

図16.2 : ps出力のカスタマイズ

また、よくある選択肢として、-elf :

```
c8:/tmp>ps -elf
F S UID      PID  PPID  C PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
4 S root        1      0  0  80   0 - 60929 -          06:45 ?          00:00:02 /usr/lib/systemd/systemd --swit
ched-root --system --deserialize 18
1 S root        2      0  0  80   0 -    0 -          06:45 ?          00:00:00 [kthreadd]
1 I root        3      2  0  60 -20 -    0 -          06:45 ?          00:00:00 [rcu_gp]
....
0 S coop       6302   2365  1  80   0 - 182213 -          06:50 tty2        00:02:33 gnome-system-monitor
0 S coop      47591  37378  0  80   0 - 288155 -          07:55 pts/2        00:00:22 evince LFS301.pdf
0 S coop      47602   2089  0  80   0 - 53719 -          07:55 ?          00:00:00 /usr/libexec/evince
....
0 S coop      54206   2715  0  80   0 - 58937 -          09:23 pts/4        00:00:00 bash
0 S root      54295   1140  0  80   0 - 54263 -          09:23 ?          00:00:00 sleep 60
4 R coop      54296   54206  0  80   0 - 67123 -          09:23 pts/4        00:00:00 ps -elf
c8:/tmp>
```

図16.3 : UNIXオプションでpsを使う

16.4 プスツリー

pstreeの使用

- 親子プロセスの関係をユーザーに示す
- プロセスIDを表示する場合は、-pを使用します。
- h [pid]で[pid]とその先祖をハイライトする

`$ pstree -aAps 31478`

```
systemd,1 --switched-root --system --deserialize 21
├─vmplayer,31478
│   ├─{vmplayer},31570
│   ├─{vmplayer},31593
│   ├─{vmplayer},32572
│   └─{vmplayer},32698
```

pstreeは、プロセスの祖先とマルチスレッドアプリケーションを視覚的に説明します：

\$ pstree -aAp 2408

```
バッシュ,2408
|-emacs,24998 pmonitor.tex
|   |-{emacs},25002
|   |   エマックス,25003
|-evince,18036 LFS201-SLIDES.pdf
|   |-{evince},18040
|   |-{evince},18046
|   `-{evince},18047
```

多くのオプションの説明については、**pstree**のmanページを参照してください; 上記では、pid=2408の情報のみを表示することにしました。

子プロセスの1つ（**evince**、pid=18036）には、それ自身の子が3つあることに注意してください。別の見方をすると

\$ ls -l /proc/18036/task

```
合計0
dr-xr-x 5 coop coop 0 Sep 11 07:15 18036
dr-xr-x 5 coop coop 0 Sep 11 07:15 18040
dr-xr-x 5 coop coop 0 Sep 11 07:15 18046
dr-xr-x 5 coop coop 0 Sep 11 07:15 18047
```


16.5 トップ

トップ

- CPU使用率が最も高いプロセスを表示するために使用します。
- プロセスは、最初はCPU使用率でソートされます
- セキュアモード (top s) で実行されない場合、ユーザーはプロセスに信号を送ることができます。
 - kキーを押す
 - プロンプトが表示されたらPIDを与える
 - 指示されたら、信号の番号を教えてください
- 豊富なオプション、古くからあるユーティリティ、豊富なインタラクティブ性
- hキーを押すと、キーバインドリストが表示されます。
- ソートオプションなどを簡単にコントロールできる。

h"を押すとヘルプが表示されます。表示は、多くの異なる基準でソートすることができます。表示は5秒ごとに更新され（ただし、これは設定可能）、ユーザーが中断（Ctrl-C）するまで実行し続ける。

```
top - 09:42:28 up 2:56, 1 user, load average: 1.20, 1.16, 1.09
Tasks: 339 total, 1 running, 338 sleeping, 0 stopped, 0 zombie
%Cpu0 :  1.0 us,  0.3 sy,  0.0 ni, 98.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu1 :  1.3 us,  0.3 sy,  0.0 ni, 98.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu2 :  0.7 us,  0.3 sy,  0.0 ni, 99.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu3 :  0.7 us,  0.3 sy,  0.0 ni, 99.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu4 :  0.3 us,  0.3 sy,  0.0 ni, 99.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu5 :  2.0 us,  0.7 sy,  0.0 ni, 97.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu6 :  1.7 us,  0.3 sy,  0.0 ni, 98.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu7 : 17.7 us,  1.3 sy,  0.0 ni, 81.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem : 15874.0 total, 3964.8 free, 3177.2 used, 8732.0 buff/cache
MiB Swap: 8096.0 total, 8086.7 free, 9.3 used, 11455.8 avail Mem

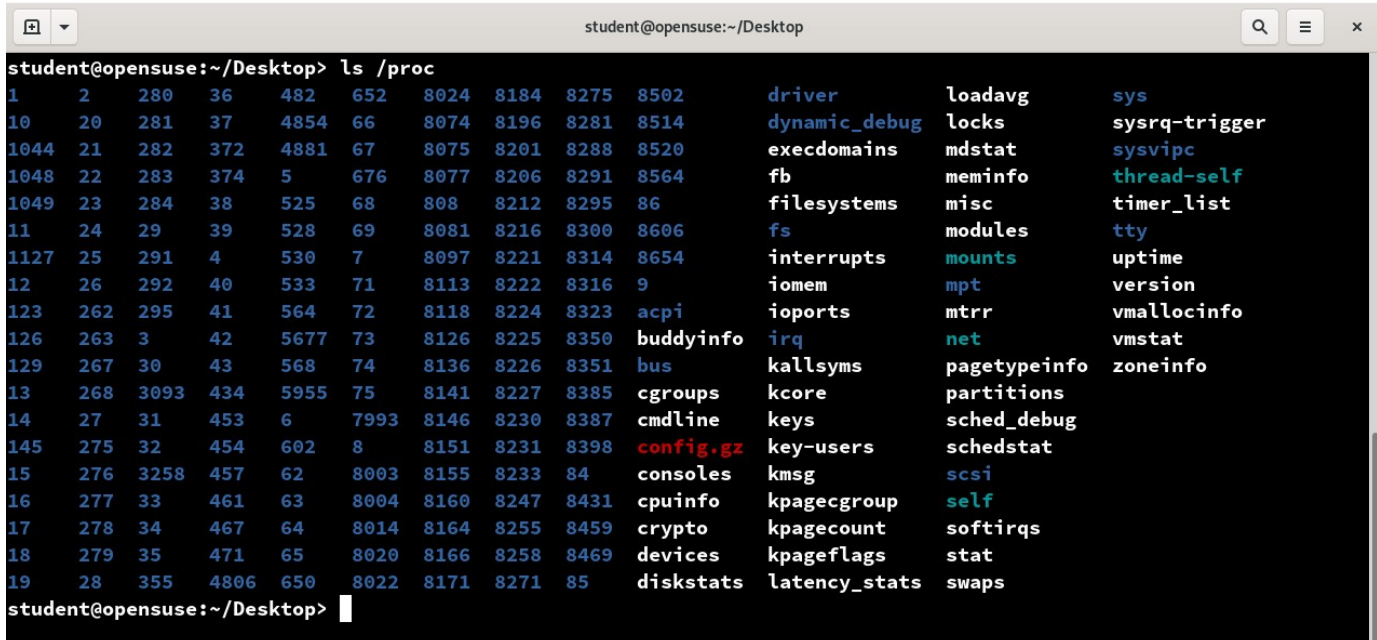
  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM     TIME+ COMMAND
 3605 coop     20    0 3793748 484588 123312 S  18.6   3.0   1:07.44 thunderbird
 2131 coop     20    0 1086964  83868  50788 S   3.3   0.5   8:21.00 Xorg
 2107 coop      9  -11 2729100 20408  13368 S   2.7   0.1  2:10.98 pulseaudio
 6302 coop     20    0 728788  49084  36336 S   1.7   0.3  2:52.34 gnome-system-mo
44118 coop     20    0 3472692 219620 116372 S   1.3   1.4   0:55.83 spotify
 2365 coop     20    0 4663756 432076 73556 S   1.0   2.7   7:15.12 gnome-shell
 3691 coop     20    0  36.8g 244560 89984 S   1.0   1.5   1:54.33 slack
50372 coop     20    0 1225676  67700  58684 S   0.7   0.4   0:25.04 slack
  405 root      0  -20      0      0      0 I   0.3   0.0   0:09.96 kworker/u17:2-i915_flip
2499 root     20    0 414964  31624  9704 S   0.3   0.2   0:08.90 sssd_kcm
44149 coop     20    0 2183944 226052 78704 S   0.3   1.4   0:23.67 spotify
56244 coop     20    0 275384   5384  4544 S   0.3   0.0   0:00.21 top
    1 root     20    0 243716  12144  8300 S   0.0   0.1   0:02.18 systemd
```

図16.4：トップの使用

procの詳細

- `proc`ファイルシステムは、カーネルデータ構造へのインターフェイスである
- `/proc`には、プロセスID (PID) で命名された各アクティブプロセスのサブディレクトリが含まれています。
- `/proc/self`は現在実行中のプロセスです。
- いくつかの調整可能なパラメーターは、`/proc`ディレクトリにあります。
- 詳しくは、`proc`のマニュアルページをご覧ください。

以下のサンプル出力では、`/proc`ファイルシステムのサブディレクトリの名前に数値が含まれていることに注意してください。これらのASCII数値文字列は、システム上のすべてのアクティブなプロセスのPIDを表します。各サブディレクトリには、そのプロセスに関する情報を含む追加のファイルが含まれています。



```
student@opensuse:~/Desktop> ls /proc
1      2      280    36     482    652    8024   8184   8275   8502   driver      loadavg     sys
10     20     281    37     4854   66     8074   8196   8281   8514   dynamic_debug locks       sysrq-trigger
1044   21     282    372    4881   67     8075   8201   8288   8520   execdomains mdstat      sysvipc
1048   22     283    374    5       676    8077   8206   8291   8564   fb          meminfo     thread-self
1049   23     284    38     525    68     808    8212   8295   86     filesystems misc         timer_list
11     24     29     39     528    69     8081   8216   8300   8606   fs          modules     tty
1127   25     291    4       530    7       8097   8221   8314   8654   interrupts  mounts      uptime
12     26     292    40     533    71     8113   8222   8316   9       iomem       mpt         version
123    262    295    41     564    72     8118   8224   8323   acpi     ioports     mtrr        vmallocinfo
126    263    3       42     5677   73     8126   8225   8350   buddyinfo irq          net         vmstat
129    267    30     43     568    74     8136   8226   8351   bus      kallsyms    pagetypeinfo zoneinfo
13     268    3093   434    5955   75     8141   8227   8385   cgroups kcore       partitions
14     27     31     453    6       7993   8146   8230   8387   cmdline keys         sched_debug
145    275    32     454    602    8       8151   8231   8398   config.gz key-users    schedstat
15     276    3258   457    62     8003   8155   8233   84     consoles kmsg        scsi
16     277    33     461    63     8004   8160   8247   8431   cpuinfo  kpagecgroup self
17     278    34     467    64     8014   8164   8255   8459   crypto   kpagecount  softirqs
18     279    35     471    65     8020   8166   8258   8469   devices  kpageflags  stat
19     28     355    4806   650    8022   8171   8271   85     diskstats latency_stats swaps
```

図16.5 : `/proc`の内容

16.6 研究室



ビデオデモンストレーションのリソース

top_demo.mp4
ps_demo.mp4

✎ 練習問題 16.1 : プロセス

1. オプション `-ef` を付けて `ps` を実行します。次に、`aux` オプションを付けてもう一度実行します。出力の違いに注目してください。
2. プロセスID、優先度、ナイス値、プロセスコマンドラインのみが表示されるように `ps` を実行します。
3. コマンドラインで `bash` と入力し、新しい `bash` セッションを開始します。 `nice` コマンドを使用して別の `bash` セッションを開始しますが、今回は `nice` 値を 10 にします。
4. 手順2と同様に `ps` を実行し、優先度と `nice` の値の違いに注意します。2つの `bash` セッションのプロセスIDに注目してください。
5. `renice` を使用して、`bash` セッションの1つの `nice` 値を 15 に変更します。もう一度、優先順位と `nice` 値の変化を観察してください。
6. `top` を実行し、出力が変化するのを観察してください。 `q` を押してプログラムを停止します。

✓ 解決策 16.1

1. `$ ps -ef`
`$ ps aux`
2. `$ ps -o pid,pri,ni,cmd`

```
PID PRI NI CMD
2389      190 バッシュ
22079 19      0 ps -o pid,pri,ni,cmd
```

(注：パラメータとパラメータの間にはスペースを入れてはいけません。)

3. `$ bash`
`$ nice -n 10 bash`
`$ ps -o pid,pri,ni,cmd`

```
2389      190 バッシュ
22115      190 バッシュ
      221719 10 bash
22227   9 10 ps -o pid,pri,ni,cmd
```

4. `$ renice 15 -p 22171`
`$ ps -o pid,pri,ni,cmd`

```
PID PRI NI CMD
2389      190 バッシュ
22115     190バッシュ
          221714 15 bash
22246     4 15 ps -o pid,pri,ni,cmd
```

5. ドルトップ

 練習16.2: プロセス状態の監視

1. **dd**を使用して、**/dev/urandom**から読み取り、**/dev/null**に書き込むバックグラウンドプロセスを開始します。
2. プロセスの状態を確認する。どのような状態であるべきでしょうか？
3. **fg**コマンドでプロセスをフォアグラウンドにする。そして、Ctrl-Zを押す。これは何をするのでしょうか？もう一度、プロセスの状態を見てください、どうなっていますか？
4. ジョブズプログラムを実行してください。何を教えてくれるのでしょうか？
5. ジョブをフォアグラウンドに戻し、別ウィンドウから**kill**で終了させます。

 解答 16.2

1. `$ dd if=/dev/urandom of=/dev/null & .`
2. `$ ps -C dd -o pid,cmd,stat`

```
25899 dd if=/dev/urandom of=/dev/ R
```

SまたはRであるべきです。

3. `$ fg`
ドル `Z`
`$ ps -C dd -o pid,cmd,stat`

```
PID CMD                                エス
25899 dd if=/dev/urandom of=/dev/ T
```

状態はTであるべきです。

4. **jobs**コマンドを入力します。何を教えてくれるのでしょうか？

求人情報

```
[1]+停止中                                dd if=/dev/urandom of=/dev/null
```

5. ジョブをフォアグラウンドに戻し、別のウィンドウから**kill**コマンドを使用してジョブを終了させます。

```
$ fg
$ kill 25899
```

第17章

メモリの監視、使用状況、スワップの設定



17.1	メモリの監視とチューニング	260
17.2	/proc/sys/vm.....	262
17.3	ブイエムスタット	263
17.4	スワップ	264
17.5	アウトオブメモリーキラー (OOM)	265
17.6	研究室.....	267

学習目標

このセッションが終わるころには、次のことができるようになっているはずです：

- メモリチューニングに関わる主な（相互に関連する）考慮事項とタスクを列挙する。
- `proc/sys/vm`のエントリーを使用し、`/proc/meminfo`を解読する。
- メモリ、ページング、I/O、プロセッサの動作、およびプロセスのメモリ消費に関する情報を表示するには、`vmstat`を使用します。
- スワップエリアがどのように利用され、どのようにスワップエリアを作り、監視し、規制するのかを理解する。
- **OOM-killer**がいつアクションを起こすかを決定し、メモリを開放するためにどのプロセスを終了させるべきかを選択する方法を理解する。

17.1 メモリの監視とチューニング

メモリモニタリング

Linuxにおけるメモリの監視とチューニングのための重要なツール

ユーティリティ	目的	パッケージ
自由自在	メモリ使用量の簡単なまとめ	プロップス
バイエムスタット	仮想メモリの詳細な統計とブロックI/O、ダイナミックアップデート	プロップス
パップ	プロセスメモリーマップ	プロップス

フリー -m

	合計	ユー	フリ	共有バッファ／キ		使用可能
メモ	15901	ズド	ー	992	4502	12457
スワッ	8095	2080	9317			

RAMの価格が下がり、性能が向上したのと同時に、システムのメモリリソースに対する要求が高くなりました。

しかし、システム全体のパフォーマンスやスループットのボトルネックがメモリ関連であることはよくあることです。CPUやI/Oサブシステムは、メモリからのデータの取り出しやメモリへの書き込みを待っていることがあります。

メモリに関するシステムの動作を監視、デバッグ、チューニングするためのツールは数多く存在します。

メモリサブシステムのチューニングは、複雑なプロセスになることがあります。まず、メモリ使用量とI/Oスループットは本質的に関連しており、ほとんどの場合、メモリはディスク上のファイルの内容をキャッシュするために使用されていることに注意する必要があります。

このように、メモリパラメータを変更するとI/O性能に大きな影響を与え、I/Oパラメータを変更すると仮想メモリサブシステムに同様に大きな逆効果を与えることがある。

MemTotalです :	16254132	キロ バイト	ライトバック です :	0	キロ バイト	VmallocChunkです :	0	キロ バイト
MemFreeです :	162652	キロ バイト	アノンページズ	2780004	キロ バイト	Percpuです :	8352	キロ バイト
MemAvailableで す :	12234416	キロ バイト	マッピングされ ました :	679632	キロ バイト	HardwareCorruptedです :	0	キロ バイト
バッファース :	847972	キロ バイト	シュメムです :	523880	キロ バイト	アノンヒュージページズ 958464		キロ バイト
キャッシュされる :	10959104	キロ バイト	KReclaimableで す :	1124064	キロ バイト	シュメムフージページ	0	キロ バイト
SwapCachedです :	13276	キロ バイト	スラブです :	1233300	キロ バイト	ShmemPmdMapped :	0	キロ バイト
アクティブで す :	3252848	キロ バイト	SReclaimableで す :	1124064	キロ バイト	FileHugePagesです :	0	キロ バイト
活動休止中で す :	11343484	キロ バイト	スレクレーム :	109236	キロ バイト	FilePmdMappedです :	0	キロ バイト
アクティブ(ア ノ人)です :	393416	キロ バイト	KernelStackで す :	16512	キロ バイト	HugePages_Total :	0	
インアクティ ブ(アノ人) :	2919760	キロ バイト	ページテーブル です :	79092	キロ バイト	HugePages_Freeです :	0	
アクティブ(フ ァイル)で す :	2859432	キロ バイト	NFS_Unstable で す :	0	キロ バイト	HugePages_Rsvd :	0	
Inactive(フ ァ イル)です :	8423724	キロ バイ	バウンスする :	0	キロ	HugePages_Surp :	0	

ト /proc/meminfo

アンエビクタブルです :	75032	キロバイト	WritebackTmpです :	0	キロバイト	ハゲバゲサイズ	2048	キロバイト
\$ cat /proc/meminfo								
Mlockされた :	32	キロバイト	CommitLimit (コミットリミット) です :	16417364	キロバイト	ヒュゲットルブ	0	キロバイト
SwapTotalです :	8290300	キロバイト	Committed_ASです :	10351404	キロバイト	DirectMap4kです :	497740	キロバイト
SwapFreeです :	7911980	キロバイト	VmallocTotalです :	34359738367	キロバイト	DirectMap2Mです :	16152576	キロバイト
汚い :	60	キロバイト	VmallocUsedです :	28256	キロバイト	DirectMap1Gです :	1048576	キロバイト

疑似ファイル `/proc/meminfo` には、メモリがどのように使用されているかという情報が豊富に含まれています。

17.2 /proc/sys/vm

/proc/sys/vm

- 仮想記憶システムを制御するための多くの調整可能なノブを含む
- ほぼすべてのエントリが（rootによって）書き込み可能です
- このディレクトリに何が表示されるかは、カーネルのバージョンによって多少異なります。
- 値を変更するには、エントリに直接書き込むか、またはシスセクトル

`proc/sys/vm`のパラメータを調整する場合、一度に1つずつ調整し、効果を確認するのが通常のベストプラクティスである。主な（相互に関連する）タスクは以下の通りです：

- フラッシングパラメータを制御する。つまり、何ページまでダーティにすることが可能か、何回ディスクにフラッシュアウトさせるか、などである。
- スワップ動作の制御。つまり、ファイルの内容を反映したページをどれだけメモリに残すことができるか、反対に、他にバックストアがないためスワップアウトが必要なページをどれだけ残すことができるか、です。
- 特にCOW（Copy On Write）技術により、多くのプログラムは要求されたメモリ量を完全に必要としないため、メモリのオーバーコミットをどの程度許容するかを制御することができます。

メモリのチューニングは微妙で、あるシステムの状況や負荷ではうまくいっても、他の状況では最適とは言い切れないことがあります。

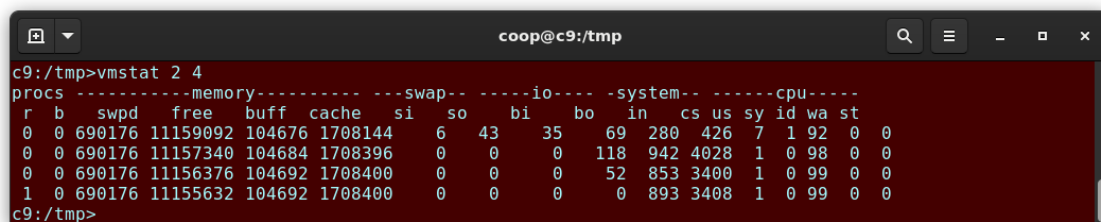
```
student@gentoo:~$ ls /proc/sys/vm
admin_reserve_kbytes      drop_caches              mmap_min_addr            page-cluster
block_dump                extfrag_threshold        mmap_rnd_bits            panic_on_oom
compact_memory            hugepages_treat_as_movable  mmap_rnd_compat_bits    percpu_pagelist_fraction
compact_unevictable_allowed hugetlb_shm_group        nr_hugepages             stat_interval
dirty_background_bytes    laptop_mode              nr_overcommit_hugepages  stat_refresh
dirty_background_ratio    legacy_va_layout         nr_pdflush_threads       swappiness
dirty_bytes               lowmem_reserve_ratio      oom_dump_tasks           user_reserve_kbytes
dirty_expire_centisecs    max_map_count            oom_kill_allocating_task vfs_cache_pressure
dirty_ratio               memory_failure_early_kill overcommit_kbytes         watermark_scale_factor
dirtytime_expire_seconds  memory_failure_recovery  overcommit_memory
dirty_writeback_centisecs min_free_kbytes          overcommit_ratio
```

 17.1 : /proc/sys/vm

17.3 バイエムスタット

バイエムスタット

- メモリ、ページング、I/O、プロセッサのアクティビティ、プロセスに関する情報を表示します。
- 豊富なオプション
\$ vmstat [オプション] [遅延] [カウント]。
- delayは秒単位で、reportは回数単位で繰り返されます。
- 1行目は前回のリポートからの平均値
- 連続する行は、指定された間隔の間の活動を示しています。



```

coop@c9:/tmp
c9:/tmp>vmstat 2 4
procs -----memory----- --swap-- -----io----- -system-- -----cpu-----
r  b  swpd  free  buff  cache  si  so  bi  bo  in  cs  us  sy  id  wa  st
0  0  690176 11159092 104676 1708144 6  43  35  69 280 426 7 1 92 0 0
0  0  690176 11157340 104684 1708396 0  0  0  118 942 4028 1 0 98 0 0
0  0  690176 11156376 104692 1708400 0  0  0  52 853 3400 1 0 99 0 0
1  0  690176 11155632 104692 1708400 0  0  0  0 893 3408 1 0 99 0 0
c9:/tmp>

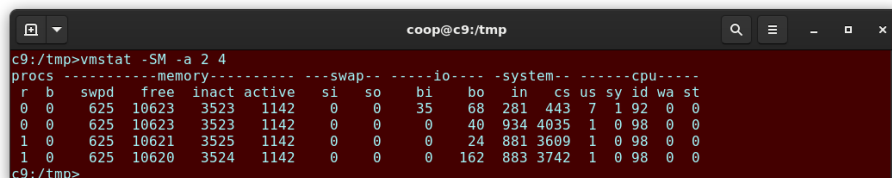
```

図17.2 : vmstat

オプション -S m を指定すると、メモリ統計の単位は KB ではなく MB になります。

アクティブページは最近使用されたページで、クリーン（ディスクの内容が最新）またはダーティ（いずれディスクにフラッシュする必要がある）である可能性があります。これに対して、非アクティブメモリページは、最近使用されていないため、クリーンである可能性が高く、メモリプレッシャーによりすぐに解放されます：

メモリは、アクティブリストとインアクティブリストの間を行ったり来たりし、新たな参照先ができたり、使用間隔が長くなったります。



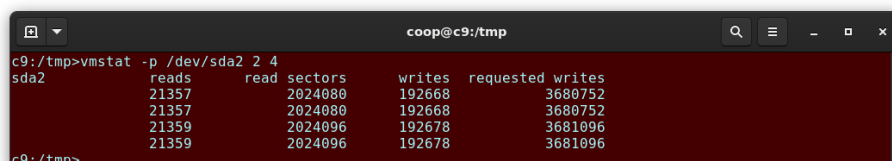
```

coop@c9:/tmp
c9:/tmp>vmstat -SM -a 2 4
procs -----memory----- --swap-- -----io----- -system-- -----cpu-----
r  b  swpd  free  inact  active  si  so  bi  bo  in  cs  us  sy  id  wa  st
0  0  625  10623  3523  1142  0  0  35  68 281 443 7 1 92 0 0
0  0  625  10623  3523  1142  0  0  0  40 934 4035 1 0 98 0 0
1  0  625  10621  3525  1142  0  0  0  24 881 3609 1 0 98 0 0
1  0  625  10620  3524  1142  0  0  0  162 883 3742 1 0 98 0 0
c9:/tmp>

```

図17.3 : vmstatの使用方法

1つのパーティションだけで、簡単な統計を取りたい場合は、-pオプションを使用します。



```

coop@c9:/tmp
c9:/tmp>vmstat -p /dev/sda2 2 4
sda2      reads      read sectors      writes      requested writes
21357      2024080      192668      3680752
21357      2024080      192668      3680752
21359      2024096      192678      3681096
21359      2024096      192678      3681096
c9:/tmp>

```

図17.4 : 1台のディスクでvmstatを使用する場合

17.4 スワップ

スワップ使用

- ディスク上の仮想メモリ
- **fdisk**でスワップパーティションを作成する
- **mkswap**: スワップパーティションまたはファイルのフォーマット
- **swapon**: スワップパーティションまたはファイルをアクティブにします。
- **swapoff**: スワップパーティションやファイルを無効にする。

ファイル名	パーテ	サイズ	中古	優先順
/dev/sda6	ィショ	位 8290300	0	-1
/tmp/swpfile	ンファ	102396	0-2	

Linuxでは、仮想記憶システムを採用しており、OSが実際よりも多くのメモリを搭載しているかのように機能することができます。このようなメモリのオーバーミッションは、2つの方法で機能します：

- 多くのプログラムは、使用許可が与えられているメモリを実際にはすべて使用していません。これは、子プロセスが **COW**（Copy On Write）技術を利用して親のメモリ領域のコピーを継承するため、子プロセスは変更があったときだけ（ページ単位で）ユニークなコピーを取得することになります。
- メモリへの負担が大きくなった場合、あまりアクティブでないメモリ領域をディスクにスワップし、再び必要なときだけ呼び出せるようにすることができます。

このようなスワップは、通常、1つまたは複数の専用パーティションまたはファイルに行われます。Linuxでは、複数のスワップ領域を使用できるため、動的にニーズを調整することができます。各領域には優先順位があり、優先順位の高い領域が埋まるまで、優先順位の低い領域は使用されません。

ほとんどの場合、推奨されるスワップサイズはシステム上の総RAMです。システムが現在使用しているスワップ領域は、`cat /proc/swaps` で確認でき、`free` で現在の使用量を報告することができます。いつでも、ほとんどのメモリは、実際に必要以上にディスクに移動したり、最適でない順序やタイミングで移動したりしないように、ファイルの内容をキャッシュするために使用されています。その代わりにダーティページ（保存されているデータをもはや反映しない、更新されたファイル内容を含むメモリ）がディスクにフラッシュアウトされます。

また、Linuxでは、他のOSと異なり、アプリケーションメモリではなく、カーネル自身が使用するメモリがスワップアウトされることがないことも特筆すべき点である。

17.5 アウトオブメモリーキラー (OOM)

OOMキラー

- システムは物理メモリを使い果たすことがあります。一つはできる：
 1. メモリが解放されるまで、それ以上のメモリ要求を拒否する。
 2. スワップスペースの使用による物理メモリの拡張
 3. 選択したプロセスを（インテリジェントに）終了させることで、メモリの使用量を減らし、システムを存続させる。
- **Linuxシステムでは、2番目と3番目の方法を実装することが最も多い**
- どのプロセスを終了させるかは、**OOMキラー**によって選択される

メモリプレッシャーに対処する最も簡単な方法は、空きメモリがある限りメモリ割り当てを成功させ、すべてのメモリを使い果たしたときに失敗させることであろう。

この場合、（少なくとも理論上は）利用可能なメモリの合計は、実際のRAMとスワップスペースのサイズになります。この場合、（少なくとも理論上は）利用可能なメモリの合計は、実際のRAMとスワップ領域のサイズになります。この場合、難しいのは、圧力がかかったときにスワップアウトするメモリのページを見極めることです。このアプローチでは、スワップスペース自体が埋まってしまうと、新しいメモリへのリクエストは失敗しなければなりません。

しかし、Linuxでは、RAMとスワップのサイズを超えるメモリ要求を許可するオーバーコミットを許可しているのです。これは無謀に思えるかもしれませんが、多くのプロセスは要求されたメモリをすべて使うわけではありません。

例としては、1MBのバッファを確保し、そのうちの数ページしか使用しないようなプログラムが挙げられます。また、子プロセスがフォークされるたびに、親プロセスのメモリ空間全体のコピーを受け取るという例もあります。LinuxはCOW（copy on write）技術を採用しているので、プロセスの1つがメモリを変更しない限り、実際にコピーを作成する必要はない。しかし、カーネルはコピーが必要になる可能性があることを想定しておかなければならない。

このように、カーネルはメモリのオーバーコミットを許可しているが、それはユーザープロセス専用のページに対してのみであり、カーネル内で使用されるページはスワップ可能ではなく、常にリクエスト時に割り当てられる。

OOMキラー・アルゴリズム

- ヒューリスティックアルゴリズムは、通常の操作に依存することを意図したものではありません。グレースフル・シャットダウンやリトレンチメントに必要なものはあるのか？
- プロセスの選択は、次のようなバッドネス値に依存します。
各プロセスの `/proc/[pid]/oom_score` を表示します。
- プロセスの `oom adj スコア` の調整は、各タスクの同じディレクトリで行うことができます。

`proc/sys/vm/overcommit_memory` の値を設定することで、オーバーコミットを変更したり、オフにしたりすることができます：

- 0: (デフォルト) オーバーコミットを許可するが、明らかなオーバーコミットは拒否し、ルートユーザーには通常のユーザーよりいくらか多くのメモリ割り当てを与える。
- 1: すべてのメモリ要求がオーバーコミットできるようになります。
- 2: オーバーコミットをオフにする。メモリコミットの合計が、スワップスペースのサイズと設定可能な割合（デフォルトでは50）のRAMに達した場合、メモリ要求は失敗します。この係数は、`/proc/sys/vm/overcommit_ratio` を変更して変更します。

これについては、Andries Brouwer氏(<https://lwn.net/Articles/104185/>)が面白いことを言っていました：

ある航空機会社が、航空機に搭載する燃料を少なくして飛ばした方が安くなることを発見した。飛行機は軽くなり、燃料も少なくて済むので、経費が節約できる。しかし、まれに燃料が足りなくなり、墜落することがあった。この問題を解決するために、同社の技術者たちは特殊なOOF（燃料切れ）機構を開発した。緊急時には、乗客を選んで機外に放り出す。（排出される犠牲者を適切に選択する問題については、多くの理論が構築され、多くの出版物が出版された。被害者は無作為に選ぶべきなのか？それとも、最も重い人を選ぶべきなのか？それとも年寄りを選ぶべきか？乗客は脱出させないためにお金を払うべきで、そうすれば犠牲者は船内で最も貧しい人になるのか。また、例えば最も重い人が選ばれた場合、それがパイロットであった場合の特例はあるのだろうか？ファーストクラスの乗客は除外されるべきなのか？OOFの仕組みができたからには、燃料不足でなくても、時々作動して乗客を脱出させる。この誤作動がどのように起こるのか、技術者たちはまだ研究中である。

17.6 研究室



ビデオデモンストレーションのリソース

スワップアンドルームデモ.mp4

練習問題17.1：スワップスペースの管理

して、現在のスワップ領域を調べます：

```
$ cat /proc/swaps
```

ファイル名	タイプ別	サイズ	中古	優先順
/dev/sda11	パーティ	位 4193776	0	-1

新しいパーティションまたはファイルを追加して、スワップ領域を追加します。ファイルを使用するには、次のようにします：

```
$ dd if=/dev/zero of=swpfile bs=1M count=1024
```

で1024+0レコード。
1024+0レコードアウト
1073741824バイト (1.1GB) コピー、1.30576秒、822MB/秒

```
$ mkswap swpfile
```

swapspace version 1の設定、size = 1048572 KiB no
label、UUID=85bb62e5-84b0-4 added-848b-4f8a289f0c4c。

(実際のパーティションの場合は、パーティション名をmkswapに与えるだけです。ただし、その上のデータはすべて消去されますのでご注意ください!)新しいスワップ領域を有効にします：

```
sudo swapon swpfile
```

swapon: /tmp/swpfile: insecure permissions 0664, 0600 suggested. swapon:
/tmp/swpfile: insecure file owner 500, 0 (root) suggested.



Red Hat、Centos、またはFedoraの場合

RHELが「安全でない」と警告していることに気づき、本当に修正する必要があります：

```
sudo chown root:root swpfile です。
```

```
sudo chmod 600 swpfile
```

swpfileが使用されていることを確認します：

```
$ cat /proc/swaps
```

ファイル名	パーティ	サイズ	中古	優先順
/dev/sda11	ィシヨ	位 4193776	0	-1
/tmp/swpfile	ンファ	1048572	0-2	

優先順位」フィールドに注意してください。優先順位の低いスワップパーティションまたはファイルは、優先順位の高いものが埋まるまで使用されません。スワップファイルを使用から外し、削除してスペースを確保します：

```
sudo swapoff swpfile
```

```
sudo rm swpfile
```

練習問題17.2: OOMキラーの発動

- Linuxカーネルが極度のメモリ不足に陥ると、恐ろしい**OOM (Out Of Memory) Killer**が起動します。これは、システムが優雅に回復するのを助けるために、殺すべき「最良の」プロセスを選択しようとするものです。
- システムにメモリ不足を強いることで、何が起こるかを観察するのです。まず、ターミナルウィンドウを開き、その中で次のように入力します：

```
$ sudo tail -f /var/log/messages
```

は、カーネルメッセージが表示されたときにそれを見ることができるようにするためです。

- さらに良い方法は、家具があることです：

```
$ dmesg -w
```

は、カーネル以外のメッセージを表示しないためです。

- この演習は、まずコマンドですべてのスワップをオフにすると、より簡単に実行できるようになります：

```
$ sudo /sbin/swapoff -a
```

で後で必ず元に戻してください。

```
$ sudo /sbin/swapon -a
```

- さて、これからシステムにメモリの負荷をかけようと思います。この方法はご自由ですが、メモリを消費するためのプログラムも用意しています：



ラボアシスタント

```
1 /* メモリのデフラグを行う簡単なプログラム、J. Cooperstein
2 2/04
3 */
4
5 #include <stdio.h>
6 #include <stdlib.h>
7 #include <unistd.h>
8 #include <string.h>
9 #include <sys/sysinfo.h>
10 #include <signal.h>
11
12 #define BS 16 /* 各ステップごとにBS*MBを確保しま
13 #define CHUNK (MB*BS)
14 #define QUIT_TIME 20
15 void quit_on_timeout(int sig)
16 {
17     printf("\nTime expired, quitting\n");
18     exit(EXIT_SUCCESS);
19 }
20
21 int main(int argc, char **argv)
22 {
```



```

23 struct sysinfo si ;
24 int j, mとする ;
25 char *c ;
26
27 /* システム上の総メモリを取得する */
28 sysinfo(&si) ;
29 m = si.totalram / MB ;
30 printf("Total System Memory in MB = %d MBn",
31 m = (9 * m) / 10 /* 10%低下させる */
32 printf("Using somewhat less: %d
33 MBentan", m) ;
34
35 m = atoi(argv[1]) ;
36 printf("Choosing instead mem = %d MBn", m) ;
37 }
38
39 signal(SIGALRM, quit_on_timeout)です ;
40 printf("Will quite in QUIT_TIME seconds if no normal
41 terminationn"); alarm(QUIT_TIME) ;
42
43 for (j = 0; j <= m; j += BS) {。
44     /* メモリリークであることは承知していますが、フリーではあり
45     ません、
46     * ということです !
47     */
48     c = malloc(CHUNK) ;
49     /* ただjでブロックを何度も埋めるだけ */
50     memset(c, j, CHUNK) ;
51     printf("%8d", j);
52     fflush(stdout) ;
53 }
54 printf("\n Sleeping for 5 secondsn");
55 sleep(5) ;
56 printf("\n 終了してmemorynを解放する");
57 }

```

引数として、何MB消費するかを受け取ります。システムがメモリを使い果たすまで、要求されるメモリの量を徐々に増やしながら実行し続けましょう。



ご注意ください

するだけで、コンパイルして実行できるはずです :

```
$ gcc -o lab_wastemem lab_wastemem.c
```

```
$ ./lab_wastemem 4096
```

となり、4GBを浪費することになります。実行中に**gnome-system-monitor**や他のメモリ監視プログラムを実行するとよいでしょう（しばらく表示が固まるかもしれませんが！）。

- OOM (Out of Memory)キラーが急襲し、生き残るためにプロセスを殺そうとするのを見るべきです。誰が最初にやられるのか？



解答 17.2

lab_wastemem.c
lab_waste.sh

第18章

I/Oモニタリングとチューニング



18.1	I/Oモニタリング	272
18.2	アイオスタット	273
18.3	ヨートップ	274
18.4	研究室	275

学習目標

このセッションが終わるころには、次のことができるようになっているはずです：

- I/Oアクティビティを監視することの重要性と、それがシステムパフォーマンスのボトルネックになる場合について理解することができる。
- **iostat** を使用して、システムの I/O デバイスのアクティビティを監視します。

271

18.1 I/Oモニタリング

I/Oモニタリングとディスクボトルネック

- 最高のパフォーマンスを得るためには、I/O (Input/Output) アクティビティを監視することが重要です。
- システムは、I/Oバウンド、CPUバウンド、メモリバウンドのいずれでもあり、測定によってのみ区別できる。
- 重要なツールもあります：
 - アイオスタット
 - ヨートップ

ディスクのパフォーマンス問題は、メモリ不足やネットワークのハードウェアやチューニングの不備など、他の要因と強く結びついていることがあります。その切り分けが難しい場合があります。

原則として、CPUがI/Oの完了を待つためにアイドル状態になっていたり、ネットワークがバッファをクリアするために待機していたりする場合、システムはI/Oバウンドとみなされることがあります。

しかし、誤解されることもあります。読み書きに使用されているメモリバッファがいっぱいになると、メモリに問題があるように見えるかもしれませんが、本当の問題はバッファが十分に速く埋まったり空になったりしないことにあります。同様に、ネットワーク転送がI/Oの完了を待っているために、ネットワークのスループットが低下している可能性があります。

リアルタイム監視とトレースは、ディスクのボトルネックを発見し、緩和するために必要なツールです。しかし、稀な問題や繰り返されない問題は、これを達成することを困難にする。

関連する変数が多く、I/Oチューニングは複雑です。

18.2 アイオスタット

アイオスタット

- すべてのI/Oアクティビティを監視するための基本的なユーティリティ
- 非常に柔軟なレポートオプション

\$ iostat [OPTIONS]です。 [デバイス] [インターバル] [カウント]。

```
coop@c8:/tmp
c8:/tmp>iostat
Linux 5.3.1 (c8)          09/30/2019      _x86_64_      (8 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           2.09    0.01    0.33    0.18    0.00   97.39

Device            tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
sda                0.44        242.46         80.25    5452482     5632
sdb                6.73         84.91         86.80    1909413    1952028
sdc                3.17         66.42        212.01    1493574    4767684
dm-0               0.17         28.37         30.31     638049     6892
dm-1               0.22        213.06         30.00    4791481         60
dm-2               0.01         30.15         30.00         3357         12
dm-3               0.01         30.16         30.00         3545         12
dm-4               0.00         30.06         30.00         1404          0
dm-5               0.00         30.00         30.00          82          0
dm-6               0.01         30.15         30.00         3412         12
dm-7               0.00         30.00         30.00          82          0
dm-8               0.00         30.06         30.00         1404          0
loop0             41.24        41.29         30.00     928544          0

c8:/tmp>
```

図18.1 : iostat

iostatは、システム上のI/Oデバイスの活動を監視するための基本的な主力ユーティリティです。多くの情報を含むレポートを作成することができ、その正確な内容はオプションで制御することができます。

CPU使用率を簡単にまとめた後、I/O統計として、tps（1秒あたりのI/Oトランザクション、論理リクエストは1つの実際のリクエストに統合できる）、単位時間あたりの読み書きのブロック（ブロックは通常512バイトのセクター）、読み書きのブロック総数が示されます。

情報はディスクパーティション別（LVMが使用されている場合はdm（デバイスマッパー）論理パーティション別）に分けられます。より詳細なレポートです：

```
coop@c8:/tmp
c8:/tmp>iostat -xk
Linux 5.3.1 (c8)          09/30/2019      _x86_64_      (8 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           2.20    0.01    0.34    0.18    0.00   97.28

Device            r/s    w/s    rkB/s    wkB/s    rrqm/s    wrqm/s    %rrqm    %wrqm    r_await    w_await    aqu-sz    rareq-sz    wareq-sz    svctm    %util
sda                0.35    0.08    232.35     30.25     0.00     0.00     0.00     0.00    16.41    73.69    127.10     662.98     3.08     1.07     0.05
sdb                2.33    5.30    81.40    90.98     0.02     7.65     0.86    59.06     3.30     1.03     34.90    17.17     0.38     0.29
sdc                1.45    1.65    67.78    202.40     0.08    1.94     5.28    54.02     1.04    13.65     46.75    122.37     0.30     0.09
dm-0               0.09    0.08    28.66     30.31     0.00     0.00     0.00     0.00    165.62    131.17     304.18     3.99     1.10     0.02
dm-1               0.21    0.00    202.69     30.00     0.00     0.00     0.00     0.00    35.93    21.90     972.10     2.86     1.21     0.03
dm-2               0.01    0.00     30.14     30.00     0.00     0.00     0.00     0.00    19.50    11.89     17.86     1.33     0.48     0.00
dm-3               0.01    0.00     30.15     30.00     0.00     0.00     0.00     0.00    19.05    20.00     18.56     1.33     0.42     0.00
dm-4               0.00    0.00     30.06     30.00     0.00     0.00     0.00     0.00    36.25     30.00     22.29     1.33     0.82     0.00
dm-5               0.00    0.00     30.00     30.00     0.00     0.00     0.00     0.00    50.36     30.00     2.93     0.00     0.86     0.00
dm-6               0.02    0.00     30.18     30.00     0.00     0.00     0.00     0.00    118.46    18.33     10.69     1.33     0.27     0.00
dm-7               0.00    0.00     30.00     30.00     0.00     0.00     0.00     0.00     78.43     30.00     2.93     0.00     0.79     0.00
dm-8               0.00    0.00     30.06     30.00     0.00     0.00     0.00     0.00     49.84     30.00     22.29     0.00     0.70     0.00
loop0             39.27    0.00    39.31     30.00     0.00     0.00     0.00     0.00     30.06     30.00     1.00     0.00     0.01     0.06

c8:/tmp>
```

図18.2 : iostatの使用

18.3 ヨートップ

ヨートップ

- **iostat**は現在のI/Oアクティビティを表示します。
- トップと同様に定期的に更新
- オプションを使用すると、乱雑にならないようにするのに便利です。

File Edit View Search Terminal Help

Total DISK READ :			116.67 M/s	Total DISK WRITE :			132.23 K/s
Actual DISK READ :			116.67 M/s	Actual DISK WRITE :			0.00 B/s
TID	PRI	USER	DISK READ	DISK WRITE	SWAPIN	IO>	COMMAND
17932	be/4	root	0.00 B/s	0.00 B/s	0.00 %	99.99 %	[kworker/2:0]
3571	be/4	coop	0.00 B/s	0.00 B/s	0.00 %	99.99 %	skype-bin
15601	be/4	coop	0.00 B/s	81.67 K/s	0.00 %	99.99 %	vmware-vmx -ssnapshot.num-ntu-17-04.vmx [vmx-vcpu-3]
19800	be/4	coop	0.00 B/s	0.00 B/s	0.00 %	99.99 %	gnome-screenshot -i -w
17186	be/4	coop	0.00 B/s	46.67 K/s	0.00 %	0.00 %	vmware-vmx -ssnapshot.num-17-04.vmx [vmx-vthread-16]
15598	be/4	coop	0.00 B/s	3.89 K/s	0.00 %	0.00 %	vmware-vmx -ssnapshot.num-ntu-17-04.vmx [vmx-vcpu-0]
19782	be/4	root	116.67 M/s	0.00 B/s	0.00 %	0.00 %	cat ./VIRTUAL/FC-25-LATEX/FC-25.vmdk
1	be/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	systemd --switched-root --system --deserialize 21
2	be/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[kthreadd]
4	be/0	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[kworker/0:0H]
6	be/0	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[mm_percpu_wq]
7	be/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[ksoftirqd/0]

図18.3 : iostatの使用

もう一つの非常に便利なユーティリティは**iostat**で、rootとして実行する必要があります。これは、現在のI/O使用量の表を表示し、定期的に更新します。PRIOの列で、beは**best effort**、rtは**real time**を表しています。

```

student@ubuntu: ~
student@ubuntu:~$ iostat --help
Usage: /usr/sbin/iostat [OPTIONS]

DISK READ and DISK WRITE are the block I/O bandwidth used during the sampling
period. SWAPIN and IO are the percentages of time the thread spent respectively
while swapping in and waiting on I/O more generally. PRIO is the I/O priority at
which the thread is running (set using the ionice command).

Controls: left and right arrows to change the sorting column, r to invert the
sorting order, o to toggle the --only option, p to toggle the --processes
option, a to toggle the --accumulated option, i to change I/O priority, q to
quit, any other key to force a refresh.

Options:
--version          show program's version number and exit
-h, --help        show this help message and exit
-o, --only        only show processes or threads actually doing I/O
-b, --batch       non-interactive mode
-n NUM, --iter=NUM number of iterations before ending [infinite]
-d SEC, --delay=SEC delay between iterations [1 second]
-p PID, --pid=PID processes/threads to monitor [all]
-u USER, --user=USER users to monitor [all]
-P, --processes  only show processes, not all threads
-a, --accumulated show accumulated I/O instead of bandwidth
-k, --kilobytes  use kilobytes instead of a human friendly unit
-t, --time       add a timestamp on each line (implies --batch)
-q, --quiet      suppress some lines of header (implies --batch)
student@ubuntu:~$

```

図18.4 : iostatのオプション

18.4 研究室

練習問題18.1 : ボニープラスプラス

bonnie++は、ドライブやファイルシステムの性能をテスト・測定する、広く利用可能なベンチマーク・プログラムです。以前の実装である**bonnie**の子孫にあたります。

結果はターミナルウィンドウからの読み込みやファイルへの指示、また**csv**形式（カンマ区切り値）への変換が可能です。コンパニオンプログラムである**bon.csv2html**と**bon.csv2txt**を使用して、htmlとプレーンテキストの出力形式に変換することができます。

どのテストを実行するか、どの程度網羅的でストレスのかかるテストを行うかについて、かなり多くのオプションがあるため、使用前に**bonnie++**の**マニュアルページ**を読むことをお勧めします。簡単なあらすじは、次のようにして得られます：

```
$ bonnie++ --help
```

```
bonnie++: 無効なオプション -- '-'
の用法 :
bonnie++ [-d scratch-dir] [-c concurrency] [-s size(MiB)[:chunk-size(b)]] 。
          [-n number-to-stat[:max-size[:min-size]][:num-directories[:chunk-size]] ]
          。 [-m machine-name] [-r ram-size-in-MiB] です。
          [-x number-of-tests] [-u uid-to-use:gid-to-use] [-g gid-to-use] です。
          [q] [-f] [-b] [-p processes | -y] [-z seed | -Z random-file] [-
          D].
```

```
バージョン : 1.98
```

というようなコマンドで簡単なテストが得られます：

```
$ time sudo bonnie++ -n 0 -u 0 -r 100 -f -b -d /mnt
```

のところです：

- **-n 0**は、ファイル作成テストを行わないことを意味します。
- **-u 0**はrootで実行することを意味します。
- **-r 100**は、100MBのRAMを搭載しているように見せかけることを意味します。
- **-f**は、1文字ごとのI/Oテストをスキップすることを意味します。
- **-b**は、書き込みのたびに**fsync**を行うことを意味し、キャッシュに書き込むだけでなく、ディスクにフラッシュすることを強制します。
- **-d /mnt**は、作成された一時ファイルを置くディレクトリを指定するだけで、十分な容量（この場合は300MB）があることを確認してください。

もし、あなたがメモリサイズの数字を提供しない場合、プログラムはシステムの容量を計算し、2-3倍の大きさのテストファイルを作成します。これは、感覚をつかむのに時間がかかるため、ここでは行いません。



レッドハットについて

RHEL/CentOS 9では、現在利用可能なパッケージはありませんが、以前のバージョンは動作し、**EPEL 8**のリポジトリから入手できます：https://dl.fedoraproject.org/pub/epel/8/Everything/x86_64/Packages/b/bonnie++-1.98-1.el8.x86_64.rpm でインストールします：

```
$ sudo dnf localinstall bonnie++.*rpm
```



```
c8:/tmp>sudo bonnie++ -n 0 -u 0 -r 100 -f -b -d /mnt
```

```
uid:0、gid:0を使用しています。
知的に書く...済 リライト...済
知的に読む...完了
を開始する...完了...完了...完了...完了...。

バージョン          -----逐次出力----- --逐次入力-----ランダム-----。
1.98                -Per Chr- --Block-- -Rewrite- -Per Chr- --Block -----シーク-
                    -。
名称：サイズ      300M          271m 20 288m 15          ++++++ +++ 3916 22
など c8            30歳          二一              12倍   18075us

1.98,1.98,c8,1,1616174245,300M,,8192,5,,,277062,20,294543,15,,,+++++,+++,3916,22,,,,,,,,,,,,,\
30us,20us,,12us,18075us,,,,,
```



Ubuntuの場合

Ubuntu 20.04のシステムで、同じ物理マシンのVMwareハイパーバイザーの下で仮想マシンとして動作しています：

```
$ sudo bonnie++ -n 0 -u 0 -r 100 -f -b -d /mnt
```

```
sing uid:0, gid:0.
知的に書く...済 リライト...済
知的に読む...完了
を開始する...完了...完了...完了...完了...。

バージョン          -----逐次出力----- --逐次入力-----ランダム-----。
1.98                -Per Chr- --Block-- -Rewrite- -Per Chr- --Block -----シーク-
                    -。
名前：サイズ      300M          287m 88 ++++++ +++          ++++++ +++ ++++++ +++
など ubuntu        3805us   四六半時          2136us   1852年

1.98,1.98,ubuntu,1,1616156268,300M,,8192,5,,,293955,88,+++++,+++,,,,,,+++++,+++,+++++,+++,,,,,,\
,,,,3805us,4650us,,2136us,1852us,,,,,
```

パフォーマンスが低下しているのがよくわかります。これまでの出力を**bonnie++.out**というファイルとして保存したと仮定して、出力をhtmlに変換することができます：

```
$ bon_csv2html < bonnie++.out > bonnie++.html
```

でプレーンテキストに変換することもできます：

```
$ bon_csv2txt < bonnie++.out > bonnie++.txt
```

ドキュメントを読んだら、より長く、より大きく、より野心的なテストを試してみてください。オフにしたテストのいくつかを試してみてください。システムの挙動が良好な場合は、将来システムが病気になったときのベンチマーク比較のために結果を保存しておいてください。



練習問題18.2: fsマーク

fs markベンチマークは、複数のディレクトリやドライブで非同期I/Oを多用し、ファイルシステムに低レベルのバッシングを与えるものです。これはRick Wheelerによって書かれたかなり古いプログラムですが、時の試練に耐えてきました。<http://sourceforge.net/projects/fsmark/> からダウンロードできます。tarballを入手したら、それを解凍してコンパイルします：

```
tar zxvf fs_mark-3.3.tgz
$ cd fs_mark
$ make
```

表面しか触れないので、READMEファイルを読んでください。というようなエラーでコンパイルが失敗した場合：

```
$ make
```

```
----
/usr/bin/ld: -lc が見つかりません。
```

これは、静的バージョンのglibcがインストールされていないためです。Red Hatベースのシステムでこれを行うには、次のようにします：

```
$ sudo dnf install glibc-static
```

とSUSE関連システムで使うことができます：

```
sudo zypper install glibc-devel-static.
```

Debianベースのシステムでは、関連する静的ライブラリは共有ライブラリと一緒にインストールされるため、追加のパッケージを探す必要はありません。



Red Hat、Centos、またはFedoraの場合

- RHEL 8 では、glibc-static が codeready-builder リポジトリで利用可能で、これを有効にする必要がある場合があります。
- しかし、CentOS 8 では、PowerTools レポに含まれています。もし、あなたのシステムにすでに [/etc/yum.repos.d/ CentOS-PowerTools.repo](#) があれば、そのファイルに `enabled=1` が設定されていること

テストのために、それぞれ10KBのサイズの2500個のファイルを作成し、各書き込みの後にfsyncを実行してディスクにフラッシュアウトするつもりです。これは、`/tmp`ディレクトリでコマンドを実行することができます：

```
$ fs_mark -d /tmp -n 2500 -s 10240
```

この実行中に、拡張 `iostat` 統計を収集します：

```
$ iostat -x -d /dev/sda 2 10
```

を別のターミナルウィンドウで表示します。

注意すべきは、fs markが報告する1秒あたりのファイル数と、iostatが報告する利用されている帯域幅のパーセンテージです。これが100%に近づいている場合は、I/Oバウンドしていることになります。

第19章

仮想化の概要



19.1	仮想化入門.....	280
19.2	ホストとゲスト.....	282
19.3	エミュレーション.....	283
19.4	ハイパーバイザー.....	285
19.5	リブパート.....	289
19.6	けいさんきそうち.....	291
19.7	KVM.....	294
19.8	クラウドイット.....	296
19.9	研究室.....	299

学習目標

このセッションが終わるころには、次のことができるようになっているはずです：

- 仮想化の概念と普及の経緯を理解する。
- ホストとゲストの役割を理解する。
- エミュレーションと仮想化の違いについて説明する。
- ハイパーバイザーの種類を区別する。
- Linuxディストリビューションがlibvirtをどのように使用し、どのように依存しているかを熟知している。
- qemuハイパーバイザーを使用します。
- KVMのインストール、使用、管理。

279

19.1 仮想化入門

仮想化とは？

- 抽象化レイヤーのことです：
 - 物理リソースが仮想リソースに置き換わる
- 種類は以下の通りです：
 - オペレーティングシステム（例：仮想マシン、VM）
 - ネットワーク（例：Software Defined Networking, SDN）
 - ストレージ（ネットワークアタッチドストレージ、NASなど）
 - アプリケーション（例：コンテナなど）
- 抽象化層は、ハードウェアではなく、オペレーティングシステム固有のソフトウェアに置き換わる可能性があります。

このセクションでは、**仮想マシン（VM）**の作成、展開、および保守について説明します。仮想マシンは、オペレーティングシステム全体を仮想化したインスタンスで、サーバーやデスクトップ/ワークステーションの役割を果たすことができます。

外界は、VMをあたかもネットワーク上のどこかに存在する実際の物理的なマシンであるかのように見ます。VM内で動作するアプリケーションは、一般的に非物理的な環境を意識することはない。

その他、仮想化には以下のような種類があります：

- **ネットワークです：**
ハードウェアの種類やルーターなど、実際の物理ネットワークの詳細は抽象化されており、その上で動作するソフトウェアやそれを構成するソフトウェアが知る必要はない。
- **ストレージです：**
複数のネットワーク・ストレージ・デバイスは、ディスクのような1つの大きなストレージ・ユニットのように見えるように構成されます。
- **アプリケーションです：**
アプリケーションは、後述する**コンテナ**などのスタンドアローン形式に分離されます。

物理マシンと仮想マシンの間には、必ずしも無視できない違いがあります。例えば、低レベルでのパフォーマンスチューニングは、VMとその下に存在する物理マシンの両方に対して（別々に）行う必要があります。

仮想化の歴史

- 数十年前にメインフレームに実装されたのが始まりです：
 - ハードウェアの有効活用を可能にする
 - オペレーティングシステムは、ハードウェアよりも早く進歩することが多い
 - マイクロコード駆動
 - 特に使いやすくない
- 仮想化技術がPCやワークステーションに移行した：
 - 当初は**Emulation**で行っていた
 - 仮想化をサポートするために強化されたCPUは、パフォーマンスの向上、設定の容易さ、VMのインストールと移行の柔軟性を実現しました。

仮想化には長い誇れる歴史があります。初期のメインフレームからミニコンピュータに至るまで、仮想化は制限の拡大、デバッグ、管理の強化のために利用されてきました。

今日、仮想化は様々な形で至る所に見られるようになりました。

さまざまな形態や実装がありますが、それぞれ特有の利点があります。

19.2 ホストとゲスト

ホストとゲスト

- **ホストです：**
1つまたは複数の仮想マシンを管理する基礎となる物理的なオペレーティングシステム。
- **ゲストの方です：**
1つまたは複数のアプリケーションを実行する、完全なオペレーティングシステムのインスタンスであるVM。クライアントと呼ばれることもある。
- ゲストが特定のホストを気にしない、知らない
- ゲストを別のホストに移行することが可能
- パフォーマンスチューニングは、ゲストだけでなくホストでも行う必要があります。

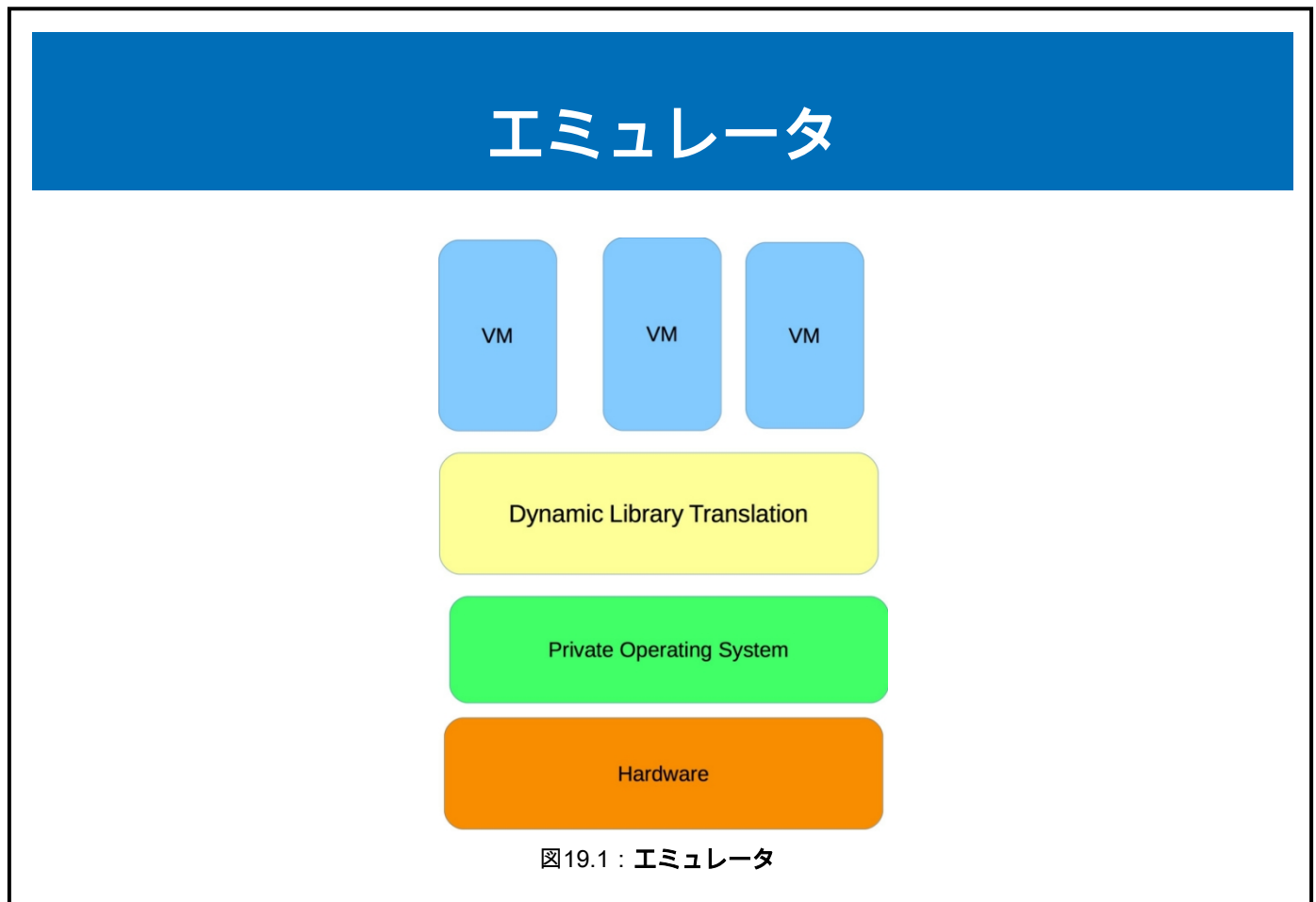
ほとんどの場合、ゲストはどのホストで実行されているかを気にする必要はなく、あるホストから別のホストへ、時には実際に実行しながら移行することができます。

実は、物理マシンの中身を丸ごとVMにコピーして、仮想マシンに変換することが可能です。専用のソフトウェア・ユーティリティを使えば、これを簡単に行うことができます。

CPU使用率、ネットワークスループット、メモリ使用率などの低レベルのパフォーマンスチューニングは、多くの場合、ホスト上で行うのがベストです。

アプリケーションのチューニングは、ほとんどゲストで行うことになります。

19.3 エミュレーション



PCアーキテクチャにおける仮想化の最初の実装は、**エミュレータ**を使用することでした。現在のOS上で動作するアプリケーションは、別のOSからは特定のハードウェア環境として見えることになります。エミュレータは一般に、動作に特別なハードウェアを必要としません。

Qemuはそのようなエミュレータの1つです。

エミュレーションと仮想化の比較

- エミュレータは、完全にソフトウェアで動作します。
- ハードウェアの構成がソフトウェアに置き換わる
- 異なるアーキテクチャの仮想マシンを実行するのに便利：例えば、**x86**ホスト上で**ARM**ゲストマシンを実行する場合など
- ハードウェアが完成する前から、新しいCPUに対応したOSを開発するために使われることが多い。
- 性能は比較的遅い

19.4 ハイパーバイザー

仮想化の種類 ハイパーバイザー

- 仮想マシンの下で動作し、仮想マシンを管理する層
- 仮想マシンモニター (VMM) と呼ばれる
- 複数のVMがハードウェアリソースを透過的に共有できるようにする。
- 基本は2種類：
 - **ハードウェアの仮想化**
ゲストシステムは、VMとして動作していることに気づかない。完全仮想化とも呼ばれる
 - **準仮想化**
ゲストシステムは、VMとして動作していることを認識し、より良い動作ができるように変更されています。
- 最近のほとんどのハードウェアは、ハードウェア仮想化能力を備えている (BIOSでオンにする必要がある)

ホストシステムは、実行するソフトウェアに関して正常に機能する以外に、ゲストを開始、終了、管理するハイパーバイザーとして機能する。仮想化には2つの基本的な方法があります：

- **ハードウェア仮想化**：ゲストシステムは、仮想化されたゲストとして動作していることを意識することなく動作し、この方法で動作させるための改造は必要ありません。
- **パラ仮想化**：：ゲストシステムは、自分が仮想化環境で動作していることを認識し、仮想化環境で動作するように特別に変更されている。

IntelとAMDの広く使われているCPUは、x86アーキテクチャに仮想化拡張機能を組み込んでおり、ハイパーバイザーが完全に仮想化された (つまり、変更されていない) ゲストOSを、わずかな性能低下で実行できるようになっています。

Intel extension (Intel Virtualization Technology) は、通常VT、IVT、VT-32、VT-64と略され、開発コード名Vanderpoolでも知られています。2005年春から利用できるようになりました。AMDの拡張機能は通常AMD-Vと呼ばれ、現在も開発コード名Pacificaと呼ばれることがあります。これら2つの拡張機能がどのように機能するかについての詳しい説明と比較は、<https://lwn.net/Articles/182080/> を参照してください。

CPUがハードウェア仮想化拡張機能をサポートしているかどうかは、`/proc/cpuinfo`で直接確認できます。IVT対応チップの場合は、flagsフィールドにvmx、AMD-V対応チップの場合は、同じフィールドにsvmと記載されています。また、CMOSで仮想化機能がオンになっていることを確認する必要があるかもしれません。

パラ仮想化ゲストは、オペレーティングシステムの選択肢がより限定される傾向にありますが、元来、完全仮想化ゲストよりも効率的に動作する傾向がありました。仮想化技術の進歩により、このような利点は縮小または解消され、完全仮想化に必要なハードウェアサポートが広く利用できるようになったため、準仮想化の利点は少なくなり、普及も進んでいない。

外部ハイパーバイザーとカーネル内ハイパ

- ハイパーバイザーは、ホストOSのカーネルに対して外付けにすることができます：
ゼン
- ハイパーバイザーは、ホストOSカーネルに内蔵することができます：
KVM
- このコースでは、すべてオープンソースであり、外部のサードパーティ製ハイパーバイザープログラムを必要としない**KVM**に焦点を当てます。

内部ハイパーバイザーと外部ハイパーバイザーの主な違いはアーキテクチャで、Xenはマクロカーネルアーキテクチャを採用し、共有リソースを管理する小さなカーネルを使用しています。Xenカーネルは、まずgrubブートローダーによってロードされ、その後、Linuxカーネルが仮想マシンとしてロードされます。この初期VMは、Xenカーネルを設定・制御する機能を持ち、通常Domain0と呼ばれます。外部というラベルは、Xenが最初にロードされ、次にLinuxがロードされることに由来します。

これに対し、KVMはLinuxカーネルの一部であり、カーネルモジュールによって起動される。共有リソースはLinuxカーネルによって管理されます。KVMには独立したカーネルが存在しないため、内部とみなされます。

具体的な違いについては議論があると思いますが、KVMもXenもLinuxにハイパーバイザー機能を提供しています。

専用ハイパーバイザー

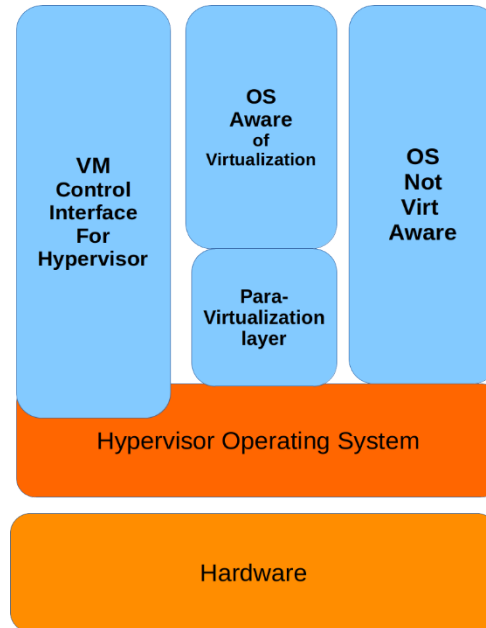


図19.2 : 専用ハイパーバイザー

エミュレーションの次は、ハイパーバイザーのプログラムを特別に設計された軽量なカーネルに統合することが、仮想化の展開における次のステップとなります。

Xenと**VMware ESX**（とその仲間たち）は、OSに組み込まれたハイパーバイザーの例です。

カーネル内のハイパーバイザー

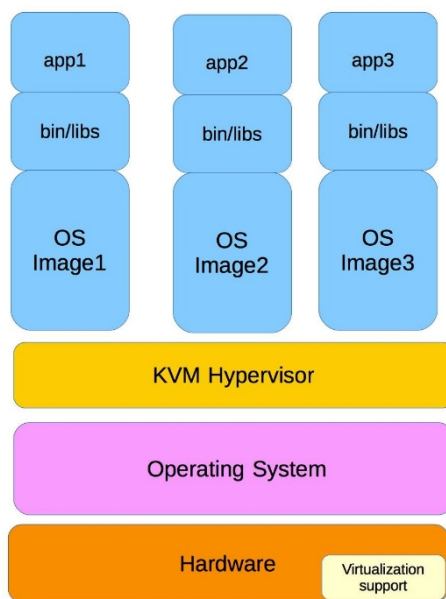


図19.3 : カーネル内のハイパーバイザー

KVMプロジェクトは、**Linux**カーネルにハイパーバイザーの機能を追加しました。これにより、カーネルの機能を活用し、カーネルがハイパーバイザーとなることを可能にした。

これまで述べてきたように、このような仮想化のためには、特定のCPUチップの機能や設備が必要とされ、配備されていたのです。

19.5 リブバート

リブバート

- 仮想化技術に触れるためのツールキット
- 多くのハイパーバイザーに対応した共通APIインターフェイス
 - KVM/QEMU
 - LXC
 - バーチャルボックス
 - ブイエムウェア
 - 等々
- のマネジメントを提供する。
 - 仮想マシン
 - 仮想ネットワーク
 - ストレージ
- すべてのエンタープライズLinuxディストリ

ビューションで利用可能 <https://www.libvirt.org>

多くのアプリケーションプログラムが libvirt とインターフェースを取っており、最も一般的なものとして **virt-manager**, **virt-viewer**, **virt-install**, **virsh** が挙げられます。現在サポートされているハイパーバイザーの全リストは <https://www.libvirt.org/drivers.html> で見るができます。

• LXC (Linuxコンテナ)	• VMware Workstation/Player
• オープンブイセツ	• ゼン
• QEMU/KVM/HVF	• マイクロソフトハイパーバイ
• テスト (テストに使用 します)	• ヴィルトゥオツツォ
• バーチャルボックス	• Bhyve (The BSD Hypervisor)
• ブイエムウェアイーエ スエックス	• クラウドハイパーバイザー

と、より詳細で素敵なリストは、<https://wiki.qemu.org/Documentation/Platforms> で見るができます。

libvirtを使用したプログラム

- 多くのユーティリティがlibvirtを使用しています：

```
student@ubuntu: ~  
student@ubuntu:~$ ls -lF /usr/bin/virt*  
-rwxr-xr-x 1 root root 63840 Aug 9 08:50 /usr/bin/virt-admin*  
-rwxr-xr-x 1 root root 57 Dec 14 2016 /usr/bin/virt-clone*  
-rwxr-xr-x 1 root root 59 Dec 14 2016 /usr/bin/virt-convert*  
-rwxr-xr-x 1 root root 51440 May 10 18:25 /usr/bin/virtfs-proxy-helper*  
-rwxr-xr-x 1 root root 18480 Aug 9 08:50 /usr/bin/virt-host-validate*  
-rwxr-xr-x 1 root root 59 Dec 14 2016 /usr/bin/virt-install*  
-rwxr-xr-x 1 root root 908632 Aug 9 08:50 /usr/bin/virt-login-shell*  
-rwxr-xr-x 1 root root 59 Dec 14 2016 /usr/bin/virt-manager*  
-rwxr-xr-x 1 root root 11191 Aug 9 08:49 /usr/bin/virt-pki-validate*  
-rwxr-xr-x 1 root root 55 Dec 14 2016 /usr/bin/virt-xml*  
-rwxr-xr-x 1 root root 4700 Aug 9 08:49 /usr/bin/virt-xml-validate*  
student@ubuntu:~$ \
```

図19.4 : libvirtベースのユーティリティ

- 正確なリストは、お使いのLinuxディストリビューションに依存します。
- 全リストは、<https://www.libvirt.org/apps.html> でご覧いただけます。

このコースでは、コマンドラインユーティリティを多用するのではなく、堅牢なGUIであるvirt-managerを使用して作業を進めますが、これはより柔軟で、グラフィカルでないサーバーでも使用することができます。

19.6 けいさんきそうち

QEMUとは？

- クイックEMUlator
- 3段階です：
 1. **libvirt**：共通ライブラリとAPI
 2. **QEMU**：エミュレータ
 3. **KVM**です：アクセラレータ
- ハードウェアエミュレーションを行うハイパーバイザー
- ホストとゲストは異なるアーキテクチャ（CPU）であってもよい
- アクセラレータ（**KVM**など）と組み合わせて、ネイティブ並みの高速化を実現する。
- 新しい処理チップの開発に役立つ

QEMUとは、Quick **EMU**latorの略です。2002年にFabrice Bellard氏によって書かれたものである。(Bellard氏は、一時期、 π の計算で2.7兆桁に達する世界記録を保持していたことでも知られている)

QEMUは、ハードウェアエミュレーション、または仮想化を行うハイパーバイザーです。ホストアーキテクチャとエミュレートされたアーキテクチャの間でバイナリ命令を動的に変換することで、CPUをエミュレートします。

ホストとエミュレートされたアーキテクチャは、異なっても同じでもよい。ホストとゲストの両方のオペレーティング・システムには多数の選択肢があります。

また、QEMUは、OS全体ではなく、特定のアプリケーションだけをエミュレートするために使用することができます。

QEMU単体では、ホストマシンに比べてかなり遅い。しかし、**KVM**（Kernel Virtual Machine）と併用することで、ネイティブホストの速度に近い速度に到達することができます。

ゲストOSは、**QEMU**で動作させるために書き換えを必要としません。

QEMUは、いつでも仮想マシンを保存、一時停止、復元することができます。

QEMUは、**GPL**の下でライセンスされたフリーソフトウェアです。

QEMUは、以下のような多くのアーキテクチャをサポートする機能を備えています：

IA-32（i386）、x86-64、MIPS、SPARC、ARM、SH4、PowerPC、CRIS、MicroBlaze、その他。

<https://wiki.qemu.org/Documentation/Platforms> をご参照ください。

QEMUはクロスコンパイルが可能なため、組み込みプロセッサの開発を行う際に非常に便利です。

実際、**QEMU**は、まだ物理的に生産されていない、あるいは市場にリリースされていないプロセッサの開発に使われるこ

とが多いようです。

サードパーティ製ハイパーバイザー

- QEMUは低速だが、アクセラレータを使えばほぼネイティブの速度に達することができる
のようなものです：
 - KVM
 - ゼン
 - オラクル仮想ボックス
 - VMware ESXとVMware Workstation/Player
- KVMはQEMUと非常に緊密に統合されており、同じプロジェクトから生まれました。

QEMUはそれ自体で使用すると、比較的遅いです。しかし、サードパーティのハイパーバイザーと統合することで、ネイティブに近い速度に到達することができます。これらのシステムの中には、QEMUに非常に近いところがあれば、もっと離れたものもあることに注意してください。主なものをいくつか挙げます：

- KVMは特にQEMUと緊密に連携しています。ホストとターゲットが同じアーキテクチャの場合、フルアクセラレーションと高速性を実現します。KVMはLinuxにネイティブです。これについては、詳しく説明します。
- XenもLinuxにネイティブであり、アーキテクチャが提供している場合はハードウェア仮想化モードで実行することができ、x86やいくつかのARMのバリエーション。
- Oracle Virtual Boxはqcow2形式のイメージを使用することができ、qemuと非常に密接な関係があります。

このコースでは、仮想マシンの設定と実行に **virt-manager** を使用することを推奨します（ラボでも使用します）。また、**qemu** コマンドラインユーティリティを使用して仮想マシンを実行する方法についても説明します。

画像フォーマット

- 主な2つのFormat :
 1. ロー最もシンプル
 2. **qcow2**: コピーオンライト2
- その他多くのフォーマットに対応可能
- フォーマット間の変換が可能 :


```
$ qemu-img convert -O vmdk myvm.qcow2 myvm.vmdk
```

QEMUは、ディスクイメージファイルの多くの形式をサポートしています。しかし、主に使用されるのは2つだけで、残りは歴史的な理由と変換ユーティリティのために利用できます。この2つの主なフォーマットは以下の通りです :

- **raw** (デフォルト)
これは最もシンプルで、他の**非QEMU**エミュレータにエクスポートするのが簡単です。空のセクタはスペースを取りません。
- **キュキュウツ**
COWはCopy On Writeの略です。多くのオプションがある。**man qemu-img**を参照してください。

対応フォーマットの一覧を取得する :

```
$ qemu-img --help | grep formats :
```

```
サポートされているフォーマット blkdebug blklogwrites blkreplay blkverify copy-on-read ファイル ftp
ftps gluster
→ host_cdrom host_device http https iscsi iser luks nbd null-aio null-co nvme qcow2 quorum raw rbd ssh
→ スロットル VHDX VMDK VPC
```

特に注意してください :

- **vdi**です : **Oracle Virtual Box**で使います。
- **vmdk**です : **VMware**で使います。

なお、**qemu-img**はフォーマット間の変換に使用することができます :

```
$ qemu-img convert -O vmdk myvm.qcow2 myvm.vmdk
```

をデフォルトのオプションで使うことができます。完全な可能性については、**マニュアルページ**を参照してください。

19.7 KVM

KVMとLinux

- カーネルベース仮想マシン
- ハードウェア仮想化エクステンションが必要
- 共同処理関係でカーネルを資源として利用する
- カーネルがゲストの命令をエミュレート
- アヴィ・キヴィティが投稿し、2007年にすぐさま統合された
- 準仮想化の拡張とサポートがあります。

KVMは、メモリ管理、スケジューリング、同期などのコンピューティングリソースにLinuxカーネルを使用します。仮想マシンを実行する際、**KVM**はLinuxカーネルとコ・プロセッシングの関係を結びます。

この形式では、**KVM**はVMXまたはSVM命令を使用して、1つまたは複数のCPU内で仮想マシンモニターを実行します。同時に、Linuxカーネルは他のCPU上で実行されます。

仮想マシンモニターは、仮想マシンモニターを引き継がせる命令を実行するまで、ハードウェアの全速力で動作しているゲストを実行します。

この時点で、仮想マシンモニターは、ゲストの命令をエミュレートするために任意のLinuxリソースを使用したり、最後の命令でゲストを再起動したり、何か他のことを行うことができます。

KVMモジュールをロードしてゲストを起動することで、Linuxをハイパーバイザーに変身させます。Linuxの人格はそのままですが、ハードウェアの仮想マシン・モニターもあります。**cgroups**、**nice**、**numactl**などの標準的なLinuxリソースおよびプロセス制御ツールを使用して、仮想マシンを制御することができます。

KVMは、Windows Virtual Desktop製品の一部として（マージ前に）初めて登場しました。2007年のアップストリームマージ時、**KVM**は最近のx86 64プロセッサを必要としました。x86 64プラットフォームでは、KVMは主にプロセッサの仮想化サブシステム用のドライバです（ただし、常にそうであるとは限りません）。

KVMは、2007年に3つのLinuxカーネルモジュールとして登場しました。同時に提供されたQEMUの改良版と組み合わせることで、ランタイムサービスのほとんどにLinuxカーネルを使用するハイパーバイザーを実現しました。

KVMの作者であるAvi KivityがLinux開発コミュニティにソースコードを提出した直後、LinusはKVMのソースコードをマージした。

KVMを自分のLinuxツリーに入れた。これは多くの人にとって驚きでした。

KVMの管理

- コマンドラインとグラフィカルインターフェースの両方
- コマンドラインツールは以下の通りです：
 - ビルト・エックス
 - ケームー
- グラフィカルなインターフェースは、以下の通りです：
 - バーチャルマネージャー
 - コックピット
 - キムチ
 - オープンスタック
 - オヴァート

ここでは `virt-manager` に絞って説明します。

仮想マシン・イメージの作成、変換、操作、デプロイ、保守のための低レベルのコマンドが多数用意されています。より専門的な知識を身につければ、それらの使い方に慣れるでしょう。しかし、すべての基本的な操作については `virt-manager` で十分なので、これを使用することにします。

19.8 クラウドイット

クラウドイット

cloud-initというパッケージがあります：

- ブート時にクラウドイメージをカスタマイズするメカニズムを提供します。カスタマイズの内容は以下の通りです：
 - 初回起動時に1回だけカスタマイズ
 - イメージの起動ごとに
- ファイルまたはコマンドラインによる設定が可能
- ブートプロセスの非常に早い段階でカスタマイズを開始するために **systemd** を使用します。
- 多くの場合、cloud-initはcockpitやproxmoxのようなフル機能のグラフィカルな仮想マシン管理ツールによって「隠れて」使用されます。

cloud-initユーティリティは、ブートプロセス中にシステム上のほとんどすべての要素をカスタマイズすることができます。カスタマイズされる一般的な項目としては、ネットワーク構成、ユーザーの追加とカスタマイズ、パッケージの追加と削除があります。これ以外にも多くの要素をカスタマイズすることができます。

cloud-initプロセスは、システムを構成するためにファイルを使用することができます。これらのファイルは、**systemd**サービスによってブート時にマウントされるisoイメージに含まれています。isoイメージには、**systemd**が知っているファイル名とディレクトリ名が含まれています。イメージの作成を支援するユーティリティがあります。

ここでは、**cloud-init**の設定データを格納するために作成したcdromイメージの例を紹介します：

\$ sudo lsblk -f

名称	FSTYPE LABEL	ユーユーアイディー	マウントポイント
ループ0	iso9660 cidata	2022-09-01-10-57-10-00	
sr0	iso9660 cidata	2022-09-01-10-57-10-00	

ラベルに注目、これは**cloud-init**に特有のものです。CD-imageの中身を検証する：

\$ ls -l /mnt

```

計2
-rw-rw-r-- 1 ubuntu ubuntu 33 Sep 1 14:57 meta-data
-rw-rw-r-- 1 ubuntu ubuntu 233 Sep 1 14:57 network-config
-rw-rw-r-- 1 ubuntu ubuntu 658 Sep 1 14:57 user-data

```

cloud-initが実装のために消費する設定ディレクティブのファイルが3つあります。**cloud-init**の設定に関する詳細は、<https://cloudinit.readthedocs.io/en/latest/> にあります。

cloud-init、ブートインテグレーション

cloud-initの使用は、ブートシーケンスに統合されています。ブートシーケンスには、クラウドイニットに関連する5つのステージがあります：

- ジェネレーターです：設定データを読み込み、通常は/rundディレクトリにsystemdサービスプロファイルを作成する**systemd**機能
- ローカルデータソースを探す
- ネットワークネットワークを設定する
- **Config**です：選出されたtextbfconfigモジュールを実行します。
- **最終的に**：通常実行されるスクリプトやカスタマイズを実行します。

• ジェネレーター

- ジェネレーターファイルの場所です : `/lib/systemd/system-generators/cloud-init-generator` です。
- ジェネレータは通常、`/run/cloud-init/`ディレクトリにログを記録します。
- systemdinterface: <https://www.freedesktop.org/software/systemd/man/systemd.generator.html>

• ローカル

ローカルデータソースを探し出し、ネットワーク設定を適用する :

- ローカルデータソースの1つは、VM作成時に「CD-ROM」として使用されたisoイメージです。
- ローカルデータソースを見つけるためのsystemdサービスが存在します: `cloud-init-local.service`
- 古くなったネットワーク構成要素を一掃する

• ネットワーク

ネットワークを構成する :

- 関連するsystemdサービスは、`cloud-init.service`です。
- 関連するcloud-initモジュールは、`/etc/cloud/cloud.cfg`のcloud init modulesです。

• Configは、残りのクラウドモジュールを処理します :

- 関連するsystemdサービス : `cloud-config.service`
- cloud-modulesの詳細はこちら : <https://cloudinit.readthedocs.io/en/latest/topics/modules.html> .

• 最終

スクリプトやパッケージのインストールなど、ユーザーが作成した様々なスクリプトを仕上げる :

- 関連するsystemdサービス : `cloud-final.service`

19.9 研究室

演習19.1 : KVMが正しくセットアップされていることを確認する



非常に重要

- 以下のラボは、**Linux**がネイティブで動作する物理マシンで実行するのが最適です。
- VMWareや**Virtual Box**、あるいは**KVM**などのハイパーバイザーで動作するVirtual Machine内で実行することも可能でしょう。ただし、この場合、**ネストされた仮想化**が適切に実行されていることが必要です。
- これがうまくいくかどうかは、使用するハイパーバイザー、基盤となるホストOS（**Windows**、**Mac OS**、**Linux**など）、およびLinuxや**Windows**のバージョン、特定のカーネルなど、特定のバリエーションに依存します。
- さらに、特定のハードウェアにも依存します。例えば、**VMWare**では様々な**x86 64**マシンでネスト仮想化が動作していますが、**Oracle Virtual Box**では一部のマシンでのみ動作していることが確認されています。
- もしこれがうまくいけば、ネイティブのハードウェアで実行した場合と比べてパフォーマンスが低下しますが、これから行う簡単なデモンストレーションの練習では重要ではありません。
- あなたの走行距離は変化します！もしそれが動作しない場合、私たちはそれを回転させるためにあなたを助ける

1. まず、ハードウェア仮想化が利用可能で有効であることを確認します：

```
$ grep -e vmx -e svm /proc/cpuinfo
```

ここで、vmxは**INTEL** CPU用、svmは**AMD**用です。どちらかが表示されない場合：

- 物理的なマシンであれば、もしかしたら解決できるかもしれません。マシンを再起動し、BIOSの設定で仮想化をオンにできるかどうか確認してください。IT担当者によっては、「セキュリティ」上の理由でこれを不可能にしている場合があるので、そのポリシーを変更してもらうよう努力してみてください。
- ハイパーバイザーで動作する仮想マシン上で、ネスト化された仮想化が操作できない状態です。

2. これらのいずれかの理由で、ハードウェア仮想化がない場合、**virt-manager**を実行できるかもしれませんが、パフォーマンスが弱くなります。
3. 関連するすべてのパッケージがシステムにインストールされている必要があります。正確なリストを作成するために努力することは可能です。しかし、正確な名前や要件は時代とともに変化し、ほとんどのエンタープライズ向けディストリビューションには、必要なソフトウェアのすべて（またはほとんどすべて）が同梱されています。
4. 最も簡単で最良の方法は、弊社が既に提供したスクリプトを実行することです：

```
$ ./ready-for.sh --install LFS307
```

私たちが苦労したところ

あるいは、**RPM**システム上では、オーバーキルも可能です：

```
$ sudo dnf|zypper install kvm* qemu* libvirt* です。
```

この方法でやっても、収納量は多くないです。

Ubuntuを含む**Debian**パッケージベースのシステムでは、好きなパッケージのインストール手順で同等のことを行う必要があります。



非常に重要

- **libvirtd**を他のハイパーバイザーと同時に実行すると、悲惨な結果になる可能性があるため、実行しないでください。これは、システムをクラッシュさせたり、使用している仮想マシンに損害を与えたりする可能性があります。



- のように、他のハイパーバイザーの停止と無効化の両方を推奨します：

```
$ sudo systemctl stop vmware
```

```
$ sudo systemctl disable vmware
```

または

```
$ sudo systemctl stop vboxdrv
```

```
$ sudo systemctl disable vboxdrv
```

演習19.2 仮想マシンのインストールと実行のためにKVMとvirt-managerを使用します。

この演習では、TinyCoreLinux (<http://www.tinycorelinux.net>) によってビルドされたビルド済みisoイメージを使用します。

Debian、CentOS、Ubuntu、Fedora、openSUSEなど、他のLinuxディストリビューションのインストールisoイメージに置き換えることも可能です。基本的な手順は同じで、新しいVMを構築するためのインストール段階が異なるだけで、実際の物理的なマシンに新しいインストールを構築するのと何ら変わりはありません。

画面キャプチャ画像付きで順を追って説明しますので、自信のある方はvirt-managerを起動するだけでも試してみてください。と、GUIがそれなりにわかりやすく構築されているので、必要なステップを踏めるかどうか確認してみてください。

1. libvirtd が起動していることを確認し、virt-manager をタイプして起動します：

```
sudo systemctl start libvirtd
```

```
$ sudo virt-manager
```

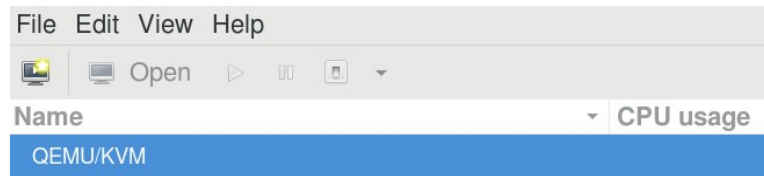


図 19.5: virt-manager の起動

2. ファイル」→「新しいマシンを作成」をクリックします：

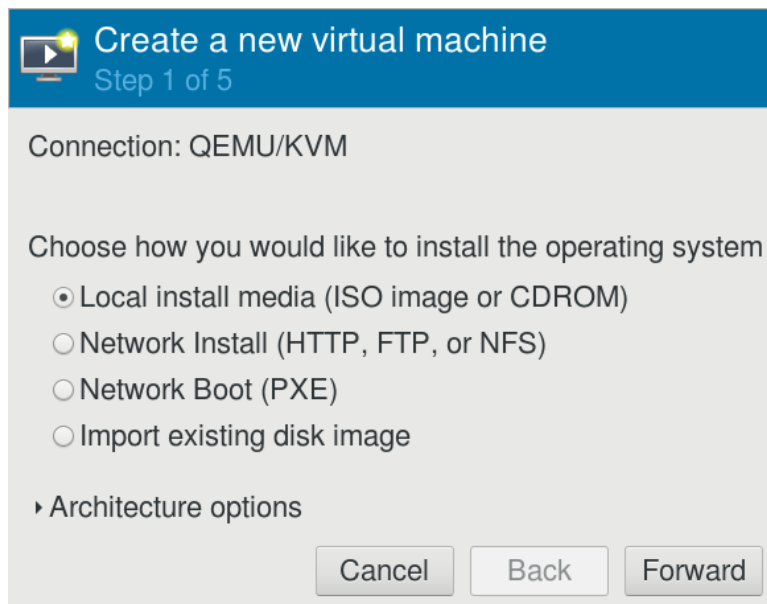


図 19.6 : virt-manager による仮想マシンの作成

3. TinyCoreLinuxの3種類のisoインストールイメージをRESOURCES/s_19ディレクトリに収録しています：

[Core-current.iso](#)

[CorePlus-current.iso](#)

[TinyCore-current.iso](#)

(上流に新しいバージョンがあるかどうかは <http://www.tinycorelinux.net> で確認できますが、これらは問題ないでしょう)。CorePlus-currentが一番大きく頑丈で、フルグラフィックでインストールできるので、これを使うことにします。ただし、他のものはかなり早く使える。

ファイルシステムを操作して、必要な画像を選びます：

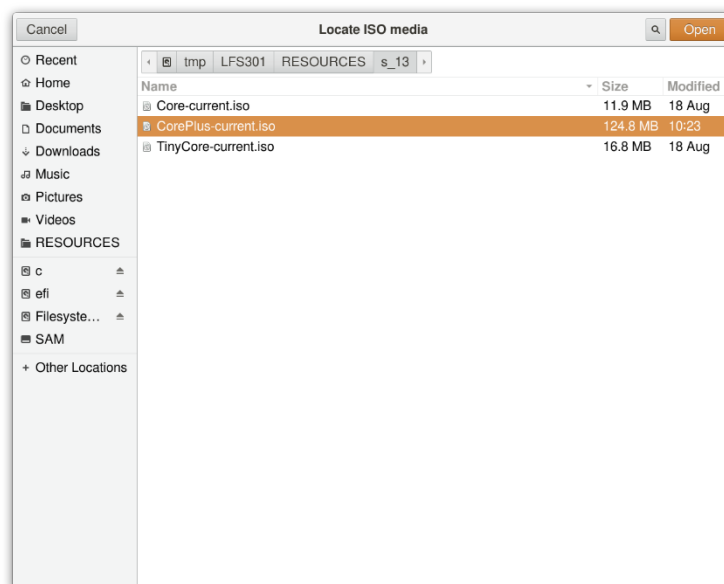


図 19.7: virt-manager での TinyCoreLinux iso イメージの選択

4. 次に、使用するメモリの量とCPUやコアの数をリクエストする必要があります。これらのイメージはかなりミニマムです。256MBを選択すれば十分すぎるほどで、どこまで下げられるか楽しみかもしれません！

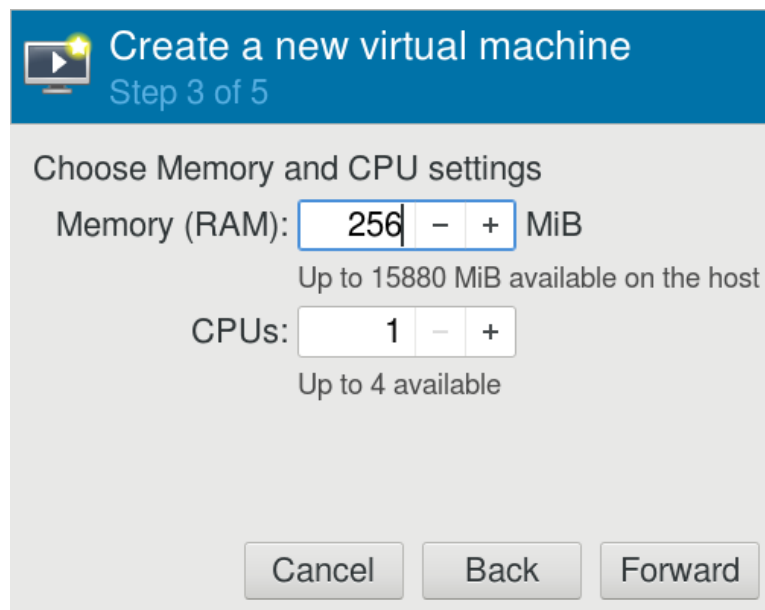


図 19.8: virt-manager でのメモリと CPUS の設定

5. 次に、作成する VM の場所とサイズを設定する必要があります。TinyCoreLinuxでは実際にはほとんど必要ありませんが、GUIでは0.1GB（約100MB）以下は選択できません（コマンドラインからなら簡単に容量を少なく設定できます）。

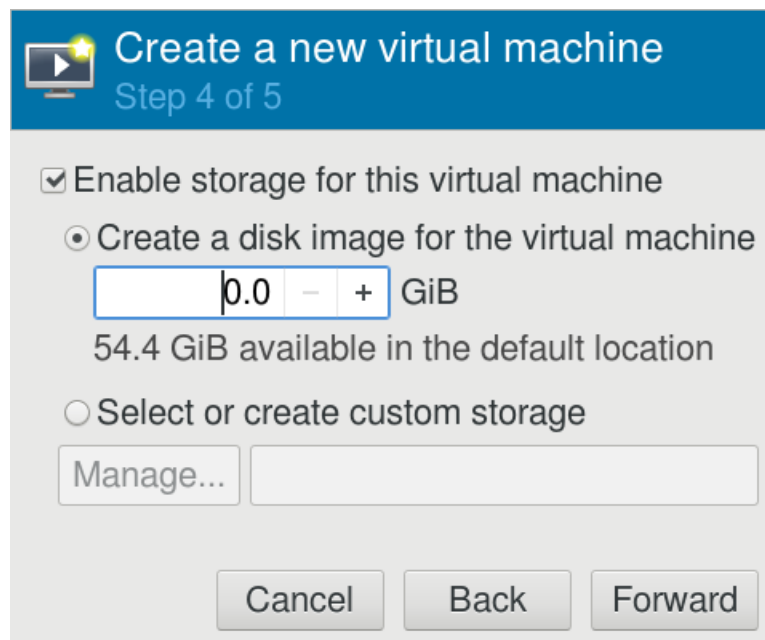


図 19.9: virt-manager でのディスクストレージの設定

選択またはカスタムストレージの作成をクリックしない場合、画像は `/var/lib/libvirt/images` に配置されます。イメージはかなり大きくなる可能性があるため、別の場所に置くように設定することをお勧めします。または、`/var/lib/libvirt` の `images` ディレクトリを、次のように他の場所へのシンボリックリンクで置き換えることができます：

```
$ cd /var/lib/libvirt
$ sudo mv images images_ORIGINAL
sudo mkdir /tmp/images
$ sudo ln -s /tmp/images イメージズ
```

(画像ファイルは/tmpとは別の場所に置きたいところでしょうが、わかりいただけると幸いです)。

6. これで、TinyCoreLinuxのインストールディスクから、自分のVMのインストールを開始する準備ができました：

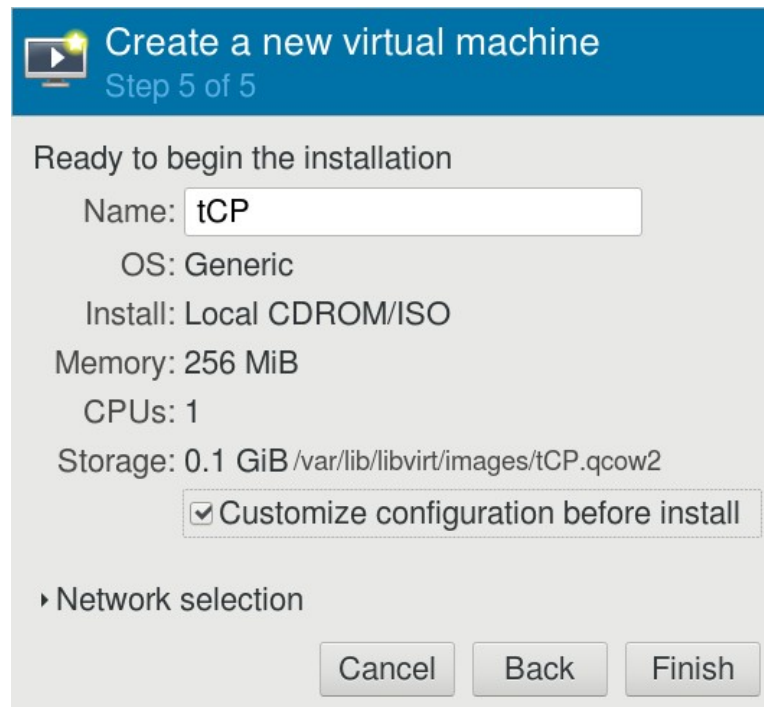


図 19.10 : virt-manager での VM インストールの開始



ご注意ください

インストールする前に、設定の**カスタマイズ**をクリックすることをお勧めします。他の変更も可能ですが、マウスポインタはデフォルトで**PS2**デバイスとして設定されていますので、**USBタブレット**入力ポインタを追加した

次の画面で「ハードウェアの追加」をクリックし、これを実行します：

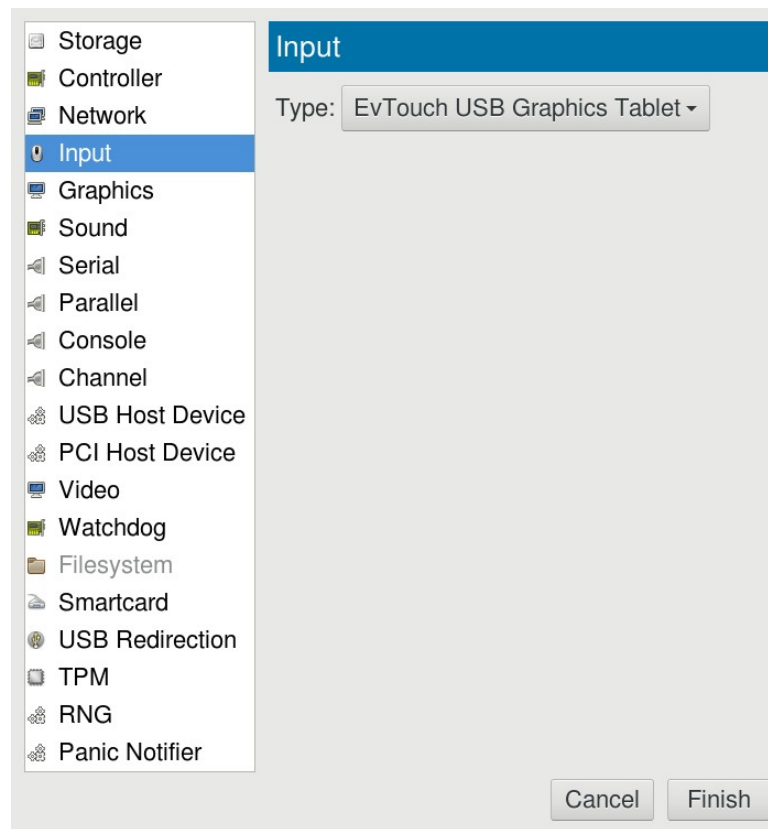


図 19.11: virt-manager で VM に入力デバイスを追加する

7. いよいよインストールを開始します :

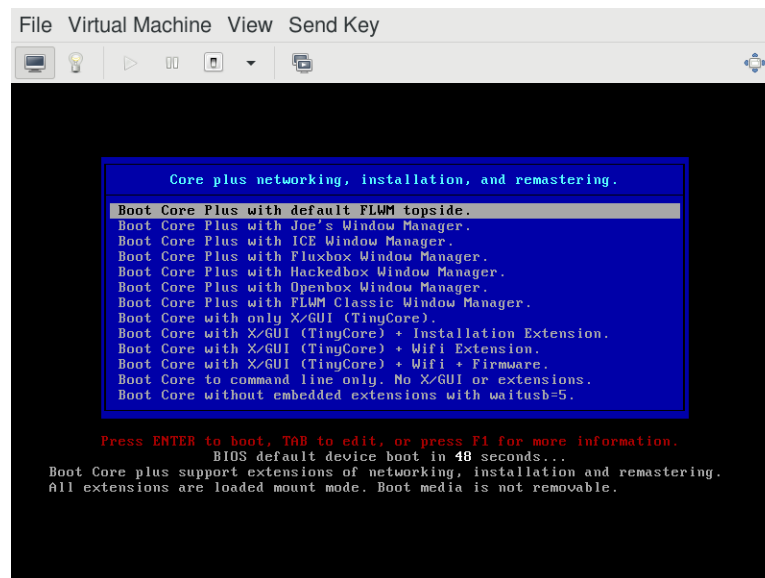


図 19.12: virt-manager を使用したインストールメディアへのブート

グラフィカル・インターフェースは他にも選択できますが、ここでは最初のもの、つまりデフォルトを選んでreturnを押してください。

8. しばらく時間がかかりますが、最終的には以下の画面が表示されます :

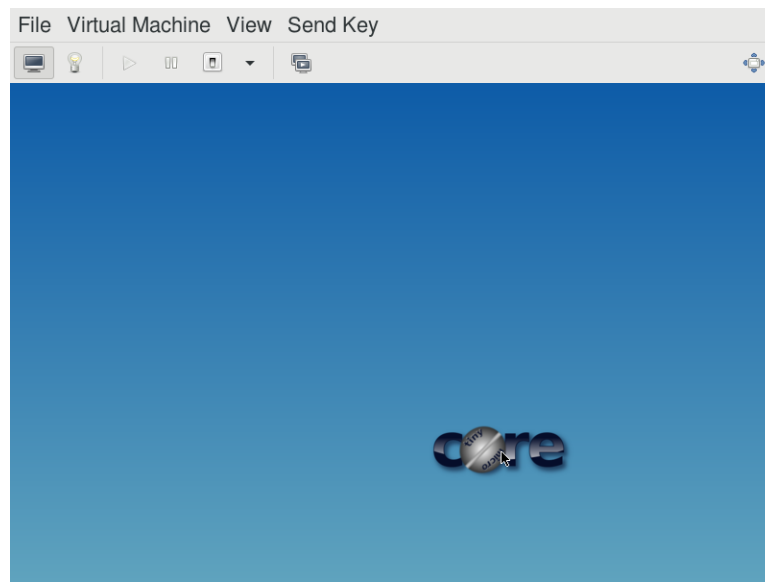


図 19.13 : TinyCoreLinux の最初の画面

ここで何をすればいいのかわからないが、下の方にあるアイコンを見る必要があるので、リサイズして画面を縦長にする必要がある：

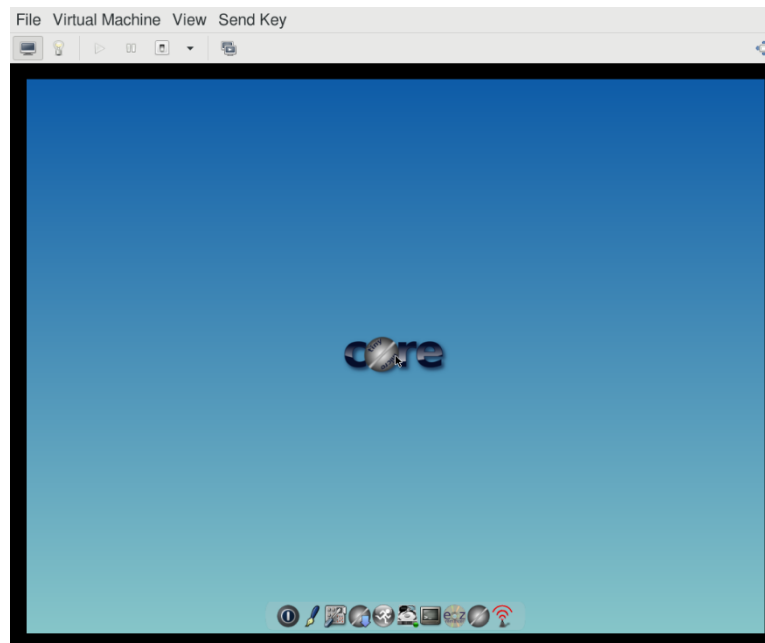


図19.14 : TinyCoreLinuxの最初の画面をリサイズしたもの

- ターミナルアイコンをクリック（または背景上で右クリックし、ターミナルを開いてください。フォントは残念ながらマイクロスコープであることに注意してください。次に次のように入力します。

ティーシーインストール

をウィンドウに表示します。

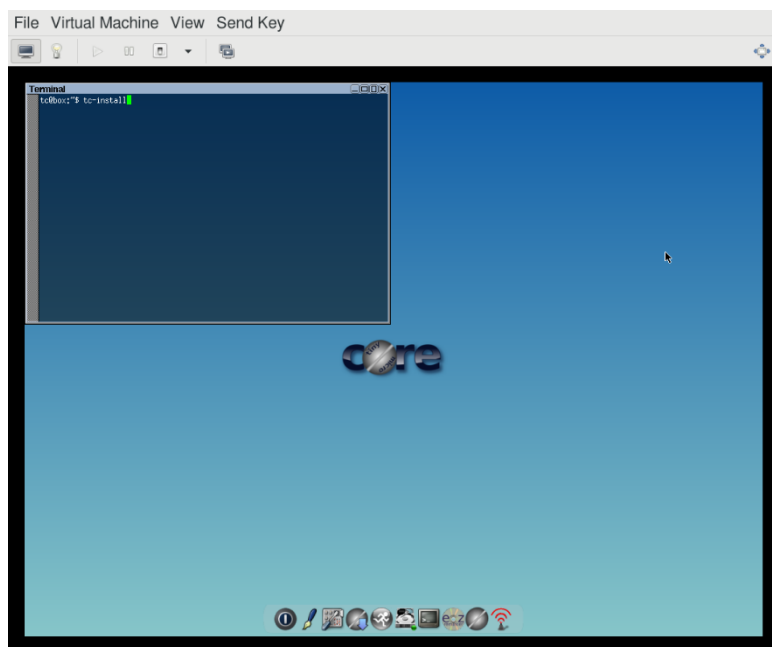


図 19.15 : tc-install の実行

10. ディスク全体とsdaを選択し、前進矢印をクリックします：

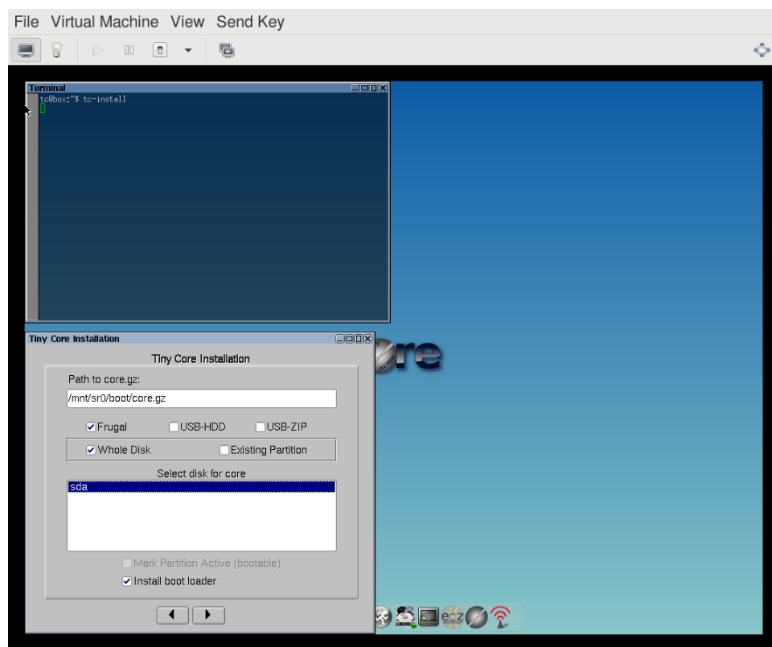


図19.16 : VM内のディスクの選択

11. しばらくはクランクして、各ステップが出カウィンドウに反映されます。

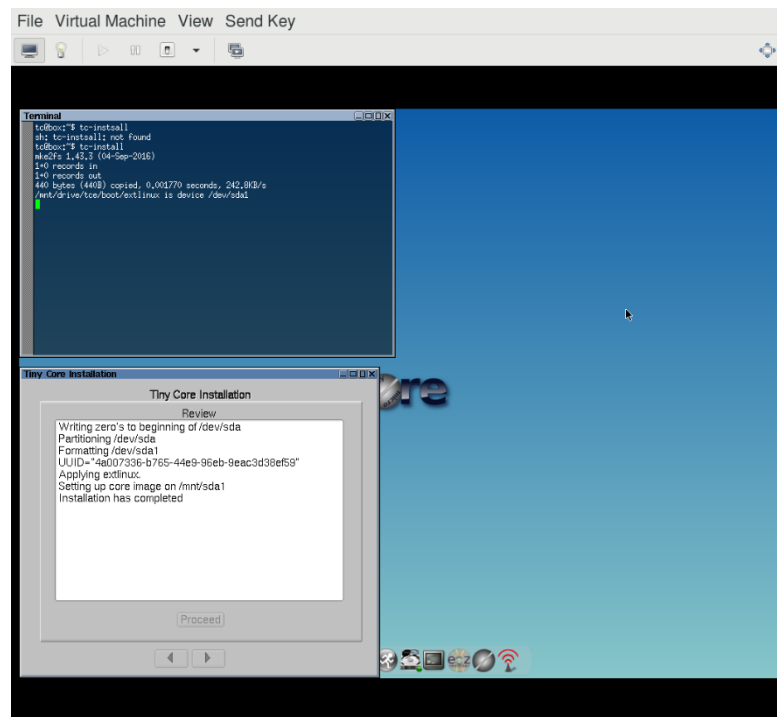


図 19.17: virt-manager でのインストールの終了

インストールが完了したら、「ファイル」メニューから仮想マシンをシャットダウンすることができます。

12. **virt-manager**を再び起動し（もし終了していた場合）、次のようなものが表示されるはずです：

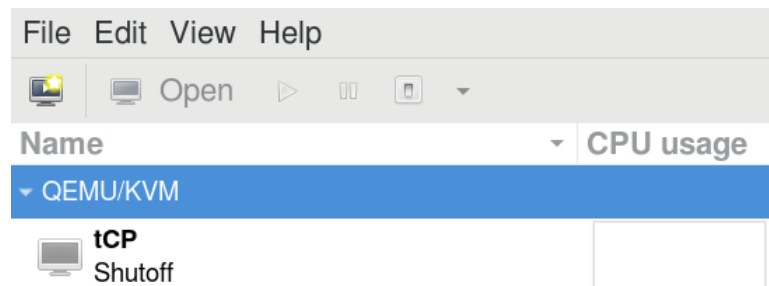


図19.18 : virt-managerで新しいVMを実行する

VMを右クリックし、開いて実行します。新しい仮想マシンが立ち上がっているはずです！(もし、混乱し

オリジナルのインストールイメージを実行している場合は、新しいディスクイメージに**tc-install**プログラムがないことに注意して、それがそうでないことを確認することができます。

練習問題19.3: エクストラクレジット: コマンドラインから実行する

コマンドラインから

```
$ sudo qemu-img create -f qcow2 /var/lib/libvirt/myimg.qcow2 24M
```

```
$ sudo qemu-system-x86_64 -hda /var/lib/libvirt/myimg.qcow2 -cdrom CorePlus-current.iso -usbdevice タブレット
```



ソースからQEMUをビルドする

RHEL/CentOS 8にはqemu-system-*バイナリは付属していません。このため、これまでの演習で使用した**kvm-qemu**ソフトウェアを使用することを想定しています。

これらのディストリビューションや、これらのプログラムが提供されていないシステムでは、常にソースからコンパイルすることができます。最新バージョンを入手するために、ディストリビューションがバイナリを提供していても、さまざまな理由でこれを行いたい場合があります。

なお、これにはビルド段階で約3GBのストレージが必要で、かなりの時間を要します。また、開発用ソフトウェアが不足している可能性があり、動作させるためにいくつかの作業を行う必要があります。(ninja*やlibpulseaudio-develなどです)。

という手順で行います：

1. QEMUのソースコードを取得します：

```
$ git clone https://git.qemu.org/git/qemu.git
$ cd qemu
$ git submodule init
$ git submodule update --recursive
```

2. x86_64 バージョンの qemu-system のみをビルドするように QEMU を設定する; 使用する予定のアーキテクチャのみをビルドする。

```
$ ./configure --target-list=x86_64-softmmu
```

3. QEMUを非rootユーザーでビルドする。

```
$ make -j $(nproc)
```

4. QEMUをrootで/usr/local/bin/にインストールします。

```
sudo make install
```

練習問題19.4cloud-initを使用してVMイメージをカスタマイズする

概要

この演習では、**cloud-init**を使用して、ブート時に新しい仮想マシンイメージをカスタマイズします。

このセクションで使用するコマンドのコピーは、[LFS307/SOLUTIONS/s 19/ディレクトリ](#)のtarballにあります。

1. はじめに、クラウド対応のVMイメージが必要です。**Ubuntu Bionic (18.04)** クラウドイメージをダウンロードします：
`$ wget https://cloud-images.ubuntu.com/bionic/current/bionic-server-cloudimg-amd64.img`
2. テスト用の新しいsshキーを作成し、変数pubkeyに公開キーを設定します：

```
$ ssh-keygen -t ed25519 -b 4096 -f id_ed25519 -C test1 -N "" -q
$ pubkey='cat id_ed25519.pub' とする。
```

3. 作成したsshキーをファイルcloud_init.cfgに追加します。プレースホルダ <sshPUBKEY> を新しく生成されたキーで置き換えます。変更されていないcloud_init.cfg ファイルを調べます。

```
$ cat cloud_init.cfg
```



クラウドinit.cfg

```
#cloud-config ホスト名
: test1
fqdn: test1.example.com
manage_etc_hosts: true
users :
  - 名前 : ubuntu
    sudo: ALL=(ALL) NOPASSWD:ALL です。
    groups: users, admin
    home : ホーム :
      /home/ubuntu shell :
        /bin/bash
    lock_passwd: false
    ssh-authorized-keys
      :
        - <sshPUBKEY>になります。
ssh_pwauth: false # sshによる認証のみ (コンソールアクセスは
そのままログインできます)
disable_root: false
chpasswd :
  をリストアップします : |
    ubuntu:linuxの期
    限切れ : False

package_update: true パ
ッケージ :
  - qemu-guest-agent
# /var/log/cloud-init-output.log に書き込むメッセージ
first_message: "シフトは45秒後にシステムをリブートし、"
```

4. cloud_init.cfgに公開鍵の内容を追加して修正します :

```
$ sed "s%- <sshPUBKEY>%- $pubkey%" cloud_init.cfg
```



クラウドinit.cfg

```
#cloud-config ホスト名
: test1
fqdn: test1.example.com
manage_etc_hosts: true
users :
  - 名前 : ubuntu
    sudo: ALL=(ALL) NOPASSWD:ALL です。
    groups: users, admin
    home : ホーム :
      /home/ubuntu shell :
        /bin/bash
    lock_passwd: false
    ssh-authorized-keys
      :
        - ssh-ed25519 AAAAC3NzaC1lZD41NTF5AAAAAAGQ8IHNhbnRpdjE4LW40YXZpZD41NTF5AAAA test1@test1
```



```
ssh_pwauth: false
disable_root: false
chpasswd :
  をリストアップします : |
    ubuntu:linuxの期
    限切れ : False

package_update: true /パ
ッケージ :
  - qemu-guest-agent
# /var/log/cloud-init-output.log に書き込むメッセージ
final_message : "システムは$UPTIME秒後によく立ち上がり
ました。"
```



トラブルシューティング

ssh-authorized-keysの扱いには十分注意してください。sedコマンドは失敗する可能性があり、その場合は手動で `cloud_init.cfg`。

5. 設定ファイルを入力として、isoイメージファイルをビルドします :

```
$ cloud-localds -v --network-config=network_config_static.cfg test1-seed.qcow2 cloud_init.cfg
```

6. 仮想マシンを起動します :

```
$ sudo virt-install --name test1 ㊦。
  --virt-type kvm \
  --メモリ 2048 --vcpus 2 ㊦。
  --ブートHDD,メニュー=オン □。
  --disk path=test1-seed.qcow2,device=cdrom \
  --disk path=snapshot-bionic-server-cloudimg.qcow2,device=disk ㊦。
  --graphics none ㊦。
  --os-variant ubuntu18.04 木号
  --network network:default \
  --コンソール pty,target_type=serial
```

システムが起動すると多くの出力がありますが、最後には次のような表示になるはずです :

```
[ 9.309226] cloud-init[1006]です : Cloud-init v. 22.2-0ubuntu1 ~18.04.3.
  は、Tue, 30 Aug 2022 18:29:07 +0000 で 'modules:final' を実行しています。
9.16秒アップしました。[ 9.311630] cloud-init[1006]です : 9.30秒後によくシス
テムが立ち上がりました
[OK] クラウドユーザー/ファイナルスクリプ
トの実行を開始しました。[ OK] ターゲットに
到達 クラウドイニットターゲットです。

Ubuntu 18.04.6 LTS test1 ttyS0
test1 login:
```

7. クリーンアップする。

以下は、仮想マシンのクリーンアップを行います。VMの現在のステータスを確認します :

```
$ sudo virsh list --all
```

アイド 名	状態
称	ランニ

tst1 VMを停止します。

```
$ sudo virsh destroy test1
```

ドメイン「test1」破棄

VMの定義を削除します：

```
$ sudo virsh undefine test1
```

ドメイン「test1」が未定義になりました。

第20章

コンテナの概要



20.1	コンテナ	314
20.2	アプリケーションの仮想化	315
20.3	コンテナと仮想マシンの比較	316
20.4	ドッカー	317
20.5	Dockerコマンド	319
20.6	ポッドマン	320
20.7	研究室	323

学習目標

このセッションが終わるころには、次のことができるようになっているはずです：

- コンテナを定義する基本的なパラメータとメソッドを理解する。
- アプリケーションを仮想化することの意味を知る。
- コンテナと仮想マシンの違いを理解する。
- **Docker**に精通し、適切に使用するために必要な手順を知っていること。
- Podmanが**Docker**をどのように置き換えてきたかを知る。
- **Docker**、**Podman**ともに、コンテナに関連する主要なコマンドを使用します。

20.1 コンテナ

コンテナの基礎知識

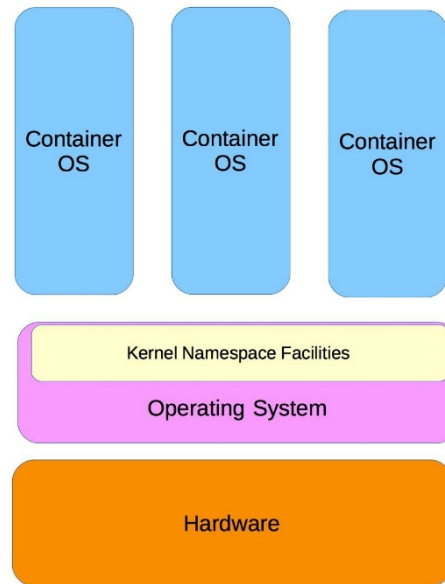


図 20.1 : コンテナ

ハイパーバイザーと Linux カーネルの統合がさらに進み、オペレーティングシステムレベルの仮想マシン（コンテナ）の作成が可能になりました。コンテナはLinuxカーネルの多くの機能を共有し、名前空間やcgroupsといった最近のカーネルに追加された機能を利用することができます。コンテナは非常に軽量で、仮想マシン全体を持つことに伴うオーバーヘッドを軽減します。

コンテナの最初のフレーバーは、OSコンテナでした。このタイプのコンテナは、initプロセスを実行し、複数のアプリケーションを起動する機能を備えたオペレーティングシステムのイメージを実行します。

LXC（Linux Containers）はその一例です。

20.2 アプリケーションの仮想化

アプリケーションの仮想化

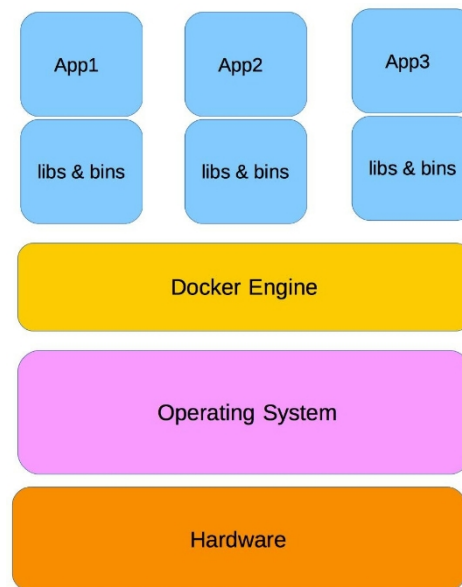


図20.2 : アプリケーションの仮想化

仮想マシンに関連するオーバーヘッドをさらに削減するために、**アプリケーション仮想化**の人気が高まっています。アプリケーション仮想化では、各コンテナに対して1つのアプリケーションを実行します。多くの単一のアプリケーション・コンテナは、通常、1台のマシンで初期化されます。より小さなコンポーネントを使用することで、より大きな柔軟性が生まれ、通常仮想化に関連するオーバーヘッドを削減することができます。

Dockerはそんなプロジェクトのひとつです。

20.3 コンテナと仮想マシンの比較

コンテナと仮想マシンの比較

- プロセスおよびサービスの実行
 - VMは完全なオペレーティングシステムを実行し、多くのサービスやアプリケーションを実行することができる
 - コンテナは通常、1つのものを実行します
- VMはコンテナより多くのリソースを使用する
- 複数のコンテナで1つのOSカーネルを共有し、各VMはそれぞれ独立した
- 容器は持ち運びに便利
- コンテナはVMの内部で実行することができる
- コンテナのセキュリティは難しい
- 通常、コンテナの方が起動が速い
- しかし、仮想マシンは依然として最適なソリューションであることが多い。

仮想マシンとコンテナは、どちらも重要なニーズを満たしています。しばらくの間、それぞれの設備は日の目を浴び、ほとんどすべてのものに対応する方法と見なされてきました。

どちらも長い歴史があります：

- メインフレームコンピュータには、何十年も前からソフトウェアパーティショニングや仮想マシンが、かなり特殊な形で存在していました。
- オペレーティングシステムには、長年にわたってchrootやBSD Jailの実装があり、コンテナと基本的な分離の動機を共有しています。

多くの異なるサービスやアプリケーションを緊密に統合する必要がある場合、サービスとして機能する仮想マシンが最適なソリューションとなる場合があります。

実行されるアプリケーションが、幅広いサービスやその他のライブラリ、アプリケーションを備えた完全なオペレーティングシステム環境を想定して書かれたものであれば、仮想マシンが最適である可能性があります。

ワークロードのスケーリングは、コンテナと仮想マシンでは異なる。KubernetesやMesosのようなオーケストレーションシステムは、必要なコンテナの数を適切に決定し、ロードバランシングを行い、必要に応じてイメージを複製し、削除することなどができます。

20.4 ドッカー

ドッカー

Dockerは、ターゲットアプリケーションをサポートするために必要なサービスを構築するために、多くの個々のイメージを使用してアプリケーションレベルの仮想化を行います。これらのイメージはコンテナにパッケージ化されます。

- 画像はコンテナ内のコンポーネント
- 画像には、以下のものが含まれる場合があります。
 - アプリケーションコード
 - ランタイムライブラリ
 - システムツール
 - ..その他、アプリケーションに必要なものなら何でもOKです。
- イメージは**Docker Hub**やレジストリサーバーに存在することもある

Dockerには豊富なドキュメントがあり、<https://docs.docker.com>

Dockerのウェブサイト<https://www.docker.com>、ドキュメント、チュートリアル、トレーニング情報があります。

Dockerの最も魅力的な特徴の1つは、アプリケーションを依存するすべてのコードとサービスとともにパッケージ化し、最小限のオーバーヘッドで1つのユニットとしてデプロイできることです。このデプロイは、何度でも簡単に繰り返すことができます。このため、エンドアプリケーションをサポートするために、階層化されたサービスを備えたサーバーを構築する必要がなくなります。

Dockerのステップ

Dockerコンテナ化されたアプリケーションの立ち上げは、数ステップしかない場合もあります。

- お好きなツールで**Docker**サービスパッケージをインストールします。
- **Docker**サービスを起動します。
- **Docker Hub**またはプライベートリポジトリから適切なイメージを検索します。
- 画像を引きます。
- イメージを実行する
- で、最後にアプリケーションをテストします。

上記で詳述した手順は、**Docker**アプリケーションをテストするためのごく最小限の例に過ぎません。

もちろん、イメージを作成し、システム変数や設定パラメータを設定し、その結果を新しいイメージとして保存する関数など、さまざまなオプションを使用することができます。場合によっては、書き込み不可能なイメージではなく、書き込み可能なイメージが必要なこともあります。

Dockerコマンドのほとんどは、個別のマニュアルページがあります。例えば、`docker(1)`, `docker-search(1)`, `docker-pull(1)`, `docker-create(1)`, `docker-run(1)` などです。

20.5 Dockerコマンド

dockerコマンド

- **docker**コマンドには40以上のサブコマンドがある
 - 50種類以上のオプションがあるもの
 - `docker <コマンド> --help`で詳細を確認できます。
- | | | |
|-------|-------|------|
| • 走る | • ネット | • 立ち |
| • 生み | ワーク | 上がり |
| 出す | • アール | り |
| • エグゼ | エム | • 立ち |
| ック | • ルミ | 止ま |
| • ps | | |
| • イメー | | |

dockerのサブコマンドはたくさんあり、それらはやや自己文書化される傾向があります。よく混同されるのは、`run`、`create` および `exec` を使用します。`ps` コマンドは、実行中のコンテナ、または `-all` オプションを含めるとすべてのコンテナを一覧表示します。

`run` は新しいコンテナを起動し、その中でコマンドを実行します。一般的なオプションは、`tty` にアタッチするための `-t` と、コンテナをバックグラウンドで実行するための `-d` です。

`create` コマンドは、コンテナを作成するコマンドです。コンテナの設定やアタッチメントを構成するための多くのオプションがあります。

コンテナがすでに起動しており、その中で何かを実行したい場合は、`exec` コマンドを使用することができます。また、`-t -d` オプションも使用できます。

`images` コマンドは、様々な出力で画像を表示します。`rmi` コマンドは、デフォルトで画像を削除し、タグ付けされていない親を削除します。

また、シェル関数を利用して、すべてのコンテナに対して操作を行うことも可能です。例えば、停止しているコンテナをすべて削除する場合：

```
docker rm $(docker ps -a -q).
```

20.6 ポッドマン

ポッドマン

- RHEL8/CentOS8では純粋なDockerからPodmanに置き換わっている
 - Podmanはコンテナの作成と管理に子/親のフォークモデルを使用します。
 - Dockerはサーバー/クライアントモデルを採用し、管理はバックグラウンドで動作するデーモンが担当する
 - 約束された利益：
 - セキュリティの向上
 - オーバーヘッドを除く
- `$ sudo dnf install podman podman-docker`
- エミュレーション層により、ドッカーコマンドとの後方互換性を実現。

ここでは、RHEL/CentOS 8上でdockerエミュレーション層を使用してラボ演習を実行する方法を紹介します。他のディストリビューションでは、パッケージングシステムにPodmanを追加しています。

Ubuntu 20.04では、公式リポジトリにすでにPodmanが含まれているので、そのまま実行することができます：

```
$ sudo apt update
sudo apt install podman
```


ポッドマン、ルートフル vs ルート

ほとんどのコンテナ管理ユーティリティは、非 root ユーザーがコンテナを実行するという概念を持っています。Podmanは非常に明確な違いがあります。

- 根強い
 - **sudo** を使って rootful にする
 - rootfulがすべてのリソースを表示しない
 - は、コンテナとホストシステムとの間にネットワークブリッジをデフォルトで持っています。
 - 設定に `/etc/containers` ディレクトリを使用します。
- 根無し草
 - ユーザー関連リソース（イメージ、ネットワーク、コンテナ）のみを見ることができます。
 - は、ホストシステムとのブリッジがなく、孤立したプライベートネットワークを持っています。

設定ファイルの優先順位は、以下の通りです：

1. ディストリビューションのデフォルトです：
`/usr/share/containers/containers.conf`
2. 管理者用オーバーライドファイルです：
`/etc/containers/containers.conf`
(このファイルが存在しない場合もあります。)
3. ユーザーの好みに合わせて
`HOME/.config/containers/containers.conf`

ネットワークとユーザーレベルのアクセスは、以下の通りです：

- **slirp4netns**をPodmanネットワークインターフェースで利用する。
- **podman v4**の時点で、**netavark**パッケージはルートレスおよびルートフルコンテナに使用することができます。

Podman networkingの詳細については、以下を参照してください：

https://github.com/containers/podman/blob/main/docs/tutorials/basic_networking.md

Podman、リポジトリ

コンテナイメージのリポジトリが存在する場所は、いくつかあります：

- **/etc/containers/registries.conf**
 - システムワイドコンフィギュレーションファイルの場所
 - に別々の **.conf** ファイルを持つこともできます。
/etc/containers/registries.conf.d
(フォーマットV2のみ)
- **HOME/.config/containers/registries.conf**が存在する場合は、そちらが先に使用されます。競合する場合は、最初に一致したものが使用されます。

リポジトリ情報のオプションで、よくある項目があります：

- **registries.search**は、**Podman**検索コマンドに使用されるリポジトリを示します。検索リポジトリが定義されていない場合、検索は何も情報を返しません。
- **registries.insecure** (デフォルト) レジストリから画像を取得するには**TLS**が必要ですが、このオプションは暗号化されていないHTTPトラフィックを許可します。
- **registries.block**コンテナイメージは、リストされた場所から引き出すことができません。
- は、**repositories**ファイルの追加パラメータです：
unqualified-search-registries オプションは、例えば **pod- man** 検索コマンドに使用する画像レジストリをカンマで区切ったリストで指定できるようにします：

unqualified-search-registries = ["docker.io"] です。

詳細は **man 5 containers-registries.conf** を参照してください。

20.7 研究室

練習20.1 : Podmanコンテナのインストール、テスト、実行



概要

この演習は、**Podman**パッケージが利用可能なディストリビューションを対象としています。

Podmanのパッケージがないディストリビューションでは、代わりに**Docker**を使用します（次の演習に進んでください）

表20.1 : ポッドマンと配布状況

流通	ステータス	改善策
CentOS-8-Stream	一部のリポジトリが誤動作する	リポジトリを docker.io に削減する。
CentOS-9-Stream	作品紹介	とも
Ubuntu-20.04	パッケージは利用できません	Docker を使う
Ubuntu-22.04	がない場合があります。 会議資料	repositories.conf のリポジトリレコードを更新する。

1. **Podman**がインストールされていることを確認する：お使いのディストリビューションに適したコマンドを選んでください：

```
$ sudo dnf install podman podman-docker slirp4netns      # RHEL/CentOS, Fedoraの場合。
$ sudo apt install podman podman-docker slirp4netns      # Ubuntu-22.04、Debian
sudo zypper install podman podman-docker slirp4netns     # openSUSE
```

2. **Podman**ソケットを起動します：

```
sudo systemctl start podman.socket
```

で正常に動作していることを確認するとよいでしょう。

```
$ sudo systemctl status podman.socket?
```

```
student@centos9stream:~
[student@centos9stream ~]$ sudo systemctl status podman.socket
● podman.socket - Podman API Socket
   Loaded: loaded (/usr/lib/systemd/system/podman.socket; disabled; vendor preset: disabled)
   Active: active (listening) since Sun 2022-08-21 12:52:57 CDT; 1 day 9h ago
     Until: Sun 2022-08-21 12:52:57 CDT; 1 day 9h ago
    Triggers: ● podman.service
              Docs: man:podman-system-service(1)
               Listen: /run/podman/podman.sock (Stream)
               CGroup: /system.slice/podman.socket

Aug 21 12:52:57 centos9stream systemd[1]: Listening on Podman API Socket.
[student@centos9stream ~]$
```

図20.3 : Podmanソケットの状態を確認する

もし失敗を示すものがあれば、**/var/log/messages**や、システムにあるその他のログファイルを調べて、手がかりを得る必要があります。



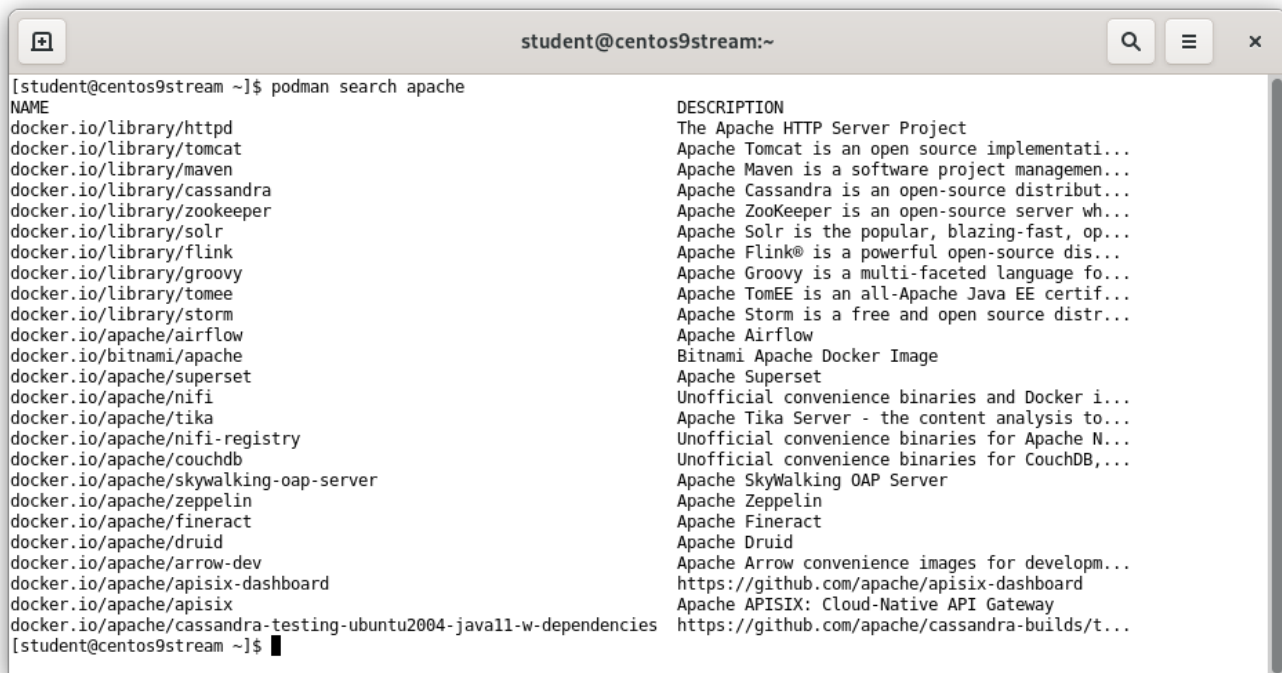
カーネルバージョンが問題になることがある

標準的なディストリビューションカーネルを実行している場合は、問題ありません。しかし、カスタム **Linux** カーネルを実行している場合は、特にネットワークに関して、適切な設定オプションを選択する必要がある可能性があります。この点については、ここで説明するには複雑すぎるので、より難しい練習をしたいのであれば、ディストリビューションから提

3. コンテナレジストリは、実行可能なコンテナのコレクションである。これらは、パブリック、プライベート、またはローカルのコレクションである可能性があります。ディストリビューションによっては、いくつかのレジストリが定義された状態で出荷されるものもあれば、何も定義されていないものもあります。

で、**httpd**のイメージコンテナを検索します：

```
sudo podman search apache
```



```
student@centos9stream:~$ podman search apache
NAME                                DESCRIPTION
docker.io/library/httpd             The Apache HTTP Server Project
docker.io/library/tomcat            Apache Tomcat is an open source implementati...
docker.io/library/maven             Apache Maven is a software project managemen...
docker.io/library/cassandra         Apache Cassandra is an open-source distribut...
docker.io/library/zookeeper         Apache ZooKeeper is an open-source server wh...
docker.io/library/solr              Apache Solr is the popular, blazing-fast, op...
docker.io/library/flink             Apache Flink® is a powerful open-source dis...
docker.io/library/groovy            Apache Groovy is a multi-faceted language fo...
docker.io/library/tomee             Apache TomEE is an all-Apache Java EE certif...
docker.io/library/storm             Apache Storm is a free and open source distr...
docker.io/apache/airflow            Apache Airflow
docker.io/bitnami/apache            Bitnami Apache Docker Image
docker.io/apache/superset           Apache Superset
docker.io/apache/nifi               Unofficial convenience binaries and Docker i...
docker.io/apache/tika               Apache Tika Server - the content analysis to...
docker.io/apache/nifi-registry      Unofficial convenience binaries for Apache N...
docker.io/apache/couchdb            Unofficial convenience binaries for CouchDB,...
docker.io/apache/skywalking-oap-server Apache SkyWalking OAP Server
docker.io/apache/zeppelin           Apache Zeppelin
docker.io/apache/fineract           Apache Fineract
docker.io/apache/druid              Apache Druid
docker.io/apache/arrow-dev          Apache Arrow convenience images for developm...
docker.io/apache/apisix-dashboard  https://github.com/apache/apisix-dashboard
docker.io/apache/apisix             Apache APISIX: Cloud-Native API Gateway
docker.io/apache/cassandra-testing-ubuntu2004-java11-w-dependencies https://github.com/apache/cassandra-builds/t...
```

図20.4 : ポッドマンサーチの使用

(上記のコマンドで**apache**の代わりに**httpd**を使っても、非常に似た結果になりました)。



トラブルシューティング

httpdまたは**apache**を含むコンテナイメージのリストが期待されます。もしそうでない場合は、ディストリビューションがリポジトリ情報でデフォルトの検索エントリを定義していない可能性があります。これは、以下のコマンドを入力し、検索コマンドを再テストすることで、簡単に修正することができます：

```
$ sudo sh -c 'echo unqualified-search-registries = ["docker.io"] >> /etc/containers/registries.conf'.
```

ここまでくれば、もう大丈夫でしょうから、これからは詳細な出力は表示しません。

4. コンテナを回収する：

```
$ sudo podman pull docker.io/httpd
```

すべてのコンポーネントがダウンロードされるまで、数分かかる場合があります。

5. インストールされているコンテナを一覧表示します：

```
sudo podman images
```

6. 画像に関連するコンポーネントをリストアップします。

```
$ sudo podman images --all
```

7. **httpd podman**コンテナを起動します。**httpd**デーモンに接続されたので、ターミナルがハングアップするように見えます。

8. 上記の出力にあるIPアドレスを指すグラフィカルなウェブブラウザを開くことができます。(上記の出力に示されたアドレスを使用しないでください！)
- または、新しいターミナルウィンドウを開き（**httpd**コンテナが実行されているウィンドウを殺さないでください！）、次のコマンドのいずれかを実行することにより、テキストベースのブラウザを使用することができます（特にグラフィカル環境でない場合）：

```
$ lynx http://10.88.0.33
$ w3m http://10.88.0.33
$ elinks http://10.88.0.33
```

お使いのシステムにインストールされているテキストベースのブラウザを使用してください。

9. コンテナとPodmanコンテナを停止し、クリーンアップします。

sudo podman コンテナ リスト

コンテナIDイメージ → NAMES	コマンド	CREATED	STATUS	ポーツ
5eb53be5d5a0	docker.io/library/httpd:latest	httpd-foreground	12分前 Up 12分前	
→ nice_khoranaさん				

\$ sudo podman container kill nice_khorana

nice_khoranaさん

10. これにより、イメージとそれに関連するストレージが **/var/lib/docker** または **/var/lib/containers** のいずれかに残されます（システムおよびディストリビューションによって異なる）。もし再利用する必要がなければ、これでクリーンアップできます：

sudo podman コンテナ prune

警告実行されていないコンテナはすべて削除されます。
。本当に続行しますか？ [y/N] y

```
00b785df8f14fa6ff679e759329f327de01d8d0efce645f7d708ef73c168d0f5
23ad214fbba9e2b1387c006f364a7e473af7c93f4d36940d001a237ea7e8309d
27526c4912abdb22e56813d26f19554f9cd5ae3123a789a534c96253496652f2
27d627aff076f871683cb92e6f89db6efbf47cc150fb5e71f9349c41b03a9ea5
35e7b44229d056073cb39a718a6c4cc624fc6b5e69c861908c15d9881ff45803
3e885f9f02a861adf49858516dff15c4432fab96afaf0cc40d62d66832857225
47c43e3eb2eff9092c82e806f64ce3b1bf720a922715d97a4ba5fa1c0d18de7e
```

sudo podman イメージリスト

レポトリリー	TAG	画像ID	CREATED	SIZE
docker.io/library/httpd	最新版	f2a976f932ec	3週間前	149 MB
quay.io/libpod/banner	最新	5ba9aec95f0f	2ヶ月前	12.1 MB

sudo podman image rm docker.io/library/httpd

タグなし : docker.io/library/httpd:latest
Deleted: f2a976f932ec6fe48978c1cdde2c8217a497b1f080c80e49049e02757302cf74



演習20.2: Dockerコンテナのインストール、テスト、実行



概要

この演習は、**docker**パッケージが利用可能なディストリビューションが対象です。

表20.2 : Dockerと配布状況

流通	ステータス	改善策
CentOS-8-Stream	ディストリビューションに対応したバックがない 百代	Podmanを使用する
CentOS-9-Stream	ディストリビューションに対応したバックがない 百代	Podman-dockerを使う
Ubuntu-20.04	Dockerパッケージが利用可能	Docker を使う
Ubuntu-22.04	Dockerパッケージは利用可能ですが、リポジトリが不足している可能性があります。 会議資料	repositories.confのリポジトリレコードを更新する。

1. dockerがインストールされていることを確認します。あなたのディストリビューションに適したコマンドを選んでください：

```
$ sudo apt install docker.io          # Ubuntu-20.04
sudo zypper install                    docker# openSUSE
```

2. **docker** サービスを起動します：

```
sudo systemctl start docker
```

で正常に動作していることを確認するとよいでしょう。

```
sudo systemctl status docker
```

```
student@ubuntu: ~
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2022-08-26 08:36:50 PDT; 20h ago
 TriggeredBy: ● docker.socket
    Docs: https://docs.docker.com
   Main PID: 1501 (dockerd)
     Tasks: 12
    Memory: 69.4M
    CGroup: /system.slice/docker.service
            └─1501 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Aug 26 08:36:50 ubuntu dockerd[1501]: time="2022-08-26T08:36:50.226589734-07:00" level=warning msg=>
Aug 26 08:36:50 ubuntu dockerd[1501]: time="2022-08-26T08:36:50.226599527-07:00" level=warning msg=>
Aug 26 08:36:50 ubuntu dockerd[1501]: time="2022-08-26T08:36:50.227324626-07:00" level=info msg="Lo
Aug 26 08:36:50 ubuntu dockerd[1501]: time="2022-08-26T08:36:50.384952729-07:00" level=info msg="De
Aug 26 08:36:50 ubuntu dockerd[1501]: time="2022-08-26T08:36:50.441934895-07:00" level=info msg="Lo
Aug 26 08:36:50 ubuntu dockerd[1501]: time="2022-08-26T08:36:50.496674282-07:00" level=info msg="Do
Aug 26 08:36:50 ubuntu dockerd[1501]: time="2022-08-26T08:36:50.497183754-07:00" level=info msg="Da
Aug 26 08:36:50 ubuntu systemd[1]: Started Docker Application Container Engine.
Aug 26 08:36:50 ubuntu dockerd[1501]: time="2022-08-26T08:36:50.518491590-07:00" level=info msg="AP
Aug 26 08:43:37 ubuntu dockerd[1501]: time="2022-08-26T08:43:37.590234369-07:00" level=info msg="ig
lines 1-21/21 (END)
```

図 20.5 : Docker サービスの状態を確認する



トラブルシューティング

もし失敗を示すものがあれば、`/var/log/messages`や、システムにあるその他のログファイルを調べて、手がかりを得る必要があります。



i

カーネルバージョンが問題になることがある

標準的なディストリビューションカーネルを実行している場合は、問題ありません。しかし、カスタム **Linux カーネルを実行している場合は**、特にネットワークに関して、適切な設定オプションを選択する必要がある可能性があります。この点については、ここで説明するには複雑すぎるので、より難しい練習をしたいのであれば、ディストリビューションから提供されたカーネルを使用するようにしてください！

3. コンテナレジストリは、実行可能なコンテナのコレクションである。これらは、パブリック、プライベート、またはローカルのコレクションである可能性があります。ディストリビューションによっては、いくつかのレジストリが定義された状態で出荷されるものもあれば、何も定義されていないものもあります。

で、**httpd**のイメージコンテナを検索します：

```
$ sudo docker search apache
```

```

root@ubuntu: ~
student@ubuntu:~$ sudo -i
root@ubuntu:~# docker search apache
NAME                DESCRIPTION                STARS     OFFICIAL   AUTOMATED
httpd               The Apache HTTP Server Project  4132     [OK]
tomcat              Apache Tomcat is an open source implementati...  3382     [OK]
cassandra           Apache Cassandra is an open-source distribut...  1404     [OK]
maven               Apache Maven is a software project managemen...  1391     [OK]
zookeeper           Apache ZooKeeper is an open-source server wh...  1266     [OK]
solr                 Apache Solr is the popular, blazing-fast, op...  921      [OK]
apache/airflow      Apache Airflow                383
flink                Apache Flink® is a powerful open-source dist...  346      [OK]
apache/nifi          Unofficial convenience binaries and Docker i...  258
storm               Apache Storm is a free and open source distr...  176      [OK]
apache/zeppelin     Apache Zeppelin                161
apache/superset     Apache Superset                144
groovy              Apache Groovy is a multi-faceted language fo...  131      [OK]
apache/skywalking-oap-server Apache SkyWalking OAP Server  120
tomEE                Apache TomEE is an all-Apache Java EE certif...  97       [OK]
bitnami/apache      Bitnami Apache Docker Image    82
apache/apisix        Apache APISIX: Cloud-Native API Gateway        46
apache/druid         Apache Druid                    46
ubuntu/apache2      Apache, a secure & extensible open-source HT...  40
apache/nifi-registry Unofficial convenience binaries for Apache N...  38
apache/couchdb       Unofficial convenience binaries for CouchDB,...  22
apache/tika          Apache Tika Server - the content analysis to...  19
apache/apisix-dashbaod https://github.com/apache/apisix-dashbaod    18
apache/fineract      Apache Fineract                  9
apache/arrow-dev     Apache Arrow convenience images for developm...  3
root@ubuntu:~#
  
```

図20.6 : docker searchの使用

(上記のコマンドで**apache**の代わりに**httpd**を使っても、よく似た結果になります)。ここまでくれば、もう大丈夫でしょうから、これからは詳細な出力は表示しません。

4. コンテナを回収する：

```
$ sudo docker pull docker.io/httpd
```

すべてのコンポーネントがダウンロードされるまで、数分かかる場合があります。

5. インストールされているイメージを

一覧表示します：

```
$ sudo docker images
```

6. **httpd の docker** コンテナを起動します。**httpd**デーモンに接続されたので、ターミナルがハングアップするように見えます。

```
$ sudo docker run httpd
```

```
AH00558 : httpd : 172.17.0.2 を使用して、サーバーの完全修飾ドメイン名を確実に決定で
きませんでした。このメッセージを表示しないようにするには、グローバルに
'ServerName' ディレクティブを設定します。
```

```
.....
```

7. 上記の出力にある IP アドレスを指すグラフィカルな Web ブラウザを開くことができます。あるいは、テキストベースのブラウザを使うこともできます（特にグラフィカルな環境にない場合）。新しいターミナルウィンドウを開き（`httpd` コンテナが実行されているウィンドウを殺さないでください！）、次のコマンドのいずれかを実行します：

```
$ lynx http://172.17.0.2
$ lynx w3m://172.17.0.2
$ lynx elinks://172.17.0.2
```

お使いのシステムにインストールされているテキストベースのブラウザを使用してください。

8. 容器を止めて、片付けます。

```
$ sudo docker container list --all
```

```
コンテナ ID イメージ コマンド 作成 ステータス ポート 名
959a4ab95626 httpd "httpd-foreground" 2時間前 終了(0)
2時間前 pensive_sammet
```

```
$ sudo docker container rm pensive_sammet
```

```
ペンシル・サメット
```

9. これにより、イメージとそれに関連するストレージが `/var/lib/docker` または `/var/lib/containers` のいずれかに残されます（システムおよびディストリビューションによって異なる）。もし再利用する必要がなければ、これでクリーンアップできます：

```
sudo docker container prune
```

```
警告実行されていないコンテナはすべて削除されます
。本当に続行しますか？ [y/N] y
00b785df8f14faff679e759329f327de01d8d0efce645f7d708ef73c168d0f5
23ad214fbba9e2b1387c006f364a7e473af7c93f4d36940d001a237ea7e8309d
27526c4912abdb22e56813d26f19554f9cd5ae3123a789a534c96253496652f2
27d627aff076f871683cb92e6f89db6efbf47cc150fb5e71f9349c41b03a9ea5
35e7b44229d056073cb39a718a6c4cc624fc6b5e69c861908c15d9881ff45803
3e885f9f02a861adf49858516dff15c4432fab96afaf0cc40d62d66832857225
47c43e3eb2eff9092c82e806f64ce3b1bf720a922715d97a4ba5fa1c0d18de7e
```

```
$ sudo docker image list --all
```

レポトリ	TAG	画像ID	CREATED	SIZE
エイチティーティーピーディー			最新	a981c89925124 日前 145MB

```
$ sudo docker image rm httpd
```

```
タグなし: httpd:latest
Untagged: httpd@sha256:70999c4a17c796dd28f86f9c847b30f28abaed6ef1fd72a44282b1c941238804
Deleted: sha256:a981c8992512d65c9b450a9ecabb1cb9d35bb6b03f3640f86471032d5800d825
Deleted: sha256:0fbd49f82b45671985b3275bbe52bb36496047a8893146e7e6fc8b258c0c7274
Deleted: sha256:d4d2d5c5d610b9cd6063e316954eb0bdc0af36cf8c94570658278504f4c6bbf8 Deleted:
sha256:1c463a675bc2ad1cd823d7944a0c7cbea87635a248257455c5fd355c98bcfd0a Deleted:
sha256:6b66301e945a6b716b5fdef63c4e2e2dd692ec6aa5a84abb1165dec8a22538c6
```

するのもよいでしょう：

```
sudo systemctl stop docker
```


第21章

LinuxのファイルシステムとVFS



21.1	ファイルシステムの基礎知識	332
21.2	ファイルシステムの概念	333
21.3	仮想ファイルシステム(VFS)	335
21.4	利用可能なファイルシステム	336
21.5	ジャーナリングファイルシステム	338
21.6	特殊なファイルシステム	339
21.7	研究室	340

学習目標

このセッションが終わるころには、次のことができるようになっているはずです：

- ファイルシステムの基本的な構成について説明する。
- VFS(Virtual File System)の役割を理解する。
- Linuxで利用できるファイルシステムは何か、実際のシステムで利用できるファイルシステムは何かを知る。
- ジャーナリングファイルシステムは、なぜ大きな進化を遂げたのかを理解することができます。
- Linuxにおける特殊なファイルシステムの使用について説明する。

21.1 ファイルシステムの基礎知識

ファイルシステムの基礎知識

- Linuxプログラムは、ディスクセクタではなく、ファイルの読み書きを行います。
- ファイルは、物理的なI/O層をカモフラージュする抽象化されたものである。
- ファイルシステムは、物理的なパーティションに使用可能なフォーマットを作成します。
- UNIXライクなファイルシステムは、ツリー階層を採用している
 - ディレクトリには、ファイルや他のディレクトリが含まれる
 - すべてのパスやノードはルート（/）ディレクトリの下にある
- 複数のファイルシステムを1つのツリー構造に統合することができる（そして通常そうする）。
- Linuxでは、ファイルシステム・ソフトウェアとの通信に仮想ファイル

アプリケーションプログラムは、ファイルが保存されている実際のハードウェア上の物理的な場所を扱うのではなく、**ファイル**の読み書きを行います。

ファイルとその名前は、物理的なI/O層をカモフラージュする抽象化されたものです。コマンドラインから直接ディスクに書き込む（ファイルシステム層を無視する）ことは非常に危険であり、通常は低レベルのオペレーティングシステムソフトウェアによってのみ行われ、ユーザーアプリケーションによって行われることはありません。例外として、エンタープライズ・データ・ベースのような非常にハイエンドなソフトウェアでは、ファイルシステム関連のレイテンシをスキップするために、このような**cmdアクセス**を行うことがある。

ローカルファイルシステムは一般的にディスクパーティション内に存在し、ディスク上の物理パーティション、または**論理ボリュームマネージャ（LVM）**により管理される論理パーティションであることができます。ファイルシステムは、ネットワークの性質上、ネットワーク越しのローカルシステムには、その真の物理的な姿が完全に隠されていることもある。

21.2 ファイルシステムの概念

イノード

- ファイルの名前は、より基本的なオブジェクトである**inode**のプロパティに過ぎません。
- Inodeは、ファイルシステム上に保存されるとともに、メモリ上のデータ構造でもある。
- Inodeは、以下のようなファイルに関する情報を記述し、保存します：
 - パーミッション
 - ユーザー、グループの所有権
 - サイズ
 - タイムスタンプ(ナノ秒)
 - * 最終アクセス時間
 - * 最終修正時刻
 - * 変更時間
- ファイル名はinodeには格納されず、ディレクトリに格納されます。

inodeとは、ディスク上のデータ構造で、ファイルの位置などの属性を記述して保存するものです。

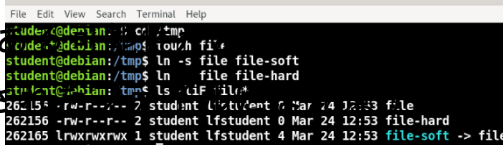
Linuxのファイルシステムに含まれるすべてのファイルは、そのファイル自身のinodeと関連付けられています。ファイルに関するすべてのデータは、そのinodeに含まれています。inodeは、名前、場所、ファイル属性（パーミッション、所有権など）、アクセス時間などのプロパティを追跡するためにオペレーティングシステムによって使用されます。このため、ファイルに関するすべてのI/O活動は、通常、そのファイルのinodeにも関与します。

タイムスタンプが3つあることに注意してください：

1. **アクセス時間**：ファイルが何らかの目的で最後にアクセスされた時間。
2. **修正時間**：ファイルの内容が最後に修正された時間。
3. **変更時間**：パーミッション、所有権、ファイル名、ハードリンクなどの変更により、ファイルのinodeが最後に変更された時間です。

ハードリンクとソフトリンク

- ハードリンクはinodeを指します。
 - 2つ以上のファイルが同じinode（ハードリンク）を指すことができる
 - ハードリンクされたファイルはすべて同じファイルシステム上にある必要がある
- ソフト（またはシンボリック）リンクは、関連するinodeを持つファイル名を指し示します。
 - ソフトリンクされたファイルは、異なるファイルシステム上にある可能性がある
 - ターゲットが存在しない、またはまだマウントされていない、ぶら下がった状態で
- ハードリンクされたファイルは、削除されたファイルに置き換えられることがある
 - ある場所で



```

student@debian:~$ cd /tmp
student@debian:/tmp$ touch file
student@debian:/tmp$ ln -s file file-soft
student@debian:/tmp$ ln file file-hard
student@debian:/tmp$ ls -li file
-rw-r--r-- 2 student lfstudent 0 Mar 24 12:53 file
-rw-r--r-- 2 student lfstudent 0 Mar 24 12:53 file-hard
lrwxrwxrwx 1 student lfstudent 4 Mar 24 12:53 file-soft -> file

```

ディレクトリファイルは、ファイル名とinodeを関連付けるために使用される特定のタイプのファイルです。関連付けるには、2つの方法があります（またはリンク）ファイル名をinodeで指定します：

- ハードリンクは、inodeを指す。オプションなしでlnを使用することで作成されます。
- ソフト（またはシンボリック）リンクは、関連するinodeを持つファイル名を指し示します。ソフトリンクは、lnに-sを付けて作成します。オプションを使用することができます。

ディレクトリ・ファイルの内容とinodeの各関連付けは、リンクと呼ばれます。リンクはlnで追加作成することができます。2つ以上のディレクトリエントリが同じinode（ハードリンク）を指すことが可能であり（かなり一般的）、ファイルは複数の名前でも知られ、それぞれがディレクトリ構造で独自の場所を持つことができます。しかし、どの名前が使用されていても、1つのinodeしか持つことができません。

プロセスがパス名を参照するとき、カーネルはディレクトリを検索して対応するinode番号を探します。名前がinode番号に変換された後、inodeはメモリにロードされ、その後のリクエストで使用されます。

通常、ファイルを変更しても、同じinodeを参照するハードリンクが壊れることはありません。しかし、ファイルをコピーして変更してから置き換えたり、ファイルを削除して置き換えたりして、その過程でリンクがなくなった新しいファイルを作成する（ひどく書かれた）アプリケーションもあります。ですから、この動作が意図されたものでない場合は、目を離さないようにしてください。

21.3 仮想ファイルシステム(VFS)

ブイエフエス

Virtual FileSystem (**VFS**) のことです :

- 抽象化レイヤー
- カーネルと実際のファイルシステム (すべての記憶媒体上) の間で通信を行う。
- 実ファイルシステムは**VFS**レイヤーに登録する

Linuxは、すべての最新のオペレーティングシステムと同様に、**仮想ファイルシステム(VFS)**を実装しています。アプリケーションがファイルにアクセスする必要がある場合、**VFS**抽象化レイヤーと対話し、VFSはすべてのI/Oシステムコール (読み取り、書き込みなど) を、特定の実際のファイルシステムに関連する特定のコードに変換します。

このため、アプリケーションは、ファイルシステム本体や、ファイルシステムが存在する物理的なメディアやハードウェアを考慮する必要がない。さらに、ネットワークファイルシステム (**NFS**など) も透過的に扱うことができます。

このため、**Linux**は他のどのオペレーティングシステムよりも多くのファイルシステムの種類を扱うことができます。この民主的な属性が、Linuxの成功の大きな要因となっています。

ほとんどのファイルシステムは、完全な読み取りと書き込みが可能です。少数のファイルシステムは、読み取りアクセスのみで、おそらく実験的な書き込みアクセスが可能です。ファイルシステムの種類によっては、特に**非UNIX**ベースのものは、**VFS**で表現するために、より多くの操作を必要とする場合があります。

VFS は、3 種類のユーザに対してどのように異なるパーミッションを指定するかを想定する必要があり、そのような動作はマウント操作に影響される可能性があるため、**vfat** のようなバリエーションは、オーナー/グループ/ワールドのフィールドに対して明確な読み取り/書き込み/実行パーミッションを持ちません。カーネル以外のファイルシステムの実装として、読み書き可能な **ntfs-3g** ([https :](https://sourceforge.net/projects/ntfs-3g/)

[//sourceforge.net/projects/ntfs-3g/](https://sourceforge.net/projects/ntfs-3g/))は、信頼性は高いが、カーネル内ファイルシステムより性能が劣る。

21.4 利用可能なファイルシステム

利用可能なファイルシステム

- **Linuxは、他のどのオペレーティングシステムよりも多くのファイルシステムの種類を扱うことができます。**
- この民主的な柔軟性が、成功の大きな要因となっています。
- ほとんどのファイルシステムは、完全な読み取り/書き込みアクセスを持っていますが、いくつかは読み取り専用アクセスを持っています。
- 一般的に使用されるファイルシステムには、**ext4**、**xfs**、**btrfs**、**squashfs**、**nfs**があります。

および**vfat**

- 現在サポートされているファイルシステムの一覧は、[/proc/filesystems](#)で見ることができます。

現在実行中のLinuxカーネルに登録され、理解されているファイルシステムタイプの一覧は、次のようにして見るすることができます：

```
$ cat /proc/filesystems
```

ノード	シスフ	ノード	ビービーエフ	ノード	自臭症
ド	タンボボ	ノード	肺結核	ノード	待ち行列
ド	ビー	ノデ	ラムズヒュッ	ノード	Pスト
ブ	ディ	ブ	テ	ノード	アヒュ
ノ	ープ	デ	rpc_pipefs	ノード	ーズブ
デ	イブ	ノード		ノード	ラック
ノード	ロッ	ノード		ノード	フュー
ノード	グループ	ノード	ちょい役	ノード	ズ
ノード	グループ2	ノード		ノード	フュー
ノード	咬合器		スカッシュフ		セック
ノード	デフトンプフ		ーズ		ル
ノード			バット		エクス
					テンド3
					エクス
					テンド
					ツ

ノデ ブノ デブ ノード ヴ	configs debugfs トレース	ノードヴ	エムエス ディーオ ーエック スファッ ト	ノードヴ	ext4 ntfs3 オーバ ーレイ
ノー ドヴ ノー ドヴ	セキュリテ ィフーズ ソックス	ノードヴ ノードヴ	エヌエス エヌエフエス フォー エヌエフエス ディー		

(nodevを持つものはストレージ上に存在しない**特殊なファイルシステム**です。) 追加のファイルシステムは、システムがそれらを使用するパーティションにアクセスしようとするときだけ、そのコードがモジュールとしてロードされるかもしれません。

ファイルシステムの種類

Linuxは多くのファイルシステムをサポートしており、そのほとんどが完全な読み取りと書き込みのアクセスを備えています：

- **ext4**: Linuxネイティブファイルシステム（それ以前の**ext2**、**ext3**も同様）
- **XFS** : **SGI**が独自に開発した高性能ファイルシステム
- **JFS** : **IBM**が独自に開発した高性能ファイルシステム
- **Windows**ネイティブです : **fat12**、**fat16**、**fat32**、**vfat**、**ntfs**
- メモリ上にのみ常駐する疑似ファイルシステム（**proc**、**sysfs**、**devfs**、**debugfs**など）。
- **NFS**、**coda**、**afs**などのネットワークファイルシステム
- など

Linuxは多くの種類のファイルシステムをサポートしており、常に追加され続けています。これらは、以下のようなものがあります：

- **ext4**など、Linux用に直接設計されたネイティブファイルシステム。
- **xfs**や**ntfs**など、他のOSから持ち込まれたファイルシステム。
- 実際のファイルシステムではないが、特定の目的を達成するためにインフラを使用する特殊なファイルシステム。
デバッグフーズ

Linuxは非常にモジュール化された構造になっており、この柔軟性が活かされている。

21.5 ジャーナリングファイルシステム

ジャーナリングファイルシステム

- クラッシュや異常終了から優雅に回復する
- リカバリーやリペアを極めて迅速に行うことができる
- 完全に完了しなければ取り消されるアトミックトランザクションを扱う。
- ジャーナルログファイルの操作
- 多くの先進的な機能を含む
- 現在利用可能：**ext4**、**ext3**、**xfs**、**jfs**、**reiserfs**、**btrfs**

ジャーナリングファイルシステムは、システムクラッシュや非グレースフルシャットダウンからほとんど破損することなく回復し、その回復も非常に速い。このことは、より多くのオペレーションを行うという代償を伴いますが、追加的な機能拡張により、その代償を補って余りあるものとなります。

ジャーナリングファイルシステムでは、操作は**トランザクション**に分類されます。トランザクションはエラーなく、**アトミカル**に完了しなければならない；さもなければファイルシステムは変更されない。トランザクションはログファイルとして管理されます。エラーが発生した場合、通常は最後のトランザクションだけを調べる必要があります。

以下のジャーナリングファイルシステムは、**Linux**で自由に利用できます：

- **ext3**は、それ以前のジャーナルしない**ext2**ファイルシステムの拡張版である。
- **ext4**は**ext3**を大幅に拡張したものです。エクステント、48ビットのブロック番号、最大16TBのサイズなどの特徴があります。ほとんどの場合
Linuxディストリビューションでは、かなり前から**ext4**がデフォルトのファイルシステムとして採用されています。
- **reiserfs**はLinuxで使われた最初のジャーナリング実装だったが、リーダーシップを失い、それ以上の開発は放棄された。
- JFSはもともと**IBM**の製品で、**IBM**の**AIX**オペレーティングシステムから移植されたものです。
- **XFS**はもともと**SGI**の製品で、**SGI**の**IRIX**オペレーティングシステムから移植されたものです。**RHEL 7**では、デフォルトのファイルシステムとして**XFS**を採用した。
- **btrfs**はジャーナリングファイルシステムの中で最も新しいもので、まだ急速に開発が進んでいます。**SUSE**のデフォルトであり**openSUSE**のシステム。

21.6 特殊なファイルシステム

特殊なファイルシステム

- 特殊なファイルシステムは、システム内部へのアクセスや調整、特定の機能を実装するために使用されます。
- マウントポイントを持つものもある（**proc**は/procに、**sys**は/sysになど）
- マウントポイントがないものもあります（**sockfs**や**pipefs**など）。
- これらの特殊なファイルシステムは、本当のファイルシステムではなく、ファイルシステムの構造的な抽象化をデータと機能を整理するための有用な方法と見なすカーネル設備またはサブシステムである。

Linuxでは、特定のタスクのために特別なファイルシステムを使用することが広く採用されています。これらは、様々なカーネルデータ構造へのアクセスやカーネル動作のチューニング、あるいは特定の機能の実装に特に有用です。一部のリストです：

表21.1：特殊なファイルシステム

ファイルシステム	マウントポイント	目的
ルートフーズ	なし	カーネルロード時に、空のルートディレクトリを提供します。
ハゲゼルブフ	どこでも	拡張ページアクセス（x86では2または4MB）を提供します。
ビーディーブイ	なし	ブロックデバイスに使用します。
ブロック	/proc	多くのカーネル構造やサブシステムに擬似的にファイルシステムでアクセスすることができます。
ソックス	なし	BSDソケットで使用
タンブフーズ	どこでも	スワッピング、リサイズが可能なRAMディスク。
えんぎがわるい	なし	System V IPC Shared Memoryで使用されます。
肺結核	なし	パイプに使用します。
雑魚寝	どこでも	様々な実行形式によって使用されます。

340	ちょい役	/dev/pts	Unix98の疑似端末で使用 第21章 Linux ファイルシステムとvfs
	usbfs	/proc/bus/usb	USB サブシステムでダイナミックデバイス用に使用されます。
	シスフ	/sys	デバイスツリーとして使用します。
	デバッグフーズ	/sys/kernel/debug になります。	ファイルアクセスの簡単なデバッグに使用します。

このような特殊なファイルシステムの中には、マウントポイントを持たないものがあることに注意してください。これは、ユーザーアプリケーションはそれらと対話しないが、カーネルはVFSレイヤーとコードを活用してそれらを使用することを意味します。

21.7 研究室



ビデオデモンストレーションのリソース

利用可能なファイルシステムを利用したデモ.mp4

練習問題 21.1 : tmpfs 特殊ファイルシステム

tmpfsは、Linuxで使用される多くの特殊なファイルシステムの1つです。これらの中には、ファイルシステムとして実際に使用されるのではなく、ファイルシステムの抽象化を利用しているだけのものもあります。しかし、**tmpfs**は、アプリケーションがI/Oを行うことができる本物のファイルシステムです。

基本的に**tmpfs**はラムディスクとして機能し、純粋にメモリ上に存在する。しかし、昔ながらの従来のラムディスクの実装が持っていなかった、いくつかの素晴らしい特性を持っています：

1. ファイルシステムは、そのサイズ（ひいては使用されるメモリ）を動的に調整します。0から始まり、マウントされた最大サイズまで必要に応じて拡張されます。
2. RAMを使い切った場合、**tmpfs**はスワップスペースを利用することができます。（ただし、ファイルシステムの最大容量以上のものを入れようとすることはできません）。
3. **tmpfs**は、**ext3**や**vfat**のような通常のファイルシステムを配置する必要はなく、ファイルやI/Oを扱うための独自のメソッドを持ち、それが実際にはメモリ上の単なるスペースであることを認識し（実際にはブロックデバイスではない）、その結果、速度に最適化されています。

そのため、mkfsコマンドであらかじめファイルシステムをフォーマットしておく必要はなく、単にマウントして使うだけでよい。

のようなコマンドで、ディレクトリ構造上の任意の場所に**tmpfs**の新しいインスタンスをマウントします：

```
$ sudo mkdir /mnt/tmpfs
```

```
$ sudo mount -t tmpfs none /mnt/tmpfs
```

ファイルシステムに与えられている容量と、使用している容量を確認します：

```
$ df -h /mnt/tmpfs
```

しかし、使用量はゼロで、**/mnt/tmpfs**にファイルを置くと増えていくだけです。

のようにマウントオプションで割り当てサイズを変更することができます：

```
$ sudo mount -t tmpfs -o size=1G none /mnt/tmpfs
```

容量がいっぱいになるまで入れてみて、どうなるか試してみたいでしょうか。使い終わったらアンマウントするのを忘れないでください：

```
$ sudo umount /mnt/tmpfs
```

事実上すべての最新Linuxディストリビューションは、**/dev/shm**に**tmpfs**のインスタンスをマウントします：

```
$ df -h /dev/shm
```

ファイルシステム	Type	Size	Used	Avail	Use%	tmpfs にマウント
		.9G	24M	9G	1%	/dev/shm

プロセス間通信の仕組みとして**POSIX**共有メモリを使用している場合など、多くのアプリケーションがこれを使用しています。**dev/shm**は、どのユーザーでもファイルの作成、読み込み、書き込みができるため、メモリ上に一時ファイルを作

成するのに適した場所です。

`dev/shm`にいくつかのファイルを作成し、**df**でファイルシステムがどのように埋まっていくかをメモしておきます。

また、多くのディストリビューションでは、例えば**RHEL**システムでは、**tmpfs**の複数のインスタンスがマウントされています：

```
$ df -h | grep ' tmpfs '
```

tmpfs	7.8G	38M	7.8G	1%	/dev/shm
tmpfs	7.8G	18M	7.8G	1%	/run
tmpfs	7.8G	0	7.8G	0%	/sys/fs/cgroup
tmpfs	1.6G	1.2M	1.6G	1%	/run/user/42
tmpfs	1.6G	56K	1.6G	1%	/run/user/1000

これは 16 GB の RAM を搭載したシステムで実行されたもので、明らかにこれらの **tmpfs** ファイルシステムが、それぞれ割り当てられたデフォルトの約 8 GB を実際に使用することはできません！



ご注意ください

ディストリビューションによっては（**Fedora**など）、（デフォルトで）**/tmp**を**tmpfs**システムとしてマウントすることがあります。このような場合、メモリ不足にならないように、**/tmp**に大きなファイルを置くことを避ける必要があります。このような場合は、メモリ不足にならないように、**/tmp**に大きなファイルを置かないようにしなければなりません。

第22章

ディスクパーティション



22.1	一般的なディスクの種類	344
22.2	ディスクのジオメトリ	345
22.3	パーティショニング	346
22.4	パーティションテーブル	348
22.5	ディスクデバイスの命名	350
22.6	blkid と lsblk	351
22.7	パーティションのサイズアップ	353
22.8	パーティションテーブルのバックアップとリストア	354
22.9	パーティションテーブルエディタ	356
22.10	エフディスク	357
22.11	研究室	358

学習目標

このセッションが終わるころには、次のことができるようになっているはずです：

- 最も一般的なハードディスクとデータバスの種類を説明し、対比する。
- ディスクジオメトリやその他のパーティショニングの概念を説明する。
- ディスクデバイスの名前の付け方、関連するデバイスノードの特定方法を理解する。
- fdisk、blkid、lsblkなどのユーティリティを使用します。
- さまざまなパーティション戦略を区別し、選択する。
- パーティションテーブルのバックアップと復元。

343

22.1 一般的なディスクの種類

一般的なディスクの種類

- **SATA** (Serial AT Attachment)
 - データ転送速度が速く、**IDE**よりケーブルが小さい。
 - **SCSI**デバイスとして認識される
- **スカジ** (Small Computer Systems Interface)
 - 一般的に速い
 - ファスト、ワイド、ウルトラ、ウルトラワイドなど多数のバージョンがある
- シリアルアタッチドSCSI
 - 新しいポイントツーポイントシリアルプロトコル
 - **SATA**より優れたパフォーマンス
- **USB**
 - フラッシュドライブ、フロッピーを含む
 - **SCSI**デバイスとして認識される
- **SSD**
 - 可動部がなく、通常のコントローラーに取り付け可能
- **IDE、EIDE** (統合ドライブエレクトロニクス、エンハンスドIDE)
 - オボロ

ハードディスクの種類はいくつもあり、それぞれ接続するデータバスの種類や、速度、容量、複数のドライブの同時動作の良し悪しなどの特徴があります。

SATAディスクは、古い**IDE**ドライブを置き換えるために設計されました。より小さなケーブルサイズ (7ピン)、ネイティブホットスワップ、より高速で効率的なデータ転送を提供します。

SASはサーバーに向いています。違いについては、<https://store.hp.com/us/en/tech-takes/sas-vs-sata>、わかりやすく解説しています。

SCSIディスクはナロー (8ビットバス) からワイド (16ビットバス) まであり、転送速度は約5MB/秒 (ナロー、スタンダード**SCSI**) から約160MB/秒 (**ウルトラワイドSCSI-3**) です。

最近の**SSD**ドライブ (Solid State Drive) は価格が下がり、可動部品がなく、回転メディアを持つドライブより消費電力が少なく、転送速度が速い。内蔵**SSD**は、従来のドライブと同じフォームファクターで、同じ筐体に設置されているほどです。

SSDはまだかなり高価ですが、価格は低下しています。同じマシンに**SSD**と回転式ドライブの両方を搭載し、アクセス頻度が高く、パフォーマンスが重要なデータ転送は**SSD**で行うことが一般的です。

22.2 ディスクのジオメトリ

ディスクのジオメトリ

- 回転ディスクは1枚以上のプラッタで構成され、各プラッタは1つ以上のヘッドで読み取られます
- ディスクの回転に伴い、ヘッドがプラッターから円形のトラックを読み取る
- 円形トラックはセクタと呼ばれるデータブロックに分割される
- 当初、セクタは512バイトのデータだったが、現在は4096バイトが一般的
- シリンダーとは、すべてのプラッターに同じトラックを配置したグループのことです
- SSDの場合、これらのジオメトリの概念は意味をなさない。

```
$ sudo fdisk -l /dev/sdc |grep -i セクター
```

```
ディスク /dev/sdc: 1.8 TiB、2000398934016バイト、3907029168セクター
単位: 1 * 512 = 512バイトのセクタ
セクタサイズ (論理/物理) : 512バイト/4096バイト
```

物理的な構造イメージは、ドライブの内部電子機器が実際にその多くを見えなくしているため、あまり意味がなくなってきました。さらに、SSDには可動部品や上記の成分のようなものではありません。

fdiskでジオメトリを確認することができます：

```
File Edit View Search Terminal Help
c7:/tmp>sudo fdisk -l /dev/sda

Disk /dev/sda: 2000.4 GB, 2000398934016 bytes, 3907029168 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
Disk label type: dos
Disk identifier: 0x000852df

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1             2048     1048578047    524288000   8e  Linux LVM
/dev/sda2    1048578048     2097154047    524288000   8e  Linux LVM
/dev/sda3    2097154048     3907028991    904937472    5  Extended
/dev/sda5    2097156096     3145732095    524288000   8e  Linux LVM
/dev/sda6    3890448384     3907028991      8290304   82  Linux swap / Solaris
c7:/tmp>
```

図 22.1 : fdiskの使用

lオプションは、対話モードに入ることなく、単にパーティションテーブルをリストアップするために使用されていることに注意してください。

歴史的にディスクは512バイトのセクタで製造されていましたが、現在では4KBが圧倒的に多く、セクタサイズを大きくす

346 ると転送速度が速くなります。Linuxでは、後方互換性のためにまだ512バイトの論理セクタサイズを使用していますが、
これは単にソフトウェアでふけをするためです。第22章 ディスクパーティショニング

22.3 パーティショニング

パーティション機構

- ディスクはパーティションに分割される
- パーティションは、ディスク上の物理的に連続した領域です
- 最も一般的なアーキテクチャでは、2つのパーティションスキームが使用されています：
 - マスタブートレコード
 - GPT (GUID パーティションテーブル)
- MBRの歴史はMSDOSの初期にさかのぼる
プライマリパーティションは最大4つまで、プライマリパーティションの1つは拡張パーティションとして指定でき、15個のパーティションからなる論理パーティションに細分化されます。
- GPTはすべての最新システムに搭載されており、UEFI (Unified Extensible Firmware Interface) をベースにしています。
デフォルトでは、最大128個のプライマリパーティションが使用できます。

- MBR方式を使用する場合：

SCSIドライブ、例えば `/dev/sda` があれば、`/dev/sda1` が第1プライマリパーティション、`/dev/sda2` が第2プライマリパーティションになります。もし拡張パーティション `/dev/sda3` を作れば、論理パーティションに分割することができます。4つ以上のパーティションはすべて論理パーティションです (拡張パーティションに含まれることを意味します)。拡張パーティションは1つしかありませんが、多数の論理パーティションに分割することができます。

注意：Linuxでは、パーティションがシリンダー境界で開始または終了する必要はありませんが、他のオペレーティングシステムは、そうしないと文句を言うかもしれません。このため、広く普及しているLinuxのパーティション作成ユーティリティでは、境界で終了するように工夫されています。当然ながら、パーティションは重ならないようにする必要があります。

- GPT方式を使用する場合：

拡張パーティションは必要なく、(デフォルトで) 128個のプライマリパーティションを持つことができます。パーティションサイズは最大²³³TBまで可能です！(MBRの場合は2TBまで)

なぜパーティションなのか？

- ユーザーとアプリケーションのデータをオペレーティングシステムファイルから分離する
- オペレーティングシステムやマシン間の共有
- システム部分ごとに異なるクォータや権限を付与することで、セキュリティ強化を実現
- サイズに関する懸念、可変および揮発性ストレージをスタブレから隔離すること
- 使用頻度の高いデータをより高速な記録媒体に置くことによる性能向上
- スワップスペースはデータから分離することができ、ハイバネーションストレージとしても使用できる

何をどのようにパーティション分けするかは、考えるべきことです。パーティションを分ける理由としては、セキュリティ、クォータ、設定、サイズ制限の粒度を上げることが挙げられます。また、データを保護するためにパーティションを分けることもできます。

一般的なパーティションレイアウトは、`/boot`パーティション、ルートファイルシステム用のパーティション、スワップパーティション、`/home`ディレクトリツリー用のパーティションです。

パーティションのサイズを変更するのは、インストールや作成時に比べて、後からでは難しいということを念頭に置いておいてください。それなりの計画を立ててください。

22.4 パーティションテーブル

MBRパーティションテーブル

- パーティションテーブルは、第1セクターの最後の64バイト、**MBR**（マスターブートレコード）の512バイトにあります。

- パーティションテーブルの各エントリは16バイト長で、4つのプライマリパーティションのうちの1つを記述しています。各エントリーの情報は以下の通りです：

- アクティブビット
- はじめに 宛先 において **CHS**（シリンダー/ヘッド/セクタ）フォーマット
- パーティション タイプ コードになります、 を示す：
Linux、Linux LVM、ntfs、swap、その他

Linuxでは、LBA（Linear Block Addressing）方式によるアドレス指定に、最後の2フィールドのみを使用します。

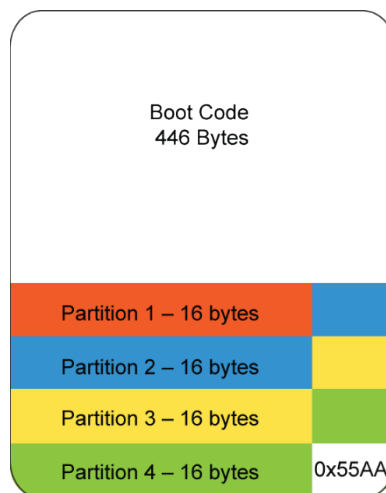


図22.2：MBRディスクパーティションテーブル

- **CHS**の終了アドレス
- 0から直線的に数えて、セクターを開始する
- パーティション内のセクタ数

ディスクパーティションテーブルは、ディスクの**MBR**に含まれ、446バイトのブートレコードに続く64バイトの部分である。ディスク上の1つのパーティションは、アクティブと表示されることがあります。システムが起動するとき、そのパーティションはマスターブートローダーがロードするアイテムを探す場所です。

拡張パーティションは1つしかありませんが、そのパーティションには複数の論理パーティションが含まれている可能性があることに注意してください。

MBRの構造は、オペレーティングシステムに依存しない規約で定義されています。最初の446バイトは、プログラムコード用に予約されています。通常、ブートローダープログラムの一部が格納されます。次の64バイトは、最大4つのエントリを持つパーティションテーブルのためのスペースです。オペレーティングシステムは、ハードディスクを処理するためにこのテーブルを必要とします。

Linux システムでは、**CHS** の開始アドレスと終了アドレスは無視されます。

マジックナンバー、シグネチャーワード、エンドオブセクターマーカと呼ばれる**MBR**の最後の2バイトは、常に0x55AAの値を持っています(興味のある方はご注意ください)。

GPTパーティションテーブル

- 最近のハードウェアは**GPT**をサポートしており、**MBR**のサポートは徐々に消えていくだろう
- 保護MBRは後方互換性のため、UEFIシステムを古い方法で起動することができます。
- ディスクの先頭と末尾に、メタ・データを記述した**GPT**ヘッダーのコピーを2つ配置する：
 - ディスク上の使用可能なブロックをリストアップする
 - パーティション数
 - パーティションエントリーのサイズ

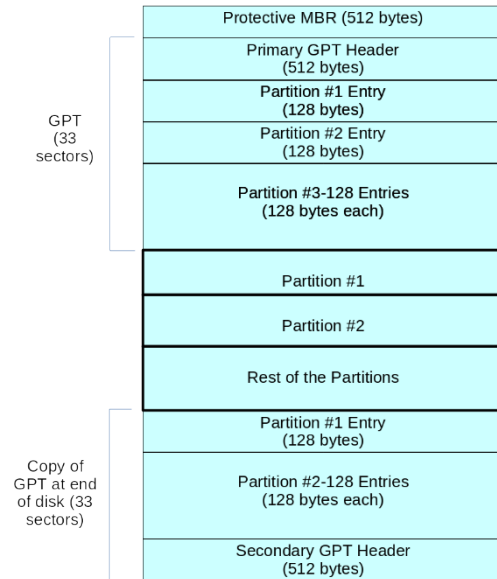


図22.3 : GPT ディスクパーティション

blkidユーティリティ（後述）は、パーティションに関する情報を表示します：

- 最新のUEFI/GPTシステムで：

```
$ sudo blkid /dev/sda8
```

```
/dev/sda8: LABEL="RHEL8" UUID="53ea9807-fd58-4433-9460-d03ec36f73a3" BLOCK_SIZE="4096" TYPE="ext4"
↳ PARTUUID="0c79e35b-e58b-4ce3-bd34-45651b01cf43"
```

- レガシーMBRシステムにおいて：

```
$ sudo blkid /dev/sdb2
```

```
/dev/sdb2: LABEL="RHEL8" UUID="6921b738-1e36-429a-89be-8b97cf2f0556" BLOCK_SIZE="4096" TYPE="ext4"
↳ partuuid="00022650-02"
```

どちらの例も、パーティションそのものではなく、パーティション上のファイルシステムを記述する一意のUUIDを与えることに注意してください。ファイルシステムが再フォーマットされると、変更されます。

GPTパーティションはまた、パーティションを記述するPARTUUIDを与え、ファイルシステムが再フォーマットされても同じままである。ハードウェアがサポートしていれば、**MBR**システムを**GPT**に移行することは可能ですが、そうすることでマシンをブリックすることは難しくありません。このように、通常、リスクと見合うだけのメリットはありません。

22.5 ディスクデバイスの命名

ディスクデバイスとデバイスノードの命名に

- `dev`ディレクトリにデバイスファイルが見つかった
- 名称形式：`xx[y[z]`
 - `xx` はデバイスの種類（通常は `sd`）です。
 - `y` はドライブ番号の文字（`a,b,c`,等）です。
 - `z` はパーティション番号
- サンプルです：
 - 最初に見つかったドライブは `/dev/sda` です。
 - `/dev/sdb2` は2台目のドライブにある2つ目のパーティションです

Linuxカーネルは、`/dev`ディレクトリにあるデバイスノードを介して、ディスクと低レベルで対話します。通常、デバイスノードはカーネルの仮想ファイルシステムのインフラを通してのみアクセスされます。デバイスノードを介した生のアクセスは、ファイルシステムを破壊する非常に効率的な方法です。

適切なrawアクセスの例として、次のようにパーティションをフォーマットすることができます：

```
$ sudo mkfs.ext4 /dev/sda9
```

SCSIおよびSATAディスクのデバイスノードは、簡単な命名規則に従っています：

- 最初のハードディスクは `/dev/sda` です。
- 2台目のハードディスクは `/dev/sdb` です。
- など

のようにパーティションも簡単に列挙できます：

- `/dev/sdb1` は、2番目のディスクの最初のパーティションです。
- `/dev/sdc4` は、3番目のディスクの4番目のパーティションです。

上記で `sd` はSCSIまたはSATAディスクを意味します。IDEディスクが見つかった時代には、`/dev/hda3` になっていたはずですが、`/dev/hdb` など。

`ls -l /dev` を実行すると、現在利用可能なディスクデバイスのノードが表示されます。

22.6 blkid と lsblk

ブリキ

- ブロックデバイスの属性を検索して印刷するコマンドラインユーティリティ
- ブロックデバイスの内容を報告する
- ブロックデバイスの属性に関する報告
- サンプルコマンドです：
\$ sudo blkid
\$ sudo blkid /dev/sda* です。
\$ sudo blkid -L root

blkidは、ブロックデバイスを探し出し、その属性を報告するユーティリティです。libblkid ライブラリで動作します。引数として、特定のデバイスまたはデバイスのリストを取ることができます。

ブロックデバイスが保持するコンテンツ（ファイルシステム、スワップなど）の種類や、コンテンツのメタデータ（LABELやUUIDフィールドなど）から属性（トークン、NAME=値のペア）を判断することができます。

blkidは、フィンガープリント可能なデータを含むデバイスでのみ動作します。例えば、空のパーティションはブロック識別UUIDを生成しません。

blkidは、特定のNAME=値のペアを持つデバイスを検索するか、1つまたは複数のデバイスのNAME=値のペアを表示するかの2つの主要な操作形態があります。

引数がない場合は、すべてのデバイスについて報告されます。デバイスを指定する方法と報告する属性を指定するオプションは非常に多くあります。

File Edit View Search Terminal Help

```
c7:/tmp>sudo blkid /dev/sda*
/dev/sda: PTTYPE="dos"
/dev/sda1: UUID="A9oz6n-r4Z9-ICGS-i3Pt-ywfK-t9wH-VsGvWa" TYPE="LVM2_member"
/dev/sda2: UUID="sBWbb3-kUDa-oPf8-U1Qo-J8fB-CfTQ-gQ1Jr5" TYPE="LVM2_member"
/dev/sda3: PTTYPE="dos"
/dev/sda5: UUID="wQlEbc-w48N-u3qz-5MAe-Q1Wm-xTf9-L3oAwf" TYPE="LVM2_member"
/dev/sda6: LABEL="SWAP" UUID="6a045706-62dc-4ca4-bc54-e07e41141585" TYPE="swap"
c7:/tmp>
```

図22.4 : blkidの使用

エクスブルック

- ブロックデバイスの情報をツリー形式で提示する

```
File Edit View Search Terminal Help
c7:/tmp>lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sdb          8:16  0 238.5G 0 disk
├─sdb2       8:18  0    1K 0 part
├─sdb7       8:23  0    27G 0 part
├─└─VG-vms   253:12  0   184G 0 lvm  /VMS
├─sdb5       8:21  0   128G 0 part
├─└─VG-local 253:10  0    24G 0 lvm  /usr/local
├─└─VG-src   253:11  0    11G 0 lvm  /usr/src
├─└─VG-vms   253:12  0   184G 0 lvm  /VMS
├─sdb1       8:17  0   19.5G 0 part /
├─sdb6       8:22  0    64G 0 part
├─└─VG-vms   253:12  0   184G 0 lvm  /VMS
loop0        7:0    0    2.5G 0 loop /usr/src/KERNELS
sda          8:0    0    1.8T 0 disk
├─sda2       8:2    0   500G 0 part
├─└─VG2-P    253:6    0   125G 0 lvm
├─└─VG2-virtual 253:4    0   350G 0 lvm  /VIRTUAL
├─└─VG2-isabelle 253:7    0   128G 0 lvm
├─sda5       8:5    0   500G 0 part
├─└─VG2-PLAY  253:9    0   100G 0 lvm
├─sda3       8:3    0    1K 0 part
├─sda1       8:1    0   500G 0 part
├─└─VG2-dead  253:1    0    70G 0 lvm  /DEAD
├─└─VG2-dead2 253:8    0   100G 0 lvm  /DEAD2
├─└─VG2-P    253:6    0   125G 0 lvm
├─└─VG2-virtual 253:4    0   350G 0 lvm  /VIRTUAL
├─└─VG2-iso_images 253:2    0   110G 0 lvm  /ISO_IMAGES
├─└─VG2-pictures 253:0    0    20G 0 lvm  /PICTURES
├─└─VG2-w7back  253:5    0    50G 0 lvm
├─└─VG2-audio  253:3    0    16G 0 lvm  /AUDIO
└─sda6       8:6    0    7.9G 0 part [SWAP]
```

図22.5 lsblkの使い方

\$ lsblk -h

使用することです：

lsblk [オプション] [<デバイス> ...] です。

ブロックデバイスの情報を一覧表示しま

す。オプションです：

- a, --all 全機種印刷
- b, --bytes SIZE を可読形式で表示するのではなく、バイト単位で表示する。
- d, --nodeps 隷属を刷るな
- D, --discard プリント廃棄機能
- z, --zone プリントゾーンモデル
- e, --exclude <list> メジャー番号でデバイスを除外する（デフォルト：RAMディスク）。
- f, --fs ファイルシステムに関する情報を出力する
- i, --ascii Ascii文字のみを使用する
- l, --include <list> 指定したメジャー番号のデバイスのみを表示する。
- J, --json JSON出力形式を使用する
- l, --list 使用リスト形式出力
- T, --tree 使用ツリー形式出力
- m, --perms パーミッションに関する情報を出力する
- n, --no-headers 見出しを印刷しない
- o, --output <list> 出力する列を指定する。
- O, --output-all 全列出力
- p, --paths プリント完全装置パス
- P, --pairs key="value" の出力形式を使う
- r, --raw 生出力形式を使用する
- s, --inverse 逆頼み
- S, --scsi SCSIデバイスに関する情報を出力します。
- t, --topology トポロジーに関する出力情報
- x, --sort <column> 出力を<column>でソートする。

-h, --help このヘルプを表示する

-V, --version ディスプレイバージョン

利用可能な出力カラム：

.....

詳しくはlsblk(8)を参照してください。

22.7 パーティションのサイズアップ

パーティションのサイズアップ

- 多くのLinuxシステムでは、最低2つのパーティションを使用します：

ルート

- ファイルシステムで使用
- ほとんどのLinuxインストールでは、複数のファイルシステムが存在します。
- サイズ変更やや不便を感じる（可能な限りLVMを使用することができる）

スワップ

- 物理メモリの拡張として使用
- 一般的な選択肢は、RAMサイズと同等
- スワップファイルだけでなく、複数のスワップパーティションがある場合があります。

- Ubuntuを含む一部のディストリビューションでは、パーティションで

ほとんどのインストールでは、複数のパーティションに複数のファイルシステムがあり、それらはマウントポイントで結合されています。ほとんどのファイルシステムでは、ルートパーティションのサイズを変更することは困難です。LVM（後述）を使用することで、これを容易にすることができます。

ルートパーティションだけでLinuxを動かすことも可能ですが、多くのシステムでは、バックアップの容易さ、ディスクドライブの効率的な使用、セキュリティの向上などのために、より多くのパーティションを使用しています。

通常、スワップサイズは物理メモリと同じサイズであることが推奨されていますが、その2倍が推奨されることもあります。しかし、正しい選択は、システムの使用シナリオやハードウェアの能力などの関連する問題に依存します。

スワップをどんどん増やしても、ある時点で無駄になってしまうので、必ずしも役立つとは限りません。メモリを増設するか、システムのセットアップを見直す必要があります。

古い回転式ハードディスクドライブでは、スワップパーティションを別に持つことがより理にかなっているかもしれませんが、SSDタイプのメディアでは、これは重要ではありません。しかし、それでも、より低速でおそらく安価なハードウェアにはスワップを置きたいと思うかもしれません。これは、パーティションでもファイルでも同じことで、より一般的な選択肢になりつつあります。

いくつかのディストリビューションでは、（オプションで）zram（www.kernel.org/doc/html/latest/admin-guide/blockdev/zram.html）を使用するようになっていきます。これは、スワップにディスクストレージを使用する代わりに、圧縮メモリを使用します。これはメモリ不足に陥りやすいですが、専門家の手にかかればパフォーマンスを向上させることができます。

22.8 パーティションテーブルのバックアップとリストア

パーティションテーブルのバックアップとリ

- マスタブートレコード

- **dd** プログラムを使用します :
- 以下のコマンドで**MBR**をバックアップします (パーティションテーブルも一緒にバックアップします)。

```
$ dd if=/dev/sda of=mbrbackup bs=512 count=1
```

- **MBR**は、以下の方法で復元することができます。

```
$ sudo dd if=mbrbackup of=/dev/sda bs=512 count=1
```

- ジーピーティー

- **MBR**システムでも使える**sgdisk**を使う

```
$ sudo sgdisk --backup-file=gptbackup /dev/sda
```

```
$ sudo sgdisk --load-backup-file=gptbackup /dev/sda
```


ディスクのパーティションテーブルを変更すると、すべてのファイルシステムのデータが消去される可能性があることを常に想定してください（そんなことはありませんが、慎重にしてください）。

したがって、この種の作業を行う前に、すべてのデータ（まだバックアップされていないもの）のバックアップを取ることが常に賢明です。**MBRシステム**の場合、**dd**はファイルの変換やコピーに使用することができます。**dd**の使用には注意が必要です：単純な入力ミスやオプションの誤用で、ディスク全体を破壊してしまう可能性があります。上記の**dd**コマンドはプライマリパーティションテーブルをコピーするだけで、他のパーティションに保存されているパーティションテーブル（拡張パーティションなど）には対応しません。

GPTシステムの場合、以下のように**sgdisk**ツールを使うのがベストです：

x8:/tmp>sudo sgdisk -p /dev/sda

```
ディスク /dev/sda: 1000215216 セクタ、476.9 GiB
Model : サムスンMZNLN512
セクタサイズ(論理/物理) : 512/512バイト
ディスクの識別子 (GUID) です : DBDBE747-8392-4B62-97AA-6214490073DC
パーティションテーブルは最大128エントリまで保持
----
  1  ---          2048          534527  260.0MiB  EF00 EFIシステムパーティ
  2          534528          ション 567295  16.0MiB 0C01 Microsoft
----
```

純粋な**MBRシステム**で実行する場合は注意してください：

c8:/tmp>sudo sgdisk -p /dev/sda

```
***** 無効なGPTと有効な
MBRが見つかり、メモリ内でMBRをGPT形式に変換しています。
***** ディスク /dev/sda: 500118192
セクター、238.5GiB
モデルです : クルーシャル_CT256MX1
セクタサイズ(論理/物理) : 512/4096バイト
----
```

22.9 パーティションテーブルエディタ

パーティションテーブルエディタ

- **fdisk** : 標準、メニュー式
- **sfdisk**: スクリプトまたはコマンドラインインターフェース
- **parted**GNU パーティション操作プログラム
- **gparted**です : **parted**のGUIベース版
- **gdisk**です : **GPT**システム用。**MBR**システムでも動作する。
- **sgdisk**: スクリプトまたはコマンドラインインターフェース

fdiskは、メニュー駆動型のパーティションテーブルエディタです。パーティションテーブルエディターの中で最も標準的で、最も柔軟なエディターの1つです。パーティションテーブルエディターと同様に、変更を加える前に、現在のパーティションテーブルの設定を書き留めるか、現在の設定のコピーを作成することを確認してください。

sfdiskコマンドは、Linuxベースの非インタラクティブなパーティションエディタプログラムであり、スクリプト作成に便利です。

sfdiskを使用する

を大切にしてください。

GPTシステムの場合、**gdisk**と**sgdisk**が同様の役割を果たします。両者とも**MBR**システムでも機能する。

parted は、GNU のパーティション操作プログラムです。パーティション（特定のファイルシステムを含む）の作成、削除、サイズ変更、移動が可能です。**parted** コマンドの GUI インターフェースは **gparted** です。

多くのLinuxディストリビューションには、CDROMやUSBメモリから実行できるライブインストール版があり、**gparted** のコピーが含まれているので、パーティショニングプログラムの実行中に実際には使用されていないディスクのグラフィカルなパーティショニングツールとして簡単に使用することができます。

22.10 エフディスク

fdiskを使う

- ベースインストールの一部で、簡単に使用できる
- メニューが豊富で、指示に従うだけでいい：
 - m:メニューを表示する。
 - p:パーティションテーブルを一覧表示します。
 - n:新しいパーティションを作成する。
 - d:パーティションを削除する。
 - t:パーティションタイプを変更する。
 - w:新しいパーティションテーブルの情報を書き込み、終了します。
 - q:変更を加えずに終了する。
- パーティションテーブルを編集する：
`$ sudo fdisk /dev/sda`
- パーティションテーブルを表示し、何もしない：
`$ sudo fdisk -l /dev/sda`

fdiskを実行するには、rootになる必要があります。扱いがやや複雑な場合があり、注意が必要です。**fdisk**のインターフェイスは、テキストメニュー駆動です。のように特定のディスクで起動した後

```
$ sudo fdisk /dev/sdb
```

という一文字のコマンドを発行することができます。

したがって、wでディスクに書き込む前に、パーティションテーブルが正しいかどうか（pで）確認することが重要であり、何か問題があれば、qで安全にジャンプアウトできます。

システムは、再起動するまで新しいパーティションテーブルを使用しません。ただし、次のコマンドを使用することができます：

```
$ sudo partprobe -s
```

を実行して、改訂されたパーティションテーブルを読み込んでみてください。しかし、これは常に確実に機能するわけではなく、新しいパーティションのフォーマットなどを行う前に再起動するのがベストです。パーティションが混ざってしまうと、致命的なことになりかねません。

いつでも、できる：

```
$ cat /proc/partitions
```

で、オペレーティングシステムが現在認識しているパーティションを調べることができます。

22.11 研究室



ビデオデモンストレーションのリソース

[fdisk_demo.mp4](#)

✎ エクササイズ22.0 ループバックデバイスを使用してパーティションをエミュレートする

このセクションの演習では、ファイルシステムをフォーマットするmkfsと、ルートファイルシステムツリーの任意の場所にマウントするmountを簡単に使用する。これらのコマンドについては、次のセッションで詳しく説明する予定です。

このセクションの最初の3つの演習では、パートDプログラムの有無にかかわらず、ループデバイス機構を使用します。



非常に重要

このコースの後の演習のために、パーティションなしのディスクスペースが必要です。1~2GBもあれば十分でしょう。

自分のネイティブマシンを使っている場合、それがあがるかどうかのどちらかです。ない場合は、パーティションとその上のファイルシステムを縮小し（最初に！）、gpartedや私たちが説明した、あるいはこれから説明する手順を使って、利用できるようにする必要があります。



ご注意ください

実際の物理的なパーティションなしのディスクスペースがある場合は、以下の手順を実行する必要はありませんが、それでも非常に有益な学習練習になります。

✎ 練習問題 22.1 : ファイルをディスクパーティションイメージとして使用する

この最初の演習では、ハードディスク全体のパーティションイメージのコンテナとして使用されるファイルを作成し、あらゆる目的で実際のハードパーティションのように使用できるようにします。次の演習では、このファイルに複数のパーティションを配置し、ディスク全体として動作させる方法を紹介します。

1. 長さ1GBのゼロだらけのファイルを作成します：

```
$ dd if=/dev/zero of=imagefile bs=1M count=1024
```

ファイルを作成するパーティションにそれほどの空き容量がない場合、もっと小さなファイルを作成することができます。

2. ファイルシステムを載せる：

```
$ mkfs.ext4 イメージファイル
```

```
mke2fs 1.42.9 (2013/12/28)
imagefileはブロックスペシャルデバイスでは
ありません。とにかく進めるか？(y,n) y
デバイスブロックの破棄：完了
-----
```

もちろん、`mkfs.ext3`、`mkfs.vfat`、`mkfs.xfs`など、異なるファイルシステムでフォーマットすることも可能です。

3. どこかに装着してください：

```
$ mkdir mntpoint
sudo mount -o loop イメージファイル mntpoint
```

これで、ファイルなどを置いて思う存分使えますね。

4. 終わったら、アンマウントしてください：

```
sudo umount mntpoint
```

ループオプションを使用してマウントする別の方法は、次のとおりです：

```
$ sudo losetup /dev/loop2 イメージファイル
$ sudo mount /dev/loop2 mntpoint
----
sudo umount mntpoint
$ sudo losetup -d /dev/loop2
```

losetupについては、この後の演習で説明します。また、`/dev/loop[0-7]`を使うこともできますが、これから説明するように、すでに使用されていないことに注意する必要があります。

実際のパーティションの代わりにループデバイスファイルを使用することは便利ですが、何らかの測定やベンチマークを行うにはかなり無価値であることに注意する必要があります。これは、ファイルシステムのレイヤーを別のレイヤーに重ねることになり、性能に悪影響を及ぼすだけだからです。また、ほとんどの場合、イメージファイルが作成されたファイルシステムの動作を利用だけです。

練習問題 22.2: ディスクイメージファイルのパーティション分割

次に複雑なのは、コンテナファイルを複数のパーティションに分割し、それぞれのパーティションをファイルシステムやスワップ領域として使用することです。

前回の演習で作成した画像ファイルを再利用するか、新たに作成することができます。

1. イメージファイルに対して**fdisk**を実行する：

```
$ sudo fdisk -C 130 イメージファイル
```

デバイスに認識されるパーティションテーブルがない
ディスク識別子 0x6280ced3 の新しい DOS ディスクラベルを構築する
。 fdisk (util-linux 2.23.2)へようこそ。

変更内容は、書き込みを決定するまで、メモリ上にのみ残ります。書き込みコマンドを使用する前に注意してください。

コマンド (mはヘルプ)：

C 130は、ドライブのインチキシリンダーの数を設定するもので、古いバージョンの**fdisk**にのみ必要です。

2. mを入力すると、コマンドの一覧が表示されます：

m

コマンド (mはヘルプ)：

m コマンドアクション

- a ブート可能なフラグをトグルする
- b edit bsd disklabel
- c dos 互換性フラグを切り替える
- d パーティションを削除する
- g 新しい空のGPTパーティションテーブルを作成する

```

G  IRIX (SGI) パーティションテーブルを作成する
I  リスト・ノウン・パーティション・タイプ
m  このメニューを印刷する
n  パーティションを追加する
o  空のDOSパーティションテーブルを新規に作成する
p  パーティションテーブルを印刷する
q  変更を保存せずに終了する
s  空のSun disklabelを新規に作成する
t  パーティションのシステムIDを変更する
u  チェンジディスプレイ/エ
   ントリーユニット V      パ
   ーティションテーブルを確認
   する
w  テーブルをディスクに書き込んで終了
x  エクストラファンクション (エキスパート専用)

```

コマンド (mはヘルプ) :

3. 新しいプライマリパーティションを作成し、256MB (または好きなサイズ) にします :

コマンド (mはヘルプ) : n

```

パーティションタイプです :
p  プライマリ (プライマリ0、エクステンド0
   、フリー4) e      ぜんめんてき
選択 (デフォルトp) : p
パーティション番号 (1~4、デフォルト1) : 1
第1セクタ (2048-2097151、デフォルト2048) :
デフォルト値2048を使用
最後のセクタ、セクタまたはサイズ{K,M,G} (2048-2097151、デフォルト2097151) :
+256M Linuxタイプでサイズ256MiBのパーティション1が設定されています。

```

4. 2番目のプライマリパーティションを256MBのサイズで追加します :

コマンド (mはヘルプ) : n

```

パーティションタイプです :
p  プライマリ (1プライマリ、0エクステンド
   、3フリー) e      ぜんめんてき
選択 (デフォルトp) : p
パーティション番号 (2~4、デフォルトは2) : 2
第1セクタ (526336-2097151、デフォルト526336) :
デフォルト値526336を使用
最後のセクタ、セクタまたはサイズ{K,M,G} (526336-2097151、デフォルト
2097151) : +256Mパーティション2 (タイプ : Linux、サイズ : 256MiB) が設定さ
れています。

```

コマンド (mはヘルプ) : p

```

ディスクのイメージファイルです : 1073 MB、1073741824バイト
、2097152セクタ 単位 = 1 * 512 = 512バイトのセクタ
セクタサイズ (論理/物理) : 512 バイト / 512 バイ
ト I/O サイズ (最小 / 最適) : 512バイト / 512バイ
ト ディスクラベルタイプ : dos

```

デバイスブー	スタート	BlocksId シ	
イメージファ	2048	526335	26214483 リ
イル1			ナックス
イメージファ	526336	1050623	26214483
イル2			リナックス

5. パーティションテーブルをディスクに書き込んで終了します：

コマンド (mはヘルプ) : w

パーティションテーブルが変更された！ディ
スクを同期しています。

これで練習はできましたが、作成した2つのパーティションを使用する方法はまだ見つかっていません。次の練習では、それを可能にするメソッドからやり直すことにしましょう。

練習問題 22.3: `losetup`と`parted`の使用

をもっと実験してみようと思っています：

- ループデバイスと`losetup`
- `parted`でコマンドラインから非介入でパーティションが切れるようになりました。

以下の手順を実行する前に、`losetup`と`parted`のマニュアルページを読んでおくことを期待します。もう一度言いますが、イメージファイルを再利用することもできますし、もっといいのは、それをゼロにして、新しく、

1. 画像ファイルをループデバイスに関連付ける：

```
$ sudo losetup -f
```

```
/dev/loop1
```

`sudo losetup /dev/loop1` イメージファイル

ここで、最初のコマンドは、最初に空いているループデバイスを見つけます。このようにする理由は、すでに1つまたは複数のループデバイスを使用している可能性があるからです。例えば、この文章が書かれているシステムで、上記のコマンドが実行される前：

```
$ losetup -a
```

```
/dev/loop0: []: (/usr/src/KERNELS.sqfs)
```

このコマンドの出力はディストリビューションによって異なります）これを無視して`/dev/loop0`に`losetup`を使用すると、ほぼ間違いなくファイルが破損してしまいます。

2. ループデバイス（イメージファイル）にディスクパーティションラベルを作成します：

```
$ sudo parted -s /dev/loop1 mklabel msdos
```

3. ループデバイスに3つのプライマリパーティションを作成します：

```
$ sudo parted -s /dev/loop1 unit MB mkpart primary ext4 0 256
$ sudo parted -s /dev/loop1 unit MB mkpart primary ext4 256 512
$ sudo parted -s /dev/loop1 unit MB mkpart primary ext4 512 1024
```

4. パーティションテーブルを確認します：

```
$ fdisk -l /dev/loop1
```

```
ディスク /dev/loop1: 1073 MB, 1073741824 bytes, 2097152 sectors
単位 = 1のセクター * 512 = 512 bytes
セクタサイズ（論理/物理）：512 バイト / 512 バイト
I/O サイズ（最小/最適）：512バイト / 512バイト
ト ディスクラベルタイプ：dos
```

デバイス	ブー	スター	終了	システム	BlockId	シ
デバイスの識別子です : 0x00050611						
/dev/loop1p1		500001	1000000	Linux	25000083	
/dev/loop1p2		500001	1000000	Linux	25000083	
/dev/loop1p3		1000001	2000000	Linux	50000083	

5. 次に何が起こるかは、使用しているディストリビューションに依存します。例えば、RHELとUbuntuでは、新しいデバイスノードが作成されたことがわかります :

```
$ ls -l /dev/loop1* . . .
```

```
brw-rw---- 1 root disk      7, 1 10月7日 14:54 /dev/loop1
brw-rw---- 1 root disk 259, 0 Oct 7 14:54 /dev/loop1p1
brw-rw---- 1 root disk 259, 3 Oct 7 14:54 /dev/loop1p2
brw-rw---- 1 root disk 259, 4 Oct 7 14:54 /dev/loop1p3
```

とし、以下ではこれらを使用します。

6. パーティションにファイルシステムを置く :

```
$ sudo mkfs.ext3 /dev/loop1p1
$ sudo mkfs.ext4 /dev/loop1p2
$ sudo mkfs.vfat /dev/loop1p3
```

7. 3つのファイルシステムをすべてマウントし、利用可能であることを表示します :

```
$ mkdir mnt1 mnt2 mnt3

$ sudo mount /dev/loop1p1 mnt1
$ sudo mount /dev/loop1p2 mnt2
$ sudo mount /dev/loop1p3 mnt3

$ df -Th
```

ファイルシステム	タイプ	サイズ	使用状況	使用率	搭載率
/dev/sda1	ext4	29G	8.5G	19G	32% /
----			2.1M	219M	1% mnt1
/dev/loop1p1	ext3	233M	2.1M	215M	1% mnt2
/dev/loop1p2	ext4	233M	0	489M	0% mnt3

8. ファイルシステムを思い存分使った後は、すべてを解き放つことができます :

```
$ sudo umount mnt1 mnt2 mnt3
$ rmdir mnt1 mnt2 mnt3
$ sudo losetup -d /dev/loop1
```

練習問題 22.4: 実物のハードディスクをパーティションで分割する

ハードディスクの未パーティション領域がある場合、**fdisk**を使って拡張パーティション内にプライマリまたは論理の新しいパーティションを作成する実験を行ってください。新しいパーティションテーブルをディスクに書き込み、新しいパーティションをフォーマットしてマウントします。

第23章

ファイルシステムの機能：属性、作成、チェック、 使用法、マウンティング



23.1	エクステンダトリビュート	364
23.2	ファイルシステムの作成と書式設定	365
23.3	ファイルシステムのトラブルシューティング	366
23.4	ファイルシステムのチェックと修復	368
23.5	ファイルシステムの使用方法	369
23.6	ディスク使用量	370
23.7	ファイルシステムのマウント	371
23.8	エヌエフエス	375
23.9	ブート時および/etc/fstab時のマウントについて	376
23.10	オートマウント	378
23.11	ネットワークブロックデバイス	380
23.12	研究室	384

学習目標

このセッションが終わるころには、次のことができるようになっているはずです：

- **lsattr**と**chattr**を使用した拡張ファイル属性の使用法について説明します。
- **mkfs**と**fsck**でファイルシステムを作成し、フォーマットする。
- ファイルシステムの問題をトラブルシュートする。
- ファイルシステムのエラーをチェックし、修正する。
- ディスクスペースの使用状況を確認するには、ユーティリティの**df**と**du**を使用します。
- ファイルシステムのマウントとアンマウントを行う。
- **NFS**などのネットワークファイルシステムを利用する。
- ファイルシステムを自動マウントする方法を理解し、ブート時にマウントされるようにする。
- ネットワークブロックデバイス（**NBD**）を理解し、使用する。

363

23.1 エクステンダトリビュート

lsattrとchattr

- ファイルに対して拡張属性を設定することができる
 - i: イミュータブルフラグ
 - a: アペンドオンリーフラグ
 - d: ダンプフラグなし
 - A: atimeアップデートフラグなし
- フラグはlsattrコマンドで見ることができます。
- フラグはchattrコマンドで設定します。
- フラグは、ファイルのinodeフラグに格納される
- ルートでのみ設定可能

拡張属性は、ファイルシステムが直接解釈しないメタデータをファイルに関連付けます。user、trusted、security、systemの4つの名前空間が存在する。system名前空間はアクセス制御リスト（ACL）に使用され、security名前空間はSELinuxで使用されます。

フラグ値はファイルinodeに保存され、rootユーザのみが変更および設定することができます。フラグの値は、lsattrで表示され、chattrで設定されます。フラグは、ファイルに対して設定することができます：

- i（不変）：変更（rootでも不可）、削除、リネームはできません。ハードリンクを作成することも、ファイルにデータを書き込むこともできません。この属性を設定または解除できるのは、スーパーユーザーだけです。
- a（アペンドオンリー）：書き込みのためにアペンドモードで開くことが可能です。この属性を設定または解除できるのは、スーパーユーザーのみです。
- d（ノーダンプ）：ダンププログラムの実行時に無視されます。スワップファイルやキャッシュファイルなど、バックアップに時間をかけたくない場合に有効です。
- A（atime更新なし）：ファイルはアクセスされたが、それ以外に変更されない場合、システムはファイルのatime（アクセスタイム）レコードを変更しない。ディスクI/Oの量が減るので、システムによっては性能が向上することがあります。

他にも設定できるフラグがあることに注意してください；man chattrと入力すると、すべてのリストが表示されます。chattrのフォーマットは次の通りです：

\$ chattr [+|-|=mode] ファイル名

lsattr は、ファイルの属性を表示するために使用します：

\$ lsattr ファイル名

23.2 ファイルシステムの作成と書式設定

マックフーズ

- **mkfs**の一般的なフォーマットです：
`mkfs [-t fstype] [オプション] [デバイス-ファイル]`。
- `[dev-file]`は通常、`/dev/sda3`や`/dev/vg/lvm1`などのデバイス名です。
- 各ファイルシステムタイプには、それぞれ固有のフォーマットオプションがあります。
- 各ファイルシステムには独自の**mkfs**プログラムがあります。
- 同等なコマンドです：
`$ sudo mkfs -t ext4 /dev/sda10`
`$ sudo mkfs.ext4 /dev/sda10`

すべてのファイルシステムタイプには、パーティション上でファイルシステムをフォーマット（作成）するためのユーティリティがあります。これらのユーティリティの総称は**mkfs**です。しかし、mkfsはファイルシステム固有のプログラムのフロントエンドに過ぎず、各プログラムは特定のオプションを持つことができます：

```
student@opensuse:~$ ls -lhF /sbin/mkfs*
lrwxrwxrwx 1 root root 14 Dec 16 16:41 /sbin/mkfs -> /usr/sbin/mkfs*
lrwxrwxrwx 1 root root 18 Dec 16 16:41 /sbin/mkfs.bfs -> /usr/sbin/mkfs.bfs*
lrwxrwxrwx 1 root root 20 Jan 29 01:15 /sbin/mkfs.btrfs -> /usr/sbin/mkfs.btrfs*
lrwxrwxrwx 1 root root 21 Dec 16 16:41 /sbin/mkfs.cramfs -> /usr/sbin/mkfs.cramfs*
lrwxrwxrwx 1 root root 19 Aug 29 2020 /sbin/mkfs.ext2 -> /usr/sbin/mkfs.ext2*
lrwxrwxrwx 1 root root 19 Aug 29 2020 /sbin/mkfs.ext3 -> /usr/sbin/mkfs.ext3*
lrwxrwxrwx 1 root root 19 Aug 29 2020 /sbin/mkfs.ext4 -> /usr/sbin/mkfs.ext4*
lrwxrwxrwx 1 root root 18 May 16 2020 /sbin/mkfs.fat -> /usr/sbin/mkfs.fat*
-rwxr-xr-x 2 root root 55K May 16 2020 /sbin/mkfs.jfs*
lrwxrwxrwx 1 root root 18 Dec 16 16:41 /sbin/mkfs.minix -> /usr/sbin/mkfs.minix*
lrwxrwxrwx 1 root root 20 May 16 2020 /sbin/mkfs.msdos -> /usr/sbin/mkfs.msdos*
-rwxr-xr-x 1 root root 28K May 16 2020 /sbin/mkfs.nilfs2*
lrwxrwxrwx 1 root root 16 May 16 2020 /sbin/mkfs.ntfs -> /usr/sbin/mkntfs*
lrwxrwxrwx 1 root root 10 May 16 2020 /sbin/mkfs.reiserfs -> mkreiserfs*
lrwxrwxrwx 1 root root 19 May 16 2020 /sbin/mkfs.vfat -> /usr/sbin/mkfs.vfat*
-rwxr-xr-x 1 root root 452K May 16 2020 /sbin/mkfs.xfs*
```


詳細は、各mkfs.*プログラムのmanページをご覧ください。

23.3 ファイルシステムのトラブルシューティング

ファイルの整合性をチェックする

- 設定ファイルやバイナリが破損していないか確認する
- 多数のメソッドを使用できる
 - 回転数
`$ rpm -V some_package`
をチェックするために、1つのパッケージ
`$ rpm -Va`
すべてのパッケージをチェックするために
 - **aide**: 侵入検知
`$ aide --check`
 - **debsums**です : **Debian**
`$ debsums options some_package`

破損した設定ファイルやバイナリをチェックする方法はいくつかあります。1つの方法は、特定のパッケージをチェックするために`rpm -V`コマンドを使用することです。また、`rpm -Va`を使用して、すべてのパッケージの完全性をチェックすることができます。

aideを使うのも、ファイルの変更をチェックする方法の一つです。`aide -check`というコマンドは、ファイルのスキャンを実行し、前回のスキャンと比較します。

Debian では、整合性チェックを行う唯一の方法は、**debsums** を使うことです。`debsums` を実行すると、そのパッケージのファイルのチェックサムをチェックします。しかし、すべてのパッケージがチェックサムを保持しているわけではないので、これはあまり役に立たないかもしれません。

ファイルシステムの破損と復旧

- fsckは、破損したファイルシステムを修正するために使用されることがあります。
- `etc/fstab`に設定ミスがないか確認する。
- ランニングが必要な場合があります
`$ mount -o remount,rw /` です。
を修正しようとする
- ファイルシステムを手動でマウントしてみる

fsckは修復を試みるために使用されるかもしれない。しかし、その前に `/etc/fstab` に設定ミスや破損がないことを確認する必要があります。実行中のカーネルが理解していないファイルシステムタイプに問題がある可能性があることにもう一度注意してください。

ルートファイルシステムがマウントされていれば、このファイルを調べることができますが、`/`は読み取り専用でマウントされている可能性があります。したがって、ファイルを編集して修正するには、次のように実行します：

```
$ sudo mount -o remount,rw /
```

を使用して、書き込み許可で再マウントします。

`etc/fstab`が正しいようであれば、`fsck`に移ることができます。まずは試してみてください：

```
$ sudo mount -a
```

を実行して、すべてのファイルシステムのマウントを試みます。これが完全に成功しない場合は、問題のあるファイルシステムを手動でマウントしてみることができます。その後、`fsck`を再度実行し、見つかったエラーを修正することができます。

23.4 ファイルシステムのチェックと修復

ファスニング

- 設定した期間またはマウント数の後に自動的に実行される
- 以前にクリーンマウントを解除していない場合に、ブート時に実行する。
- マウントされたファイルシステムで実行してはならない
- **fsck**の一般的なフォーマットです :
`fsck [-t fstype] [オプション] [デバイス-ファイル]`.
 ここで、[device-file]は通常、`/dev/sda3`や`/dev/ vg/lvm1`などのデバイス名です。
- ファイルシステムタイプは、パーティションのスーパーブロックを調べるため、通常は必要ありません。
- 同等なコマンドです :
`$ sudo fsck -t ext4 /dev/sda10`
`$ sudo fsck.ext4 /dev/sda10`

すべてのファイルシステムタイプには、エラーをチェックする（そしてできれば見つけたエラーを修正する）ために設計されたユーティリティがあります。これらのユーティリティの総称は**fsck**です。しかし、これはファイルシステムに特化したプログラムのフロントエンドに過ぎない :

```
student@opensuse:~$ ls -lhF /sbin/fsck*
lrwxrwxrwx 1 root root 14 Dec 16 16:41 /sbin/fsck -> /usr/sbin/fsck*
lrwxrwxrwx 1 root root 20 Jan 29 01:15 /sbin/fsck.btrfs -> /usr/sbin/fsck.btrfs*
lrwxrwxrwx 1 root root 21 Dec 16 16:41 /sbin/fsck.cramfs -> /usr/sbin/fsck.cramfs*
lrwxrwxrwx 1 root root 19 Aug 29 2020 /sbin/fsck.ext2 -> /usr/sbin/fsck.ext2*
lrwxrwxrwx 1 root root 19 Aug 29 2020 /sbin/fsck.ext3 -> /usr/sbin/fsck.ext3*
lrwxrwxrwx 1 root root 19 Aug 29 2020 /sbin/fsck.ext4 -> /usr/sbin/fsck.ext4*
lrwxrwxrwx 1 root root 18 May 16 2020 /sbin/fsck.fat -> /usr/sbin/fsck.fat*
-rwxr-xr-x 2 root root 407K May 16 2020 /sbin/fsck.jfs*
lrwxrwxrwx 1 root root 20 Dec 16 16:41 /sbin/fsck.minix -> /usr/sbin/fsck.minix*
lrwxrwxrwx 1 root root 20 May 16 2020 /sbin/fsck.msdos -> /usr/sbin/fsck.msdos*
lrwxrwxrwx 1 root root 10 May 16 2020 /sbin/fsck.reiserfs -> reiserfsck*
lrwxrwxrwx 1 root root 19 May 16 2020 /sbin/fsck.vfat -> /usr/sbin/fsck.vfat*
-rwxr-xr-x 1 root root 433 May 16 2020 /sbin/fsck.xfs*
```

図23.2 : fsck

を行うことで、起動時にマウントされたすべてのファイルシステムを強制的にチェックすることができます :

```
$ sudo touch /forcefsck
sudo reboot
```

チェックが成功すると、`/forcefsck`は消えます。

見つかったエラーは、`-t` オプションでひとつひとつ手動で修正するか、`-a` オプションで可能な限り自動的に修正するかを制御できる。さらに、ファイルシステムの種類によって、独自の **fsck** オプションがある場合もあります。

23.5 ファイルシステムの使用法

df : ファイルシステムの使用状況

- **df** (disk free) は、ファイルシステムの使用状況を見るために使用されます。
- ファイルシステムの使用量を表示します（単位：Kバイト-デフォルト）：
`$ df`
- ファイルシステムの使用状況を人間が読める形式で表示します：
`$ df -h`
- ファイルシステムのタイプ形式を表示する：
`$ df -T`
- inode情報を表示する：
`$ df -i`

```
x8:/tmp>df -hT
Filesystem      Type      Size  Used Avail Use% Mounted on
devtmpfs        devtmpfs  7.8G   0    7.8G   0% /dev
tmpfs           tmpfs     7.8G   0    7.8G   0% /dev/shm
tmpfs           tmpfs     7.8G  9.6M  7.8G   1% /run
tmpfs           tmpfs     7.8G   0    7.8G   0% /sys/fs/cgroup
/dev/sda8       ext4       26G   10G   15G   42% /
/dev/sda7       vfat      200M   14M   187M    7% /boot/efi
/dev/sda5       ext4       15G   9.6G   4.1G   71% /UBUNTU
/dev/sda6       ext4      389G  252G  118G   69% /ALL
/dev/sda1       vfat      256M   43M   214M   17% /Cbootefi
tmpfs           tmpfs     1.6G   1.2M   1.6G    1% /run/user/42
tmpfs           tmpfs     1.6G   4.0K   1.6G    1% /run/user/1000
/dev/loop0     squashfs  2.5G   2.5G    0  100% /usr/src/KERNELS
x8:/tmp>
```

図23.3 : dfの使用

23.6 ディスク使用量

du : ディスク使用量

- du (ディスク使用量) は、ディスク容量と使用量の両方を見るために使用されます
- 現在のディレクトリのディスク使用量を表示する :
デュ
- ディレクトリだけでなく、すべてのファイルをリストアップします :
\$ du -a
- 人間が読める形式のリスト :
\$ du -h
- 特定のディレクトリのディスク使用量を表示します :
\$ du -h somedir

以下のコマンドを試してみてください :

```
$ find . -maxdepth 1 -type d -exec du -shx {} \; | sort -hr
```

```
c8:/home/coop/.cache>find . -maxdepth 1 -type d -exec du -shx {} \; | sort -hr
338M    .
229M    ./google-chrome
86M     ./spotify
7.7M    ./tracker
6.8M    ./thunderbird
4.9M    ./gnome-software
2.5M    ./mesa_shader_cache
956K    ./gstreamer-1.0
436K    ./samba
52K     ./evolution
12K     ./fontconfig
8.0K    ./vmware
8.0K    ./Slack
8.0K    ./skypeforlinux
4.0K    ./libgweather
4.0K    ./googleearth_CACHE
4.0K    ./gnome-shell
4.0K    ./gnome-screenshot
c8:/home/coop/.cache>
```

図23.4 : duの使用

23.7 ファイルシステムのマウント

ファイルシステムのマウント

- 各ファイルシステムは、特定のディレクトリの下にマウントされます：

`マウント -t ext4 /dev/sdb4 /home`

- **ext4**ファイルシステムをマウントする
- 通常、`-t`オプションで型を指定する必要はありません。
- ファイルシステムは、ハードディスクの特定のパーティション (`/dev/sdb4`) に配置されている
- ファイルシステムは、現在のディレクトリツリーの `/home` の位置にマウントされています。
- 元の `/home` ディレクトリに存在するファイルは、パーティションがアンマウントされるまで非表示になる

Linuxでアクセス可能なファイルはすべて、ツリーの先頭をルートディレクトリ(`/`)とする1つの大きな階層的なツリー構造に編成されています。しかし、同じファイルシステムツリーの中に、複数のパーティション（それぞれが独自のファイルシステムタイプを持つことができる）が一緒に結合されていることが一般的である。これらのパーティションは、異なる物理デバイス上、さらにはネットワーク上にも存在することがあります。

`mount`プログラムは、ツリー構造上の任意の位置でアタッチすることができ、`umount`はそれらをデタッチすることができます。

マウントポイントは、ファイルシステムがアタッチされるディレクトリです。`mkdir`は、空のディレクトリを作成するために使用します。既存のディレクトリが使用され、マウントポイントとして使用される前のファイルが含まれている場合、マウント後にそれらのファイルは隠されます。これらのファイルは削除されず、ファイルシステムがアンマウントされると再び表示されます。

ファイルシステムのマウントとアンマウントは、スーパーユーザーのみ可能です。

据え付ける

- マウントの一般的なフォーム

`mount [options] <source> <director` y

- を使用してマウントすることができます：デバイスノード、ラベル、UUID

- 各ファイルシステムには、それぞれ独自のオプションがあります。一般的な例では、次のようになります：

`$ sudo mount -o remount,ro /myfs`

ファイルシステムを読み取り専用で再マウントする。

```
File Edit View Search Terminal Help
c:/tmp> mount --help
Usage:
mount [-hV]
mount -a [options]
mount [options] [--source] <source> | [--target] <directory>
mount [options] <source> <directory>
mount <operation> <mountpoint> [<target>]

Options:
-a, --all                mount all filesystems mentioned in fstab
-c, --no-canonicalize    don't canonicalize paths
-f, --fake               dry run; skip the mount(2) syscall
-F, --fork               fork off for each device (use with -a)
-T, --fstab <path>      alternative file to /etc/fstab
-h, --help               display this help text and exit
-i, --internal-only      don't call the mount.<type> helpers
-l, --show-labels        lists all mounts with LABELs
-n, --no-mtab            don't write to /etc/mtab
-o, --options <list>     comma-separated list of mount options
-O, --test-opts <list>   limit the set of filesystems (use with -a)
-r, --read-only          mount the filesystem read-only (same as -o ro)
-t, --types <list>       limit the set of filesystem types
--source <src>           explicitly specifies source (path, label, uuid)
--target <target>        explicitly specifies mountpoint
-v, --verbose            say what is being done
-V, --version            display version information and exit
-w, --rw, --read-write   mount the filesystem read-write (default)

-h, --help              display this help and exit
-V, --version            output version information and exit
...
c:/tmp>
```

図23.5：マウント

以下の実装方法は、すべて同等です：

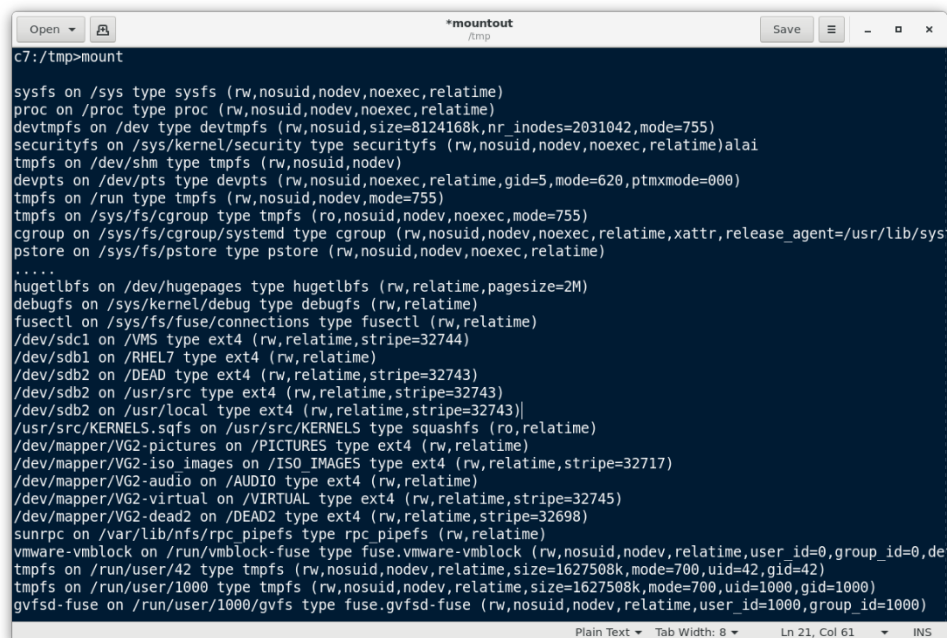
```
$ sudo mount /dev/sda2 /home
$ sudo mount LABEL=home /home
$ sudo mount-L home /home
sudo mount UUID=26d58ee2-9d20-4dc7-b6ab-aa87c3cfb69a /home
sudo mount-U 26d58ee2-9d20-4dc7-b6ab-aa87c3cfb69a /home
```

ラベルはe2labelなどのファイルシステムタイプに特化したユーティリティで割り当てられ、UUIDはファイルシステムのコンテナとしてパーティションを作成し、mkfsでフォーマットする際に割り当てられる。

デバイスを指定するこれらの3つの方法はいずれも使用可能ですが、最近のシステムでは、システムの起動方法、どのハードディスクが最初に見つかるかなどによって名前が変わる可能性があるため、デバイスノードの形式を使用することは推奨されません。

ラベルは改善されましたが、まれに同じラベルを持つ2つのパーティションが存在することがあります。しかし、UUIDは常にユニークであるべきで、パーティション作成時に作成されます。

現在マウントされているファイルシ



```
c7:/tmp>mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
devtmpfs on /dev type devtmpfs (rw,nosuid,size=8124168k,nr_inodes=2031042,mode=755)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,nodev,mode=755)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=755)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noexec,relatime,xattr,release_agent=/usr/lib/sys
pstore on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relatime)
.....
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,pagesize=2M)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
fusectl on /sys/fs/fuse/connections type fusectl (rw,relatime)
/dev/sdcl on /VMS type ext4 (rw,relatime,stripe=32744)
/dev/sdb1 on /RHEL7 type ext4 (rw,relatime)
/dev/sdb2 on /DEAD type ext4 (rw,relatime,stripe=32743)
/dev/sdb2 on /usr/src type ext4 (rw,relatime,stripe=32743)
/dev/sdb2 on /usr/local type ext4 (rw,relatime,stripe=32743)
|
/usr/src/KERNELS.sqfs on /usr/src/KERNELS type squashfs (ro,relatime)
/dev/mapper/VG2-pictures on /PICTURES type ext4 (rw,relatime)
/dev/mapper/VG2-iso_images on /ISO_IMAGES type ext4 (rw,relatime,stripe=32717)
/dev/mapper/VG2-audio on /AUDIO type ext4 (rw,relatime)
/dev/mapper/VG2-virtual on /VIRTUAL type ext4 (rw,relatime,stripe=32745)
/dev/mapper/VG2-dead2 on /DEAD2 type ext4 (rw,relatime,stripe=32698)
sunrpc on /var/lib/nfs/rpc pipefs type rpc pipefs (rw,relatime)
vmware-vmblock on /run/vmblock-fuse type fuse.vmware-vmblock (rw,nosuid,nodev,relatime,user_id=0,group_id=0,de
tmpfs on /run/user/42 type tmpfs (rw,nosuid,nodev,relatime,size=1627508k,mode=700,uid=42,gid=42)
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,size=1627508k,mode=700,uid=1000,gid=1000)
gvfsd-fuse on /run/user/1000/gvfs type fuse.gvfsd-fuse (rw,nosuid,nodev,relatime,user_id=1000,group_id=1000)
```

図 23.6 : 現在マウントされているファイルシステム

ユーマウント

- ファイルシステムのアンマウントに使用
- **umount**の一般的な形式です：
`umount [デバイスファイル | マウントポイント]`。
- 例を挙げます：
 - **home** ファイルシステムをアンマウントします：
`$ umount /home`
 - **dev/sda3** デバイスをアンマウントします：
`$ umount /dev/sda3`

ファイルシステムをアンマウントするコマンドは、**umount** (**unmount**ではありません！) であることに注意してください。**マウント**と同様に、**umount**にも多くのオプションがあり、その多くはファイルシステムタイプに特化しています。もう一度言いますが、特定のオプションの情報については、**マニュアルページ**が最良のソースです。

ファイルシステムをアンマウントする際に遭遇する最も一般的なエラーは、現在使用中のファイルシステムでアンマウントしようとした場合です。つまり、ファイルシステム内のファイルやその他のエントリーを使用しているアプリケーションが現在存在している場合です。

これは、マウントされたファイルシステム上のディレクトリでターミナルウィンドウを開いているのと同じくらい簡単なことです。そのウィンドウで**cd**を使うか、終了させるだけで、**device is busy**エラーは解消され、アンマウントが可能になります。

しかし、このエラーを引き起こす他のプロセスがある場合は、ファイルシステムをアンマウントする前に、そのプロセスを終了させる必要があります。**fuser**を使って、ファイルシステムを使用しているユーザーを見つけ、殺すことができます(これには注意が必要です。最初にユーザーに警告を出すとよいでしょう)。また、**lsof** ("list open files") を使って、どのファイルが使用されていて、アンマウントをブロックしているかを調べることもできます。

23.8 エヌエフエス

ネットワークシェア(NFS)

- 最も一般的なネットワーク共有は**NFS**
- その他、**AFS**、**SMB (CIFS)** などがあります。
- のようにマウントします：

```
$ sudo mount -t nfs myserver.com:/shdir /mnt/shdir
```
- この行を **/etc/fstab** に記述して、起動時または **mount -a** でマウントします
：

```
myserver.com:/shdir /mnt/shdir nfs rsize=8192,wsiz=8192,timeo=14,intr 0 0
```
- ネットワークがアップする前に、システムが**NFS**ファイルシステムをマウントしようとする場合があります。netdevとnoautoオプションを使用することができます (man nfs, examine **mount options**)。
- **autofs**または**automount**を使用しても解決可能です。
- **mount**には100万個のオプションがあり、いくつかは**nfs**に特有のもの

リモートファイルシステムをネットワーク共有でマウントすることで、あたかもローカルマシンにあるかのように見せることはよくあることです。歴史的に最も一般的な方法は、**NFS (Network File System)** でしょう。

NFSは1989年に**Sun Microsystems**社によって開発され、その後も継続的に更新されています。最近のシステムでは2000年以降、継続的に更新されている**NFSv4**。

ネットワークファイルシステムには、他に**AFS (Andrew File System)**、**SMB (Server Message Block)** があり、**CIFS**とも呼ばれる。

ネットワークファイルシステムは、ネットワーク共有に存在しないか、ネットワークが利用できないかのどちらかで、いつでも利用できなくなる可能性があるため、システムはこの可能性に備えておく必要があります。

したがって、このような状況では、指定された期間より長く待っている間、システムがハングアップ、つまりブロックされないように指示する必要があります。これらは、**mount**コマンドまたは**/etc/fstab**で指定することができます。

23.9 ブート時および/etc/fstab時のマウントについて

ブーツに装着する

- システムの初期化時に、コマンドを実行します：

```
$ mount -a
```

を実行すると、[/etc/fstab](#)に記載されているすべてのファイルシステムをマウントすることができます。

- 起動時にマウントするファイルシステムをローカルおよびリモートのネットワークマウントで指定可能

[/etc/fstab](#) は、起動時にマウント可能なファイルシステムおよびデバイスを定義するために使用されます。マウントできるファイルシステム、ローカルにあるファイルシステム、誰がどのような権限でマウントできるかを確認するために、ここを参照してください。ファイルやディレクトリが[/etc/fstab](#)に指定されている場合、起動時にマウントされることがあります。

これには、NFSやsambaファイルシステムなど、ローカルおよびリモートのネットワークマウントされたファイルシステムの両方が含まれる場合があります。

/etc/fstab

- ファイルシステム、その標準マウントポイント、およびマウント時に使用すべきオプションに関する情報が含まれています。
- ファイル内の各レコードは、マウントするファイルシステムに関する情報を空白で区切ったフィールドで構成されています。
 - デバイスファイル、ラベル、またはUUID
 - マウントポイント
 - ファイルシステムタイプ
 - カンマで区切られたオプションリスト
 - ダンプ周波数(またはゼロ)
 - `fsck`パス番号 (または0)。
- `mount`コマンドと`umount`コマンドは、`/etc/fstab`の情報を利用することができます。

このファイルの各レコードには、ブート時にマウントされるファイルシステムに関する情報が含まれています。レコードの最初の項目は、デバイスファイル名 (`/dev/sda1`など)、ラベル、またはUUIDのいずれかである。続いて、ファイルシステムのマウントポイント (ツリー構造のどこに挿入されるか) です。続いて、ファイルシステムのタイプ、カンマで区切られたオプションのリスト、ダンプ回数 (`dump -w`コマンドで使用) またはゼロ (`dump`で無視)、`fsck`パス番号またはゼロ (このパーティションを`fsck`しないことを意味) が続きます。

```
x8:/tmp> cat /etc/fstab
#
# /etc/fstab
# Created by anaconda on Tue Aug 27 15:29:19 2019
#
# Accessible filesystems, by reference, are maintained under '/dev/disk/'.
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info.
#
# After editing this file, run 'systemctl daemon-reload' to update systemd
# units generated from this file.
#
UUID=53ea9807-fd58-1234-9460-d03ec36f73a3 /      ext4    defaults 1 1
UUID=29C4-7777 /boot/efi vfat    umask=0077,shortname=winnt 0 2
LABEL=ALL /ALL      ext4    defaults 1 2
LABEL=UBUNTU /UBUNTU   ext4    defaults 1 2
/ALL/usr/local /usr/local none    bind 0 0
/ALL/usr/src /usr/src  none    bind 0 0
/ALL/VMS /VMS      none    bind 0 0
/ALL/RESOURCES /RESOURCES none    bind 0 0
LABEL=Sam128 /SAM      ext4    noauto 0 0
/usr/src/KERNELS.sqfs /usr/src/KERNELS squashfs loop 0 0
/dev/sda1 /Cbootefi vfat    umask=0077,shortname=winnt 0 2
x8:/tmp>
```

図23.7 : `/etc/fstab`の例

23.10 オートマウント

ファイルシステムの自動マウント

- ファイルシステムを使用時に利用可能にする（**mount**）、アイドル時に削除する（**umount**）。
- **autofs**です：
 - 長年にわたり利用可能、
 - **autofs**パッケージのインストールが必要です。
 - 柔軟性があり、**/etc**の設定ファイル
- オートマウント
 - **systemd**の一部
 - **etc/fstab**を修正し、以下のように**systemd**を再起動またはリブートします：
`sudo systemctl daemon-reload`
`sudo systemctl restart local-fs.target`

Linuxシステムには、必要なときだけファイルシステムをマウントする機能が以前からあります。歴史的には、これは **autofs** を使って行われていました。このユーティリティを使用するには、適切なパッケージ・マネージャーを使用して **autofs** パッケージをインストールし、以下のファイルを設定する必要があります。

/etc/etor

autofsは非常に柔軟でよく理解されていますが、**systemd**ベースのシステム（すべてのエンタープライズLinuxディストリビューションを含む）には、**systemd**フレームワークに組み込まれた**オートマウント機能があります**。これを設定するには、**/etc/fstab** に適切なデバイス、マウントポイント、およびマウントオプションを指定する行を追加するだけで、次のよ

```
LABEL=Sam128 /SAM ext4 noauto,x-systemd.automount,x-systemd.device-timeout=10,x-systemd.idle-timeout=30  
→ 0 0
```

うに簡単に設定できます：

を実行し、再起動するか、コマンドを発行するかしてください：

```
sudo systemctl daemon-reload  
sudo systemctl restart local-fs.target
```

次のページでは、例を挙げて選択肢を説明します。

automount 例

```

File Edit View Search Terminal Help
x7:/tmp>grep automount /etc/fstab
LABEL=Sam128          /SAM          ext4    noauto,x-systemd.automount,x-systemd.device-tim
eout=10,x-systemd.idle-timeout=30 0 0
x7:/tmp>df -h | grep SAM
x7:/tmp>ls /SAM
ISO_IMAGES  lost+found  VIRTUAL_MACHINE_IMAGES
x7:/tmp>df -h | grep SAM
/dev/sdb1    ext4          118G    71G    41G    64% /SAM
x7:/tmp>sleep 40
x7:/tmp>df -h | grep SAM
x7:/tmp>

```

図23.8 : automountの例

上記の例では、常にシステムに接続されているUSBペンドライブを、使用するときだけマウントしています。[etc/fstab](#)のオプションです：

- ノーオート
ブート時にマウントしない。ここでいうautoとは、自動マウントのことではありません。
- x-systemd.automount
systemdの自動マウント機能を利用する
- x-systemd.automount.device-timeout=10
このデバイスが利用できない場合、例えばNFSでアクセスできるネットワークデバイスの場合、ハングアップするのではなく、10秒後にタイムアウトします。
- x-systemd.automount.idle-timeout=30
30秒間使用されない場合は、アンマウントしてください。

ブート中にデバイスをマウントしても、指定したタイムアウト後にアンマウントする必要があることに注意してください。スクリーンショットは、デバイスが一度使用された後のみ利用できることを表しています。

23.11 ネットワークブロックデバイス

NBDの紹介

- ブロックデバイスをソースコンピュータ（サーバー）からターゲットコンピュータ（クライアント）にエクスポートするために設計されたLinuxプロトコル。
- **NBD**は、通信に**Unixソケット**または**TCP/IP**を使用することができます。
- サーバーからエクスポートされる単位は、1つ以上のファイル、イメージ、またはブロックデバイスであることができます。
- クライアント側では、サーバーから提示されたデータプロブは、**nbd**カーネルモジュールを介してマッピングされ、ブロックデバイスとしてアクセスされる。
- クライアント側のブロックデバイスは、以下のような名前で認識することができます：
：
`/dev/nbd0/`、`/dev/nbd1`など。
- 最も単純な構成では、データストリームは暗号化されませんが、暗号化は可能であり、NBD仕様の一部です。

一部の追加情報および参考資料は、以下の場所でご覧いただけます：**NBD**プロトコル情報:

<https://github.com/NetworkBlockDevice/nbd/blob/master/doc/proto.md>

NBD用Userlandクライアント・サーバアプリケーション

<https://github.com/NetworkBlockDevice/nbd>

ネットワークブロック装置

nbdのクライアントとサーバーのペアを構成するために、一般的な手順は次のとおりです：

- サーバーでエクスポートするものを定義する
- サーバーに共有する項目を定義する
- クライアントを接続する
- **dev/nbd0**、**/dev/nbd1**デバイスを使用することで、他のブロックデバイスと同様にパーティションやフォーマットすることができます。
- パーティションで区切られた**nbd**デバイスでは、ほぼすべてのファイルシステムタイプが使用可能です

今回使用するステップは

1. **dd**を使用して、空のファイルを作成します。
2. サーバーで共有する項目を設定ファイルに定義する。
3. **nbd**カーネルモジュールを有効化する。
4. エクスポート名、IPアドレス、ポートを使って、クライアントでサーバーに問い合わせる
5. **nbd-client**コマンドでローカルの**/dev/nbd**ブロックドライバとサーバを関連付ける。
6. **fdisk**で**nbd**をパーティション分割する。
7. **nbd**にファイルシステムを追加してマウントする。

NBDユーザーユーティリティ

nbdのクライアントやサーバーのパッケージは、以下のようなものが用意されています：

- nbdkitです： CentOSです、 Fedoraです、 Debianです、
、 ウブントゥ
- nbd-clientとnbd-serverです： Ubuntuです、 デビアン
- nbd (GitHubより) です： CentOSです、 Fedoraです、
Debianです、 ウブントゥ
- xNBD-clientとxNBD-serverです： Debian
- qemu-imgです： CentOS

一般的にクライアントとサーバーは混在可能なので、ユースケースで慎重にテストすることをお勧めします。

ubuntu22.04のユーザーユーティリティと管理者ユーティリティの例です：

- nbd-server.confは、含むサーバーの設定ファイルの一例です：
 - 接続を待機するIPアドレスとポート
 - ディスクとして書き出す記憶装置
 - いくつかのオプションの制御要素
- nbd-serverは、接続要求や通信に応答するサーバー側のコンポーネントです。
- nbd-server man ページには、サーバーの設定に関する詳細が記載されています。この情報は、ディストリビューションによって異なる場合があります。
- nbd-clientは、サーバーへの問い合わせと接続を行うために使用されます。
- nbd-client man pageには、サーバーへの接続を行うためのクライアント関連情報が記載されています。

これらのユーティリティは、ディストリビューションによってどのようにパッケージ化されるかによって、異なる名前を持ち、異なる機能を含む場合があります。

ネットワークブロックデバイスの例

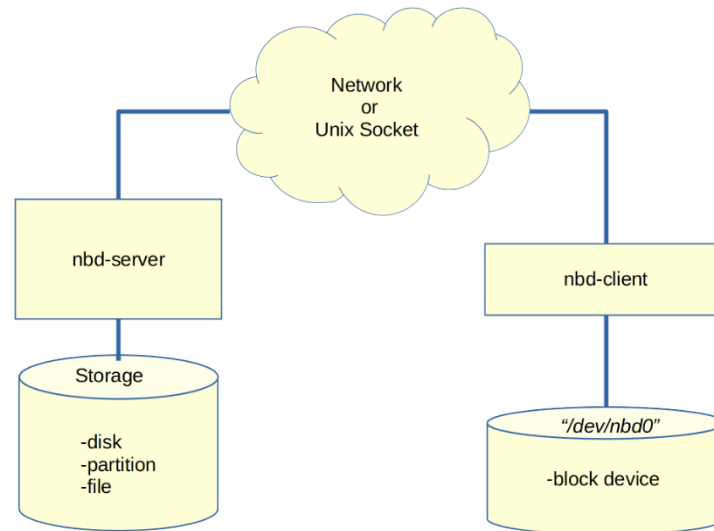


図23.9 : ネットワークブロックデバイス

クライアントとサーバーのコマンドの例をいくつか紹介します。

サーバーは**CentOS-8-Stream**で、**GitHub**の**nbd**パッケージを使用しました。クライアントは**CentOS-8-Stream**で、**GitHub**の**nbd**パッケージを使用していました。

- nbdカーネルモジュールがロードされていることを確認する。
`$ sudo modprobe -i nbd`
- 192.168.242.160のexport fooとローカルデバイス/dev/nbd10を接続します。
`$ sudo nbd-client -N foo 192.168.242.160 /dev/nbd10`

Ubuntuインストール時のコマンドの例です：

- nbdサーバーのプロセスを開始する
`$ sudo nbd-server -C nbd-server.conf`
- クライアントからサーバー上のエクスポートをリストアップする
`$ sudo nbd-client -l 127.0.0.1 10042`
- エクスポート「foo」をローカルデバイス/dev/nbd0に接続する。
`$ sudo nbd-client -N foo 127.0.0.1 10042 /dev/nbd0`

23.12 研究室



ビデオデモンストレーションのリソース

ファイルシステムを使ったデモ.mp4

練習問題 23.1 : ファイル属性の操作

1. 通常のユーザーアカウントで**touch**を使い、**/tmp/appendit**という名前の空のファイルを作成する。
2. **cat**を使って、**/etc/hosts**の内容を**/tmp/appendit**に追記します。
3. **tmp/appendit**と**/etc/hosts**の内容を比較し、相違がないことを確認します。
4. **chattr**で**/tmp/appendit**にappend-only属性を付けてみてください。ここでエラーが表示されるはずです。なぜでしょうか？
5. rootとして、append-only属性の追加を再試行する。今回はうまくいくはずだ。このファイルの拡張属性を見てみましょう。
lsattr.
6. 通常のユーザーとして、**cat**を使って**/etc/passwd**の内容を**/tmp/appendit**にコピーしてみてください。エラーが出るはずです。なぜでしょうか？
7. もう一度rootで同じことを試してみてください。エラーも出るはずです。なぜでしょうか？
8. 通常ユーザーとして、再びappendリダイレクト演算子(>>)を使用し、**/etc/passwd**ファイルを**/tmp/appendit**に追加してみてください。これでうまくいくはずだ。出来上がったファイルを調べて確認してください。
9. rootとして、**/tmp/appendit**にimmutable属性を設定し、再度拡張属性を見えます。
10. 出力を**/tmp/appendit**に追加してみる、ファイル名を変更してみる、ファイルへのハードリンクを作成してみる、通常ユーザーとrootの両方でファイルを削除してみる。
11. 拡張属性を削除することで、このファイルを削除することができます。そうしてください。

✓ 解答 23.1

1.

```
$ cd /tmp
$ touch appendit
$ ls -l appendit
```

```
-rw-rw-r-- 1 coop coop 0 Oct 23 19:04 appendit
```

2.

```
$ cat /etc/hosts > appendit
```
3.

```
$ diff /etc/hosts appendit
```
4.

```
$ chattr +a appendit
```

chattr : appendit でフラグを設定する際に許可されない操作

5.

```
sudo chattr +a appendit
$ lsattr appendit
```

```
-----a-----e-- appendit
```

6. `$ cat /etc/passwd > appendit`


```
bash: appendit : 操作が許可されていない
```

7. `$ sudo su`

```
$ cat /etc/passwd > appendit
```

```
bash: appendit : 操作が許可されていない
```

`$ 出口`

8. `$ cat /etc/passwd >> /tmp/appendit`

```
$ cat appendit
```

9. `sudo chattr +i appendit`

```
$ lsattr appendit
```

```
----ia-----e- appendit
```

10. `$ echo hello >> appendit`

```
-bash: appendit : パーミッションが拒否されました
```

```
mv appendit appendit.rename
```

```
mv: `appendit' を `appendit.rename' に移動することができません : 操作が許可されていない
```

```
$ ln appendit appendit.hardlink
```

```
ln: ハードリンク `appendit.hardlink' => `appendit' を作成します : 操作が許可されていない
```

```
$ rm -f appendit
```

```
rm: `appendit' を削除できない : 操作が許可されていない
```

```
$ sudo su
```

```
$ echo hello >> appendit
```

```
-bash: appendit : パーミッションが拒否されました
```

```
mv appendit appendit.rename
```

```
mv: `appendit' を `appendit.rename' に移動することができません : 操作が許可されていない
```

```
$ ln appendit appendit.hardlink
```

```
ln: ハードリンク `appendit.hardlink' => `appendit' を作成します : 操作が許可されていない
```

\$ rm -f appendit

rm: `appendit' を削除できない : 操作が許可されていない

\$ 出口

11. \$ sudo su

\$ lsattr appendit

----ia-----e- appendit

\$ chattr -ia appendit

\$ rm appendit

rm: 正規のファイル `appendit' を削除するか？

\$ ls appendit

ls: appenditにアクセスできません : そのようなファイルまたはディレクトリはありません

演習23.2: マウントのオプション

フレッシュパーティションまたはループバックファイル

この演習では、新しいパーティションを作成するか、ループバック ファイルを使用する必要があります。解決方法は若干異なりますが、両方の方法について詳細を説明します。

1. **fdisk**を使って、システム上に新しい250MBのパーティションを作成します（おそらく `/dev/sda` ）。あるいは、ゼロでいっぱいファイルを作って、新しいパーティションをシミュレートするためのループバック・ファイルとして使う。
2. **mkfs**を使用して、作成したパーティションまたはループバックファイルに新しいファイルシステムをフォーマットします。これを3回行い、その都度ブロックサイズを変更します。各ケースで、スーパーブロックの位置、ブロックグループの数、その他の関連情報を記録してください。
3. 新しいサブディレクトリ（ `/mnt/tempdir` とする ）を作成し、この場所に新しいファイルシステムをマウントします。マウントされたことを確認します。
4. 新しいファイルシステムのマウントを解除し、読み取り専用として再マウントします。
5. マウントされたディレクトリにファイルを作成してみてください。ここでエラーが出るはずですが、なぜですか？
6. 再度ファイルシステムのアンマウントを行います。
7. ブート時にファイルシステムがマウントされるように、 `/etc/fstab` ファイルに行を追加します。
8. ファイルシステムをマウントします。
9. バイナリファイルをファイルシステムから実行できないように、新しいファイルシステムの設定を変更します（ `/mnt/tempdir` エントリーのデフォルトを `noexec` に変更します ）。次に、ファイルシステムを再マウントし、実行可能ファイル（たとえば `/bin/ls` ）を `/mnt/tempdir` にコピーして実行してみてください。エラーが出るはずですが、なぜ？

終了後、[/etc/fstab](#)からエントリを削除することでクリーンアップすることができます。



解答 23.2



物理的パーティションソリューション

1. **fdisk**の詳細な手順は、先に説明したとおりですので、ここでは紹介しません。ここでは、作成されたパーティションが `/dev/sda11`、何か見せたいから。

```
$ sudo fdisk /dev/sda
```

```
-----
w
$ partprobe -s
```

パートプローブが機能しないこともあり、システムが新しいパーティションを知っていることを確認するために、再起動が必要です。

2.

```
$ sudo mkfs -t ext4 -v /dev/sda11
```



```
$ sudo mkfs -t ext4 -b 2048 -v /dev/sda11
```



```
$ sudo mkfs -t ext4 -b 4096 -v /dev/sda11
```

vフラグ (verbose) は要求された情報を与えることに注意してください。このような小さなパーティションでは、デフォルトが1024バイトのブロックであることがわかります。

3.

```
$ sudo mkdir /mnt/tempdir
```



```
$ sudo mount /dev/sda11 /mnt/tempdir
```


マウント | grep tempdir
4.

```
$ sudo umount /mnt/tempdir
```



```
$ sudo mount -o ro /dev/sda11 /mnt/tempdir
```


アンマウント中にエラーが発生した場合は、現在ディレクトリにいないことを確認してください。
5.

```
$ sudo touch /mnt/tempdir/afile
```
6.

```
$ sudo umount /mnt/tempdir
```
7. この行を `/etc/fstab` に記述してください :

```
/dev/sda11 /mnt/tempdir ext4 defaults 1 2
```
8.

```
$ sudo mount /mnt/tempdir
```



```
sudo mount | grep tempdir
```
9. `etc/fstab`の行を変更します :

```
/dev/sda11 /mnt/tempdir ext4 noexec 1 2
```

では、そうしてください :

```
$ sudo mount -o remount /mnt/tempdir
```

```
$ sudo cp /bin/ls /mnt/tempdir
```

```
$ /mnt/tempdir/ls
```

ここでエラーが出るはずですが、なぜですか？



ループバック・ファイル・ソリューション

1.

```
$ sudo dd if=/dev/zero of=/imagefile bs=1M count=250
```
2.

```
$ sudo mkfs -t ext4 -v
```



```
$ sudo mkfs -t ext4 -b 2048 -v /imagefile
```

```
$ sudo mkfs -t ext4 -b 4096 -v /imagefile
```

これはファイルであり、パーティションではないとの警告が出ますが、そのまま進んでください。

vフラグ (verbose) は要求された情報を与えることに注意してください。このような小さなパーティションでは、デフォルトは1024バイトのブロックであることがわかります。

3.

```
$ sudo mkdir /mnt/tempdir
sudo mount -o loop /imagefile /mnt/tempdir
マウント | grep tempdir
```
4.

```
$ sudo umount /mnt/tempdir
$ sudo mount -o ro,loop /imagefile /mnt/tempdir
```

アンマウント中にエラーが発生した場合は、現在ディレクトリにいないことを確認してください。

5.

```
$ sudo touch /mnt/tempdir/afile
```
6.

```
$ sudo umount /mnt/tempdir
```
7. この行を `/etc/fstab` に記述してください :



で、`/etc/fstab`

```
/imagefile /mnt/tempdir ext4 loop 1 2
```

8.

```
$ sudo mount /mnt/tempdir
sudo mount | grep tempdir
```
9. `etc/fstab` の行を変更します :



で、`/etc/fstab`

```
/imagefile /mnt/tempdir ext4 loop,noexec 1 2
```

では、そうしてください :

```
$ sudo mount -o remount /mnt/tempdir
$ sudo cp /bin/ls /mnt/tempdir
$ /mnt/tempdir/ls
```

ここでエラーが出るはずですが、なぜですか？

練習問題23.3: NBDの設定とテスト

概要

このラボでは、ループバックデバイスを介して通信するNBDを設定し、テストする予定です。クライアントとサーバーの両方が同じシステム上で実行されます。

NBDエクスポートの場合、空のファイルが使用されます。

`opt`ディレクトリは、エクスポートされたデバイスとコンフィギュレーションファイルのために使用されます。本番環境では、異なる場所を好む場合があります。



CentOS-stream-8の問題

作成時のCentOS-8-Streamは正常に動作していません。CentOS-Stream-9またはFedoraをご利用ください。解決待ちです。

1. 最初のステップです :



Red Hat、Centos、またはFedoraの場合

gitでダウンロードしたソースコードを格納するディレクトリを用意する。クローンを作成し、NBDプログラムを



Debian、Ubuntu、Linux Mintの場合

NBDクライアントとサーバーのパッケージをインストールします。

2. `nbd-server.conf` ファイルを `/opt` ディレクトリに作成します。最小限の、コメントなしのファイルで十分です。(もっと面白い設定ファイルは、SOLUTIONSディレクトリにあります。
3. エクスポート用のファイルを2つ作成し、所有者をstudentに設定します。
4. `nbd-server`を起動します。
5. nbdサーバーがクライアントからのクエリに応答することを確認する。
6. `nbd`カーネルモジュールをインストールし、ブロックデバイスが存在することを確認する。
7. サーバーから提供されたイメージを、ipアドレス、ポートを使って、ローカルブロックデバイスに接続します。
8. NBDの既存情報を消去する。
9. NBDを使用して、GPTを作成します。
10. NBDにパーティションを作成する。
11. NBDにext3ファイルシステムを追加してテストします。



解答 23.3

1. 最初のステップです :



Red Hat、Centos、またはFedoraの場合

```
$ mkdir ~/src
$ cd ~/src
$ git clone https://github.com/NetworkBlockDevice/nbd.git
$ cd nbd/
```

`docbook-utils` と `autoconf-archive` パッケージは、あなたのシステムにはインストールされていないかもしれませんが、今すぐインストールしてください :

```
sudo dnf install docbook-utils
sudo dnf install autoconf-archive
```

プログラムを構築する :

```
$ ./autogen.sh
$ ./configure
$ make
```



Debian、Ubuntu、Linux Mintの場合

のことで、 Debianで、 Ubuntu、または Linux Minthave debパッケージが利用可能で、NBDユーティリティをインストールします。

```
sudo apt install nbd-client nbd-server
```

```
パッケージリストの読み込み...
完了 依存関係ツリーの構築...完了
状態情報の読み込み...完了
以下の新しいパッケージがインストールされ
ます: nbd-client nbd-server
アップグレード0、新規インストール2、削除0、アップグレードなし43。
Stdoutの出力は切り捨て
```

2. **nbd-server** の設定ファイルを **/opt** ディレクトリに作成します。**/opt** ディレクトリのファイルを編集するには、**sudo** が必要です。

```
$ sudo /opt/nbd-server.conf
```



/opt/nbd-server.conf

(ジェネリック)

```
allowlist = 1
listenaddr = 0.0.0.0
ポート = 10042
```

[フー]

```
エクスポート名 =
/opt/dsk1 readonly = false
```

[バー]である。

```
エクスポート名 = /opt/dsk2
```

```
3. sudo dd if=/dev/zero of=/opt/dsk1 status=progress bs=100M count=5
sudo dd if=/dev/zero of=/opt/dsk2 status=progress bs=100M count=5
sudo chmod 777 /opt
sudo chown student.student /opt/* とする。
```

```
4. $ sudo nbd-server -C /opt/nbd-server.conf
```

```
5. $ sudo nbd-client -l 127.0.0.1 -p 10042
```

```
交渉: ...
酒場
```

```
6. $ sudo modprobe -i nbd
$ ls /dev/nbd* です。
```

```
/dev/nbd0    /dev/nbd11 /dev/nbd14 /dev/nbd3 /dev/nbd6 /dev/nbd9
/dev/nbd1    /dev/nbd12 /dev/nbd15 /dev/nbd4 /dev/nbd7
/dev/nbd10 /dev/nbd13 /dev/nbd2    /dev/nbd5 /dev/nbd8
```

```
7. $ sudo nbd-client -N foo 127.0.0.1 10042 /dev/nbd0
```


8. NBDを一掃する：

```
$ sudo dd if=/dev/zero of=/dev/nbd0 bs=1M count=5
```

で5+0の記録。
5+0レコードアウト
5242880バイト (5.2MB、5.0MiB) コピー、0.00161564秒、3.2GB/秒

9. NBDにGPTラベルを作成します：

```
$ sudo su -c "echo 'label: gpt' | sfdisk /dev/nbd0 "
```

今、このディスクを誰も使っていないことを確認する ...OK

ディスク /dev/nbd0: 4 GiB, 4294967296 バイト, 8388608 セクタ
単位 : 1 * 512 = 512バイトのセクタ
セクタサイズ (論理/物理) : 512バイト / 512バイト
I/O サイズ (最小 / 最適) : 512バイト/512バイト

>>> スクリプトヘッダを受け付けました。
>>> 完了しました。
新しいGPTディスクラベル (GUID : 000C20C1-6929-424F-93E1-28319DE1FF) が作成されました。

新しい状況です：
ディスクレーベルの種類 : gpt
ディスクの識別子です : 000C20C1-6929-424F-93E1-28319DE1FF1F

パーティションテーブルが変更された。
ioctl()を呼び出してパーティションテーブルを再読み込みする。ディスクを同期させる。

10. NBDにパーティションを追加する：

```
$ sudo su -c "エコー ';' | sfdisk /dev/nbd0 "
```

今、このディスクを誰も使っていないことを確認する ...OK

ディスク /dev/nbd0: 4 GiB, 4294967296 バイト, 8388608 セクタ
単位 : 1 * 512 = 512バイトのセクタ
セクタサイズ (論理/物理) : 512バイト / 512バイト
I/O サイズ (最小 / 最適) : 512バイト / 512バイト
ディスクラベルタイプ : GPT
ディスクの識別子です : 000C20C1-6929-424F-93E1-28319DE1FF1F

昔の状況です：

>>> 新しいGPTディスクラベル (GUID: EFFC087B-A15C-7D42-809C-4BDED58EE238) を作成しました。
/dev/nbd0p1: タイプ 'Linux filesystem'、サイズ 4 GiB の新しいパーティション 1 が作成されました。
/dev/nbd0p2: 完了しました。

新しい状況です：
ディスクレーベルの種類 : gpt
ディスクの識別子です : EFFC087B-A15C-7D42-809C-4BDED58EE238

デバイス スタート エンドセクター サイズ タイプ
/dev/nbd0p1 2048 8388574 83865274G Linux ファイルシステム

パーティションテーブルが変更された。
ioctl()を呼び出してパーティションテーブルを再読み込みする。ディスクを同期させる。

11. NBDにext3ファイルシステムを追加してテストします：

```
$ sudo mkfs.ext3 -L nbd-foo /dev/nbd0
```

```
mke2fs 1.46.5 (30-Dec-2021)
128000個の4kブロックと128000個のinodeを持つファイルシステム
を作成中 Filesystem UUID: 6a01ff5a-9ad5-4f43-8050-9f2b8df26c14
Superblock/バックアップはブロックに格納されています：
    32768, 98304
```

```
グループテーブルの割り当て：完了
inodeテーブルの書き込み：完了
ジャーナル(4096ブロック)の作成：完了
スーパーブロックとファイルシステムのアカウンティング情報を書き込む: done
```

```
$ sudo mount LABEL=nbd-foo /mnt
```

```
$ sudo touch /mnt/file1
```

```
$ ls -l /mnt
```

```
トータル16
-rw-r--r-- 1 root root      0 Jun 9 10:58 file1
drwx----- 2 root root 16384 Jun 9 10:56 lost+found
```

第24章

Ext4ファイルシステム



24.1	ext4 機能.....	394
24.2	ext4 レイアウトとスーパーブロック、ブロックグループ.....	395
24.3	ダンプ2フ.....	398
24.4	tune2fs.....	399
24.5	研究室.....	400

学習目標

このセッションが終わるころには、次のことができるようになっているはずです：

- **ext4**ファイルシステムの主な機能とディスク上のレイアウトを説明する。
- ブロックグループ、スーパーブロック、データブロック、inodeの概念について説明する。
- **dumpe2fs**および**tune2fs**ユーティリティを使用します。

24.1 ext4 機能

ext4 ファイルシステムの特徴

- ext3やext2の祖先と下位互換性があります。
- ブロックリストの代わりにエクステントを使用
- 持続的なプリアロケーション
- ディレイアロケーション
- 以前の先祖の32,000サブディレクトリの制限を増やす
- ジャーナルチェックサム
- ファイルシステムの高速チェック
- 正確なタイムスタンプ(ナノ秒)



エクステント

エクステントは、連続したブロックのグループです。エクステントを使用することで、大容量ファイルのパフォーマンスを向上させ、フラグメンテーションを低減することができます。

ext4ファイルシステムは、最大1EBのボリュームと最大16TBのファイルサイズをサポートすることができます。エクステントは、旧来のブロックマッピング機構に取って代わるものです。

ext4はext3、ext2との後方互換性があります。ファイルに対してあらかじめディスクスペースを割り当てることができます。割り当てられた領域は、通常、保証され、連続的である。また、**allocate-on-flush**（ディスクにデータを書き込むまでブロックの割り当てを遅らせる）と呼ばれるパフォーマンス技術も使用します。ext4はext3の32000サブディレクトリの制限を破りました。

ext4はジャーナルにチェックサムを使用し、信頼性を向上させます。また、ジャーナリング中のディスクI/O待ちを安全に回避することができ、その結果、パフォーマンスが若干向上します。

ext4はナノ秒単位で計測されたタイムスタンプを提供します。

24.2 ext4 レイアウトとスーパーブロック、ブロックグループ

ext4 スーパーブロックとブロックグループ

- 先頭のスーパーブロックには、ファイルシステム全体の情報が含まれている
- 続いて、連続したブロックの集合で構成されるブロック・グループ
 - 行政情報を含む
 - ブロックグループ内の情報の冗長性が高い
 - その他のブロックはファイルデータを格納
- ブロックサイズは、ファイルシステム作成時に指定する
 - 512、1K、2K、4K、8Kなどのバイトで、メモリの1ページ（x86では4KB）より大きくはならない。

ext4ファイルシステムは、ブロックグループの集合に分割されています。ブロックアロケータは、シーク時間を短縮するために、各ファイルのブロックを同じブロックグループ内に維持しようとします。デフォルトのブロックサイズは4KBで、128MBのブロックグループを作成することになります。

ext4ではジャーナルを除くすべてのフィールドがリトルエンディアン順にディスクに書き込まれます。

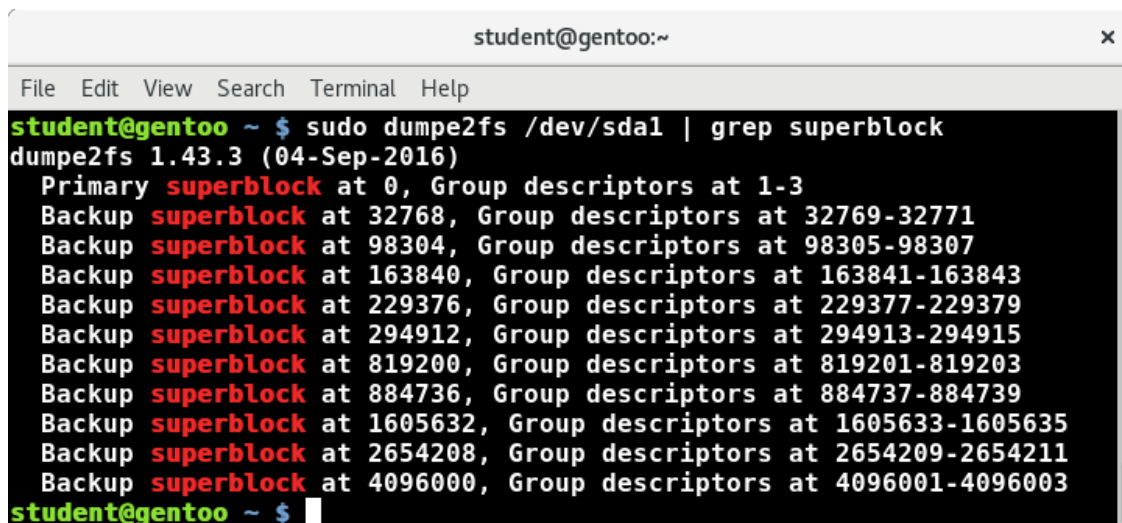
Super Block	Group Descriptors	Data Block Bitmap	Inode Bitmap	Inode Table (n blocks)	Data Blocks (n blocks)

図24.1 ext[3,4] ファイルシステムレイアウト

標準的なブロックグループのレイアウトは単純です。ブロックグループ0では、最初の1024バイトは未使用です（ブートセクターなどを置くため）。スーパーブロックは、ブロックグループ0を除いて最初のブロックから始まり、これにグループディスクリプタといくつかのGDTブロックが続きます。その後、データブロックビットマップ、inodeビットマップ、inodeテーブル、データブロックが続きます。

ブロックグループ

- スーパーブロックとグループディスクリプタの重複は、ファイルシステムの復旧に役立ちます。



```
student@gentoo:~  
File Edit View Search Terminal Help  
student@gentoo ~ $ sudo dumpe2fs /dev/sda1 | grep superblock  
dumpe2fs 1.43.3 (04-Sep-2016)  
Primary superblock at 0, Group descriptors at 1-3  
Backup superblock at 32768, Group descriptors at 32769-32771  
Backup superblock at 98304, Group descriptors at 98305-98307  
Backup superblock at 163840, Group descriptors at 163841-163843  
Backup superblock at 229376, Group descriptors at 229377-229379  
Backup superblock at 294912, Group descriptors at 294913-294915  
Backup superblock at 819200, Group descriptors at 819201-819203  
Backup superblock at 884736, Group descriptors at 884737-884739  
Backup superblock at 1605632, Group descriptors at 1605633-1605635  
Backup superblock at 2654208, Group descriptors at 2654209-2654211  
Backup superblock at 4096000, Group descriptors at 4096001-4096003  
student@gentoo ~ $
```

図24.2 ブロックグループブロックグループ

第1ブロックと第2ブロックは、どのブロックグループでも同じであり、スーパーブロックとグループ記述子を構成しています。

通常、カーネルが使用するのは最初のブロックグループのものだけで、重複したコピーはファイルシステムをチェックするときのみ参照されます。問題がなければ、カーネルは単に最初のブロックグループからコピーするだけです。

マスターコピーに問題があれば、健全なものが見つかってファイルシステム構造が再構築されるまで、次のコピーに行き、その繰り返しです。この冗長性により、ファイルシステムのチェックが定期的に行われる限り、ext2/3/4ファイルシステムを徹底的に炒めることは非常に難しくなります。

extファイルシステムファミリーの初期段階では、各ブロックグループには、スーパーブロックのコピーと同様に、各ブロックグループのグループディスクリプタが含まれていました。しかし、最適化として、今日、すべてのブロックグループがスーパーブロックのコピーとグループディスクリプタを持つわけではありません。正確な位置を確認するには、添付のスクリーンショットのように（適切なデバイスノードを入れて）調べればよいでしょう。これは、ファイルシステムをsparse_superオプション（デフォルト）で作成した場合に起こります。

スーパーブロックに迫る

- スーパーブロックには、ファイルシステムに関するグローバルな情報が含まれています
 - マウント数、最大マウント数
 - このファイルシステムのブロックサイズ
 - 1グループあたりのブロック数
 - フリーブロック数
 - フリーinode数
 - OS ID
- スーパーブロックは、複数のブロックグループに冗長的に格納されている

ディスクが正常にマウントされるたびに、マウントカウントがインクリメントされることに注意してください。ファイルシステムのチェックは、最大マウントカウント数または180日ごとのどちらか早いほうに行われます。

ブロックサイズは**mkfs**コマンドで設定することができます。

ファイルシステムのスーパーブロックは、ディスクのブロック0に格納されています。このスーパーブロックには、ファイルシステム自体に関する情報が含まれています。

24.3 ダンプ2フ

ext4のブロックとInodeの情報です : ダンプ2フ

- 最大数を設定するために使用するブロックサイズ :
 - ブロック
 - イノード
 - スーパーブロック
- ファイルシステムについては、**dumpe2fs**を参照してください :
 - リミッツ
 - 機能・フラグ
 - その他の属性

dumpe2fsプログラムを使用すると、特定のパーティションに関する情報を得ることができます。

```
$ sudo dumpe2fs /dev/sdb1

dumpe2fs 1.45.6 (20-Mar-2020) フ
ファイルシステムボリューム名 :
VMS 最終マウント先      /MMS
ファイルシステム      fce521c7-e2ce-414a-8a7e-e2311640802f
ファイルシステムマジックナンバー :
0xEF53 ファイルシステムリビジョン#
.      1 (ダイナミ
ック)
ファイルシステムの特徴 has_journal ext_attr resize_inode dir_index filetype needs_recovery extent
64bit flex_bg \ sparse_super large_file huge_file uninit_bg dir_nlink extra_isize
ファイルシステムのフラグ :
signed_directory_hash
デフォルトのマウントオプションです :
user_xattr acl ファイ
ルシステムの状態 :
クリーン
エラーの挙動 :
続ける
ファイルシステムOSの種類 : Linuxxf .
Inodeカウントです : 14352384
ブロック数です : 57388288
予約ブロック数 : 2869413
空きブロック数 : 22270800
フリーインコードです : 14352217
最初のブロック : 0
ブロックサイズ : 4096

1056のブロックビットマップ ( bg #0 + 1056 )
1072のInodeビットマップ ( bg #0 + 1072 )
1599-2110のInodeテーブル ( bg #0 + 1599 )
415空きブロック、8192空きinode、0ディレクトリー、8192未使用inode
フリーブロックです : 33822-33985, 34550-34691, 38803-38911
フリーインコード : 8193-16384
グループ2 : ( ブロック65536 ~ 98303 ) csum 0xdd9 [INODE_UNINIT, ITABLE_ZEROED].
1057のブロックビットマップ ( bg #0 + 1057 )
1073のInodeビットマップ ( bg #0 + 1073 )
2111-2622のInodeテーブル ( bg #0 + 2111 )
```

24.4 tune2fs

tune2fs : ファイルシステムのパラメータを

tune2fsは、ファイルシステムのパラメータを変更するために使用することができます。

- ファイルシステムチェック間の最大マウント数 (max-mount-count) を変更する :
- `interval-between-check` で

```
$ sudo tune2fs -i 10 /dev/sda1
```

- 変更可能なパラメーターの現在値を含むスーパーブロックの内容を一覧表示する :

```
$ sudo tune2fs -l /dev/sda1
```

基本的に **dumpe2fs** のグローバル情報を表示します。

```
$ sudo tune2fs -l /dev/sdb1
```

```
tune2fs 1.45.6 (20-Mar-2020) ファ
イルシステムボリューム名 : VMS
最後にマウントされた場所 : /VMS
ファイルシステム          fce521c7-e2ce-414a-8a7e-e2311640802f
ファイルシステムマジックナンバー :
ファイルシステムのリ      1 (ダイナミ
ビジョン番号 : ファイル  has_journal ext_attr resize_inode dir_index filetype needs_recovery extent 64bit flex_bg sparse_super large_file huge_file uninit_bg
ファイルシステムの機能 : フ dir_nlink extra_isize signed_directory_hash
ファイルシステムのフラ   user_xattr acl
グ : デフォルトのマウ   clean
ントオプション : ファ   コンティニュー
イルシステムの状態 :   リナックス
エラーの挙動 : ファイル 14352384
システム OS タイプ      57388288
: Inode カウント :      2869413
ブロック数です :      22270800
予約ブロック数です :   14352217
空きブロック数         0
フリーインコード       4096
ドです : 最初の        4096
ブロックです :
ブロックサイズ         2017年9月25日 (月) 14:14:57
フラグメントサイ      3月8日(月) 06:05:03 2021件
ズ                  3月8日(月) 06:05:03 2021件
-----
ファイルシステム      2003
が作成されました   -1
: 最終マウント時      2018年2月28日 (水) 14:24:15
間最終書き込み時      0 (<none>)である。
間マウント回数       14 TB
```

24.5 研究室

練習問題24.1デフラグ

Linuxを初めて使う人は、ファイルシステムのデフラグツールについての記述がないことに驚くことが多い。

しかし、Linuxを含むUNIX系OSのネイティブファイルシステムでは、ファイルシステムの断片化が深刻な問題とならない傾向があります。

これは主に、アクセス時間の速いディスクの最内周にファイルを詰め込もうとしないためです。その代わりに、ディスク全体に空き領域を広げ、ファイルを作成する際に、ファイル全体を1つまたは少数の断片で格納するのに十分な大きさの空きブロック領域が見つかる可能性を高めているのです。

最近のハードウェアでは、ディスクの最内周という概念は、いずれにせよハードウェアによって不明瞭にされています。



やめてください

SSDの場合、読み取り/消去/書き込みのサイクルが有限であるため、デフラグは実際にストレージメディアの寿命を縮める可能性があります。スマートなオペレーティングシステムでは、**SSD**ドライブのデフラグはデフォルトでオフ

さらに、新しいジャーナリングファイルシステム（**ext4**を含む）は、設計上**エクステント**（大きな連続した領域）で動作します。しかし、**ext4**ファイルシステムをデフラグするためのツールは存在します：

`$ sudo e4defrag`

使用方法 : `e4defrag [-v] ファイル...|ディレクトリ...|デバイス...`
 : `e4defrag -c ファイル...|ディレクトリ...|デバイス...`

e4defrag は **e2fsprogs** パッケージの一部であり、すべてのモダンな Linux ディストリビューションに搭載されているはずです。選択肢は2つだけです：

- `-v`：冗長にする。
- `-c`:実際には何もせず、解析して報告するだけです。

という論法があり得る：

- ファイル
- ディレクトリ
- デバイス全体

例

`$ sudo e4defrag -c /var/log`

<断片化されたファイル>について	今一番	サイズ/ エクステ ンション
1./var/log/lastlog	5/1	9 KB
2./var/log/sa/sa24	3/1	80 KB
3./var/log/rhsm/rhsm.log	2/1	142 KB
4./var/log/messages	2/1	4590 KB
5./var/log/Xorg.1.log.old	1/1	36 KB

フラグメンテーション 1
 [0-30 問題なし : 31-55 少し断片化 : 56- デフラグが必要] このディレクトリ (/var/log) はデフラグが必要ありません。
 された。

\$ sudo e4defrag /var/log

```
ext4 defragmentation for directory(/var/log)
[2/152]/var/log/Xorg.2.log :      100%  [ OK ]
[3/152]/var/log/Xorg.0.log.old: 100%。  [ OK ]
                                をク
                                リッ
                                クし
                                ます
                                。
[4/152]/var/log/messages-20141019.gz: 100%  [ OK ]です
                                。
[5/152]/var/log/boot.log :      100%  [ OK ]
                                をク
                                リッ
                                クし
                                ます
                                。
[7/152]/var/log/cups/page_log-20140924.gz: 100%  [ OK ]
成功する [ 112/152 ] をク
こと失敗 [ 40/152 ] リッ
                                クし
                                ます
                                。
[8/152]/var/log/cups/access_log-20141019.gz: 100%  [ OK ]
                                です
                                。
[9/152]/var/log/cups/access_logです :      100%  [ OK ]
                                です
                                。
[10/152]/var/log/cups/error_log-20141018.gz: 100%  [ OK ]
                                です
                                。
[11/152]/var/log/cups/error_log-20141019.gz: 100%  [ OK ]
                                です
                                。
[12/152]/var/log/cups/access_log-20141018.gz: 100%  [ OK ]
                                です
                                。
[14/152]/var/log/cups/page_log-20141018.gz: 100%  [ OK ]
                                です
                                。
...
[152/152]/var/log/Xorg.1.log.old :      100%  [ OK ]
                                です
                                。
```

e4defragを様々なファイル、ディレクトリ、デバイス全体で実行し、常に-cで最初に試してみてください。

一般的に、Linuxのファイルシステムがデフラグを必要とする傾向は、90%以上など非常にいっぱいになったとき、またはブートパーティションを使用するときのように、小さくて比較的大きなファイルを持つときだけであることがわかります。

 **練習問題24.2: tune2fsでファイルシステムのパラメーターを変更する**

フォーマットされたext4ファイルシステムのいくつかのプロパティをいじってみることにします。これは、最初にファイルシステムをアンマウントする必要はありません。

以下では、あなたが作成した画像ファイルを、次のように扱うことができます：

```
$ dd if=/dev/zero of=imagefile bs=1M count=1024
```

```
$ mkfs.ext4 イメージファイル
```

または、`/dev/sdaX`（変更したいファイルシステムがマウントされているパーティションを使用）をimagefileに置き換えることができます。

1. **dumpe2fs**を使用して、プロパティを調整したいファイルシステムの情報を取得します。表

示します：

- マウント回数の最大設定値（これを超えるとファイルシステムのチェックが強制的に行われます。）
- チェック間隔（ファイルシステムのチェックを強制的に行う時間の長さ）
- 予約したブロック数、総ブロック数

2. 変更する：

- 最大マウント数30に。
- チェック間隔を3週間に。
- ブロックの予約率を10%にする。

3. **dumpe2fs**を使って、もう一度ファイルシステムに関する情報を取得し、元の出力と比較します。



解答 24.2

1. \$ dumpe2fs imagefile > dumpe2fs-output-initial

```
dumpe2fs 1.42.9 (28-Dec-2013)
```

```
$ grep -i -e "マウント回数" -e "チェック間隔" -e "ブロック数" dumpe2fs-output-initial
```

```
ブロック数です :      262144
予約ブロック数 :      13107
マウント数です :      0
最大マウント数です :  -1
チェック間隔 :      0 (<none>)
                  である。
```

2. \$ tune2fs -c 30 イメージファイル

```
tune2fs 1.42.9 (28-Dec-2013) 最大マ
ウント数を 30 に設定する。
```

```
$ tune2fs -i 3w イメージファイル
```

```
tune2fs 1.42.9 (28-Dec-2013)
チェックの間隔を1814400秒に設定
```

```
$ tune2fs -m 10 イメージファイル
```

```
tune2fs 1.42.9 (28-Dec-2013)
予約ブロックの割合を10%に設定 ( 26214ブロック )
```

3. \$ dumpe2fs imagefile > dumpe2fs-output-final

```
dumpe2fs 1.42.9 (28-Dec-2013)
```

```
$ grep -i -e "マウント回数" -e "チェック間隔" -e "ブロック回数" dumpe2fs-output-final
```

```
ブロック数です :      262144
予約ブロック数 :      26214
マウント数です :      0
最大マウント数です :      30
チェック間隔 :      1814400 (3週間)
```

```
diff dumpe2fs-output-initial dumpe2fs-output-final
```

```
14c14
< 予約ブロック数です :      13107
---
> 予約ブロック数です :      26214
29c29
< 最終書き込み時間です :      2016年10月26日 ( 水 )
                  14:26:19
---
> 最終書き込み時間です : 31c31      2016年10月26日 ( 水 )
< 最大マウント数です :
```

```
> 最大マウント数です :      30
33c33,34
< チェック間隔です :      0 (<none>)である。
---
> チェック間隔 :      1814400 (3週間)
> 次のチェックはその後      2016年11月16日(水)
```


第25章

論理ボリューム管理（LVM）



25.1	論理ボリューム管理（LVM）	406
25.2	ボリュームとボリュームグループ	407
25.3	論理ボリュームの操作	408
25.4	論理ボリュームのサイズ変更	411
25.5	LVM Snapshots **について	412
25.6	研究室	413

学習目標

このセッションが終わるころには、次のことができるようになっているはずです：

- LVMの概念を説明する。
- 論理ボリュームの使い方を理解する。
- 論理ボリュームを作成します。
- 論理ボリュームを表示する。
- 論理ボリュームをリサイズする。
- LVMスナップショットを使用する。



ご注意ください

** これらのセクションは、一部または全部をオプションとして考慮することができる。これらのセクションには、背景となる参考資料、専門的なトピック、または高度なテーマが含まれています。インストラクターは、教室での経験や時間的制約に応じて、これらをカバーするかしないかを選択することができます。

25.1 論理ボリューム管理 (LVM)

エルブイエム

- 複数のディスクをグループ化してボリュームを作成
- RAID装置と同様の機能
 - RAIDデバイスの上に実際に構築することができる
- RAIDよりも拡張性が高い
 - サイズ変更（上下）が簡単にできます。
 - デバイスを追加して成長させることが可能

LVM (Logical Volume Management) は、1つの仮想パーティションを複数のチャンクに分割し、それぞれを異なるパーティションやディスクに配置することができます。

特に、論理パーティションやファイルシステムのサイズを変更し、ストレージスペースを増やしたり、配置を変更したりすることが非常に簡単になります。

1つまたは複数の物理ボリューム（ディスクパーティション）は、ボリュームグループにまとめられます。ボリュームグループは論理ボリュームに分割され、名目上の物理ディスクパーティションを模倣し、マウント可能なファイルシステムを含むようにフォーマットすることができます。

物理ボリュームや論理ボリュームの作成、削除、サイズ変更などを行うコマンドラインユーティリティは、さまざまなものが存在します。グラフィカルなツールとしては、ほとんどのLinuxディストリビューションで**system-config-lvm**が使用されています。しかし、**RHEL**はこのツールをサポートしなくなり、ファイルシステムタイプなどの最新のバリエーションで確実に動作するグラフィカルツールは現在存在しません。幸い、コマンドラインユーティリティは使い方が難しくなく、とにかく柔軟性があります。

LVMはパフォーマンスに影響を及ぼします。LVMレイヤーのオーバーヘッドから生じる追加コストは確実に存在します。しかし、非RAIDシステムでも、**ストライピング**（データを複数のディスクに分割すること）を使用すれば、ある程度の並列化の改善を達成することができます。

論理ボリュームは、RAIDデバイスと同様の機能を備えています。実際にRAIDデバイスの上に構築することができます。これにより、論理ボリュームはRAIDデバイスの冗長性とLVMのスケラビリティを持つことになります。

LVMはRAIDよりも拡張性に優れており、論理ボリュームは必要に応じて簡単にサイズ変更（拡大・縮小）することができます。より多くのスペースが必要な場合は、いつでも論理ボリュームにデバイスを追加することができます。

25.2 ボリュームとボリュームグループ

ボリュームとボリュームグループ

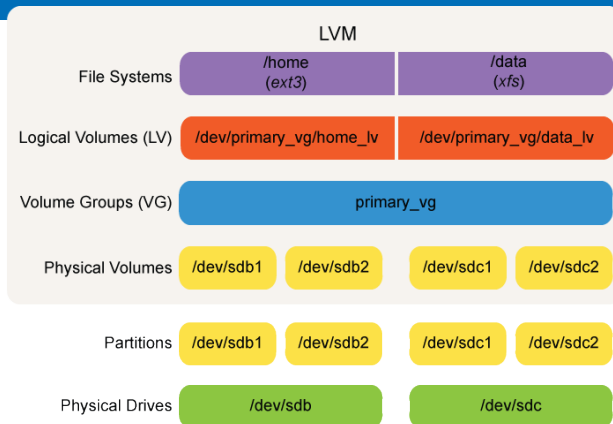


図25.1: LVMの構成要素

- デバイスは物理量に変換される

- マルチプル 物理的 へんれき は、ボリュームグループに分類されます。
 - エクステントに分割されたボリュームグループ
 - エクステントサイズを定義可能（デフォルト4MB）
- ボリュームグループから割り当てられた論理ボリューム
 - サイズまたはエクステント数で定義可能
 - ファイルシステムは、ロジカルボリューム上に構築される
 - どんな名前でもOK

パーティションは物理ボリュームに変換され、複数の物理ボリュームはボリュームグループにグループ化されます。ボリュームグループ内のスペースはエクステントに分割されます。エクステントのサイズはデフォルトで4MBですが、割り当て時に簡単に変更することができます。

ボリュームグループの作成と操作に使用されるコマンドラインユーティリティは多数あり、その名前は常に**vg**で始まります。を含む：

- **vgcreate**を使用します：ボリュームグループを作成します。
- **vgextend**する：ボリュームグループに追加する。
- **vgreduce**：ボリュームグループを縮小する。

ボリュームグループへの物理的なパーティションの出入りを操作するユーティリティは、**pv**で始まり、以下のものがあります：

- **pvcreeate**パーティションを物理ボリュームに変換します。
- **pvdisplay**を表示します：使用中の物理ボリュームを表示します。
- **pvmove**：ディスクやパーティションが何らかの理由で削除される場合に必要です。続いて、次のようになります：
- **pvremove**物理ボリュームからパーティションを削除

する。man lvmと入力すると、LVMユーティリティの全リ

ストが表示されます。

25.3 論理ボリュームの操作

論理ボリュームユーティリティ

```
student@ubuntu:~$ ls -lF /sbin/lv*
lrwxrwxrwx 1 root root      3 Feb 13  2020 /sbin/lvchange -> lvm*
lrwxrwxrwx 1 root root      3 Feb 13  2020 /sbin/lvconvert -> lvm*
lrwxrwxrwx 1 root root      3 Feb 13  2020 /sbin/lvcreate -> lvm*
lrwxrwxrwx 1 root root      3 Feb 13  2020 /sbin/lvdisplay -> lvm*
lrwxrwxrwx 1 root root      3 Feb 13  2020 /sbin/lvextend -> lvm*
-rwxr-xr-x 1 root root 2862872 Feb 13  2020 /sbin/lvm*
lrwxrwxrwx 1 root root      3 Feb 13  2020 /sbin/lvmconfig -> lvm*
lrwxrwxrwx 1 root root      3 Feb 13  2020 /sbin/lvmdiskscan -> lvm*
-rwxr-xr-x 1 root root 10312 Feb 13  2020 /sbin/lvmdump*
-rwxr-xr-x 1 root root 237624 Feb 13  2020 /sbin/lvmpolld*
lrwxrwxrwx 1 root root      3 Feb 13  2020 /sbin/lvmsadc -> lvm*
lrwxrwxrwx 1 root root      3 Feb 13  2020 /sbin/lvmsar -> lvm*
lrwxrwxrwx 1 root root      3 Feb 13  2020 /sbin/lvreduce -> lvm*
lrwxrwxrwx 1 root root      3 Feb 13  2020 /sbin/lvremove -> lvm*
lrwxrwxrwx 1 root root      3 Feb 13  2020 /sbin/lvrename -> lvm*
lrwxrwxrwx 1 root root      3 Feb 13  2020 /sbin/lvresize -> lvm*
lrwxrwxrwx 1 root root      3 Feb 13  2020 /sbin/lvs -> lvm*
lrwxrwxrwx 1 root root      3 Feb 13  2020 /sbin/lvscan -> lvm*
student@ubuntu:~$
```

図25.2 : 論理ボリュームユーティリティ

論理ボリュームを操作するユーティリティは数多く存在します。驚くなかれ、これらはすべてlvで始まります。ここでは、最もよく使われるものについて説明しますが、簡単なリストは、以下の方法で入手することができます：

\$ ls -lF /sbin/lv* を実行します。

これらのユーティリティは、起動時や修復・復旧時に必要となる可能性があるため、`/usr/sbin`ではなく、`/sbin`にあります。これらのほとんどはlvmにシンボリックリンクされており、スイスアーミーナイフのようなプログラムで、すべての作業を行います。呼び出された名前に基づいて、何をどのように要求されているかを把握します。これは、`pv*`と`vg*`のユーティリティの多くにも当てはまりますので、簡単に確認することができます。

```
student@ubuntu:~$ ls -lF /sbin/pv*
lrwxrwxrwx 1 root root      3 Feb 13  2020 /sbin/pvchange -> lvm*
lrwxrwxrwx 1 root root      3 Feb 13  2020 /sbin/pvck -> lvm*
lrwxrwxrwx 1 root root      3 Feb 13  2020 /sbin/pvcreate -> lvm*
lrwxrwxrwx 1 root root      3 Feb 13  2020 /sbin/pvdisplay -> lvm*
lrwxrwxrwx 1 root root      3 Feb 13  2020 /sbin/pvmove -> lvm*
lrwxrwxrwx 1 root root      3 Feb 13  2020 /sbin/pvremove -> lvm*
lrwxrwxrwx 1 root root      3 Feb 13  2020 /sbin/pvresize -> lvm*
lrwxrwxrwx 1 root root      3 Feb 13  2020 /sbin/pvs -> lvm*
lrwxrwxrwx 1 root root      3 Feb 13  2020 /sbin/pvscan -> lvm*
student@ubuntu:~$
```

図25.3 : 物理ボリュームユーティリティ

論理ボリュームの作成

1. ディスクドライブにパーティションを作成する（**fdisk**で8eと入力する）。
2. パーティションから物理ボリュームを作成する。
3. ボリュームグループを作成します。
4. ボリュームグループから論理ボリュームを割り当てる。
5. 論理ボリュームをフォーマットします。
6. 論理ボリュームをマウントします（必要に応じて`/etc/fstab`も更新します）。

lvcreateは、ボリュームグループ内から論理ボリュームを割り当てる。サイズは、バイト数またはエクステント数で指定します（デフォルトでは4MBであることを覚えておいてください）。名前は任意に設定可能です。

lvdisplay 利用可能な論理ボリュームをレポートします。

ファイルシステムは論理ボリュームに配置され、通常通り**mkfs**でフォーマットされます。

例えば、すでにパーティション`/dev/sdb1`と`/dev/sdc1`を作成し、タイプ8eを与えたと仮定すると、新しい論理ボリュームをセットアップして使用するための手順は次のようになります：

```
$ sudo pvcreate /dev/sdb1
$ sudo pvcreate /dev/sdc1
$ sudo vgcreate -s 16M vg /dev/sdb1
$ sudo vgextend vg /dev/sdc1
$ sudo lvcreate -L 50G -n mylvm vg
$ sudo mkfs -t ext4 /dev/vg/mylvm
$ mkdir /mylvm
$ sudo mount /dev/vg/mylvm /mylvm
```

必ず行を追加してください。

```
/dev/vg/mylvm /mylvm ext4 defaults 1 2
```

を `/etc/fstab` に追加して、これを永続的なマウントにします。

論理ボリュームを表示する

- **pvdisplay** - 物理ボリュームを表示する
\$ pvdisplay
\$ pvdisplay /dev/sda5
- **vgdisplay** - ボリュームグループを表示する
\$ vgdisplay
\$ vgdisplay /dev/vg0
- **lvdisplay** - 論理ボリュームを表示する
\$ lvdisplay
\$ lvdisplay /dev/vg0/lvm1

pvdisplayは、1つまたは複数の物理ボリュームを表示します。物理ボリューム名を省略すると、すべての物理ボリュームが表示されます。VGDISPLAYは、1つまたは複数のボリュームグループを表示します。ボリュームグループ名を省略すると、すべてのボリュームグループが表示されます。lvdisplayは、1つまたは複数の論理ボリュームを表示します。論理ボリューム名を省略すると、すべての論理ボリュームが表示されます。

```
student@ubuntu:~$ ls -lF /sbin/vg*
lrwxrwxrwx 1 root root 3 Feb 13 2020 /sbin/vgcfgbackup -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13 2020 /sbin/vgcfgrestore -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13 2020 /sbin/vgchange -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13 2020 /sbin/vgck -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13 2020 /sbin/vgconvert -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13 2020 /sbin/vgcreate -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13 2020 /sbin/vgdisplay -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13 2020 /sbin/vgexport -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13 2020 /sbin/vgextend -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13 2020 /sbin/vgimport -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13 2020 /sbin/vgimportclone -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13 2020 /sbin/vgmerge -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13 2020 /sbin/vgmknodes -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13 2020 /sbin/vgreduce -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13 2020 /sbin/vgremove -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13 2020 /sbin/vgrename -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13 2020 /sbin/vgs -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13 2020 /sbin/vgscan -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13 2020 /sbin/vgsplit -> lvm*
student@ubuntu:~$
```

図25.4 : ボリュームグループユーティリティ

25.4 論理ボリュームのサイズ変更

論理ボリュームのサイズ変更

- 物理的なパーティションのサイズを変更するよりかはるかに簡単です。
- エクステンツはボリュームグループ内のどこからでも可能です
- オンライン状態でもファイルシステムの縮小・拡張が可能
- エクステンツ数またはバイト数でサイズを変更可能（M、Gなど）
- のように **lvresize** で行うのがベストです：

```
$ sudo lvresize -r -L 20 GB /dev/VG/mylvm
```

ここで、**-r** オプションは、ボリュームサイズの変更と同時にファイルシステムのリサイズを行います。

- **resize2fs** で **lvextend**, **lvreduce** も可能です。
- リサイズユーティリティはファイルシステムに依存する

ext4 ファイルシステムを持つ論理ボリュームを成長させる：

```
$ sudo lvresize -r -L +100M /dev/vg/mylvm
```

ここで、プラス記号はスペースを追加することを示します。なお、ファイルシステムを大きくするためにアンマウントする必要はない。ファイルシステムを縮小するには

```
$ sudo lvresize -r -L 200M /dev/vg/mylvm
```

のように、ボリュームグループを縮小することもできます：

```
$ sudo pvmove /dev/sdc1
```

```
$ sudo vgreduce vg /dev/sdc1
```

25.5 LVM Snapshots **について

LVM スナップショット

- 既存の論理ボリュームの完全コピー
- バックアップ、アプリケーションテスト、VMに便利です
- スナップショットの原状はブロックマップ
- デルタを保存するためのスペースのみ使用
 - 元の論理ボリュームが変更された場合、元のデータブロックはスナップショットにコピーされます。
 - スナップショットにデータが追加された場合、そこだけに保存される

LVM Snapshotsは、既存の論理ボリュームの正確なコピーを作成します。

バックアップ、アプリケーションのテスト、VM（仮想マシン）のデプロイなどに便利です。スナップショットの元の状態は、ブロックマップとして保持されます。

スナップショットでは、デルタを保存するためのスペースしか使用しません：

- 元の論理ボリュームが変更されると、元のデータブロックはスナップショットにコピーされます。
- スナップショットにデータが追加された場合、そ

こだけに保存されます。既存の論理ボリュームのスナ

ップショットを作成するには：

```
$ sudo lvcreate -l 128 -s -n mysnap /dev/vg/mylvm
```

その後、マウントポイントを作成し、スナップショットをマウントするため：

```
$ mkdir /mysnap
```

```
$ mount -o ro /dev/vg/mysnap /mysnap
```

スナップショットを使用する場合、およびスナップショットを削除する場合：

```
$ sudo umount /mysnap
```

```
$ sudo lvremove /dev/vg/mysnap
```

スナップショットを使い終わったら、必ず削除してください。スナップショットを削除せず、変更によって容量がいっぱ

いになった場合、自動的に無効化されます。オリジナルと同じ大きさのスナップショットは、決してオーバーフローしません。

25.6 研究室



ビデオデモンストレーションのリソース

lvm_demo.mp4を使用しています。



練習問題25.1 ロジカルボリューム

250MBのパーティションを2つ使って、論理ボリュームを作成することにします。ここでは、実際にパーティション可能なディスクスペースがあると仮定しています。

1. 論理ボリューム (8e) タイプの250MBパーティションを2つ作成します。
2. パーティションを物理ボリュームに変換する。
3. myvgという名前のボリュームグループを作成し、2つの物理ボリュームをそこに追加します。デフォルトのエクステンツサイズを使用します。
4. ボリュームグループmyvgからmylvmという300MBの論理ボリュームを割り当てる。
5. 論理ボリュームmylvmをフォーマットし、`/mylvm`にマウントします。
6. 論理ボリュームに関する情報を表示するには、`lvdisplay` を使用します。
7. 論理ボリュームと対応するファイルシステムを350MBに成長させます。



解答 25.1

1. 実行する：

```
$ sudo fdisk /dev/sda
```

で、適当なハードディスクを使い、2つのパーティションを作成します。`fdisk`で`t`と入力すると、パーティションタイプを8eに設定することができます。設定しなくても問題はありませんが、混乱を避けるために設定しておくといでしょう。`w`でパーティションテーブルを書き換えて終了し、その後

```
$ sudo partprobe -s
```

または再起動し、新しいパーティションが有効になることを確認します。

2. 新しいパーティションを`/dev/sdaX`と`/dev/sdaY`と仮定しています：

```
$ sudo pvcreate /dev/sdaX
```

```
$ sudo pvcreate /dev/sdaY
```

```
$ sudo pvdisplay
```

3.

```
$ sudo vgcreate myvg /dev/sdaX /dev/sdaY
```

```
$ sudo vgdisplay
```

4.

```
$ sudo lvcreate -L 300M -n mylvm myvg
```

```
$ sudo lvdisplay
```

5.

```
$ sudo mkfs.ext4 /dev/myvg/mylvm
```

```
$ sudo mkdir /mylvm
```

```
$ sudo mount /dev/myvg/mylvm /mylvm
```

マウントを持続させたい場合は、`/etc/fstab`を編集して行を追加してください：



で、`/etc/fstab`

```
/dev/myvg/mylvm /mylvm ext4 defaults 0 0
```

6. `$ sudo lvdisplay`

7. `$ df -h`
`$ sudo lvresize -r -L 350M /dev/myvg/mylvm`
`$ df -h`

または

`$ sudo lvresize -r -L +50M /dev/myvg/mylvm`

とか、古い手法でできることがあります：

`$ df -h`
`$ sudo lvextend -L 350M /dev/myvg/mylvm`
`$ sudo resize2fs /dev/myvg/mylvm`
`$ df -h`

第26章

カーネルサービスとコンフィギュレーション



26.1	カーネルの概要.....	416
26.2	カーネルブートパラメーター	417
26.3	カーネルコマンドライン.....	418
26.4	ブートプロセスの不具合.....	419
26.5	シスセクトル.....	420
26.6	研究室.....	421

学習目標

このセッションが終わるころには、次のことができるようになっているはずです：

- カーネルが果たすべき主な責任と、それをどのように達成するかを把握する。
- カーネルコマンドラインで設定できるパラメータと、そのパラメータを1回のシステム起動時だけ、または持続的に有効にする方法について説明します。
- ブートプロセスの不具合をトラブルシューティングする。
- これらのパラメータに関する詳細なドキュメントはどこにあるのか知っている。
- **sysctl** を使用して、システム起動後、またはシステム再起動後も持続的にカーネルパラメータを設定します。

26.1 カーネルの概要

カーネルの概要

- カーネルはOSの核となる部分
- カーネルの責務は以下の通りです：
 - システムの初期化およびブートアップ
 - プロセスのスケジューリング
 - メモリ管理
 - ハードウェアへのアクセスを制御する
 - アプリケーションとストレージデバイス間のI/O（入出力）。
 - ローカルおよびネットワークファイルシステムの実装
 - ローカル（ファイルシステムのパーミッションなど）およびネットワーク上でのセキュリティコントロール
 - ネットワーク制御
- オペレーティングシステムには、カーネル以外にも多くのコンポーネントが含まれています

狭義には、Linuxは**OSのカーネルのみ**であり、カーネルと相互作用するライブラリやアプリケーションなど、他の多くのコンポーネントを含んでいます。

カーネルは、ハードウェアとソフトウェアを接続し、競合するアプリケーションやサービス間のメモリやCPU時間の割り当てなど、システムリソースを管理する不可欠な中心コンポーネントです。**デバイスドライバ**を使用して接続されたすべてのデバイスを処理し、デバイスをオペレーティングシステムで使えるようにします。

カーネルのみで動作するシステムの機能はかなり限定的であり、専用で集中的な**組み込み機器**にのみ見られるものです。

26.2 カーネルブートパラメーター

カーネルブートパラメーター

- `grub.cfg`のkernel行に表示されます。
- 起動時にインタラクティブに供給することも可能
- パラメータは、以下のどちらかで指定します：
 - 引数として単純な値
 - `param=value` の形式で、`value` には文字列、整数、整数の配列などを指定します。

```
vmlinuz root=/dev/sda6 rhgb quiet crashkernel=512M
```

- 意図的に秘密、隠された、パラメータはない
- カーネルソースの `kernel-parameters.txt` に列挙されているか、<https://kernel.org/doc/html/latest/admin-guide/kernel-parameters.html> の行に記載されています。

カーネルオプションはカーネル行の末尾に配置され、スペースで区切られる。カーネルブートパラメータの例：

```
linux /boot/vmlinuz-5.19.0 root=/dev/sda5 ro crashkernel=512M quiet selinux=0
```

- `root`: ルートファイルシステム (`root=UUID=...` や `root=/dev/sda5`、`root=LABEL=CentOS9`などの形式がある。
- `ro`: ブート時にルートデバイスをリードオンリーでマウントします。
- `crashkernel=512M`: `kdump`機能によるカーネルクラッシュダンプのために確保するメモリ量。
- `quiet`: ほとんどのログメッセージを無効にする
- `selinux=0` SELinuxを無効にする

オンマシンのカーネルのドキュメントの `man bootparam` と `kernel-parameters.txt` を見てください。これはカーネルソースか、`kernel-doc` のような名前のパッケージにあります。さらに簡単には、<https://kernel.org/doc/html/latest/admin-guide/kernel-parameters.html> をオンラインで見ることができます。

Linuxディストリビューションでは、そのディストリビューション特有のパラメータが追加されることが多いので注意が必要です (上の例の `rhgb` は Red Hat Graphical Boot の略です)。

26.3 カーネルコマンドライン

カーネルコマンドライン

- 起動時にカーネルコマンドラインでシステムに渡されるパラメータ
- **GRUB**設定ファイルのlinuxまたはlinux16の行に配置する：
grub.cfgを/bootの下のどこかに置く。
- 起動時にインタラクティブに変更可能
- 正確な外観は、Linuxのディストリビューションやバージョンによって異なります：

```
linux /boot/vmlinuz-4.15.0-58-generic \
    root=UUID=5cec328b-bcd6-46d2-bcfa-8430257cd7a5 \
    find_preseed=/preseed.cfg auto noprompt
priority=critical locale=en_US quiet crashkernel=512M-:192M
```

または

\$ cat /proc/cmdline

```
BOOT_IMAGE=/boot/vmlinuz-4.15.0-58-generic \
    root=UUID=5cec328b-bcd6-46d2-bcfa-8430257cd7a5
ro find_preseed=/preseed.cfg auto noprompt
```

カーネルのコマンドラインの例は、ディストリビューションによって異なりますが、次のようなものです：

```
linux /boot/vmlinuz-5.19.0 root=UUID=7ef4e747-afae-48e3-90b4-9be8be8d0258 ro quiet
```

```
linuxefi /boot/vmlinuz-5.2.9 root=UUID=77461ee7-c34a-4c5f-b0bc-29f4feecc743 ro crashkernel=auto rhgb
→ 静かなcrashkernel=384M-:128M
```

で、/boot/grubのようなboot下のサブディレクトリや、/boot/efi/EFI/centos/grub.cfgのような場所にあるgrub.cfgで見つけることができます。

指定されたvmlinuzファイル以降はすべてオプションです。カーネルが理解できないオプションは、init (pid) に渡されます。

=1) システム上で最初に実行されたユーザープロセス。システムがどのようなコマンドラインで起動されたかを確認するには、次のように入力します：

\$ cat /proc/cmdline

```
BOOT_IMAGE=(hd0,msdos2)/boot/vmlinuz-5.19.0 root=7f7221b8-60d8-41b9-b643-dfcc80527c37 ro rhgb quiet
→ crashkernel=512M
```



BLSFGが変えるもの

FedoraとRHEL/CentOSはBLSFG (Boot Loader Specification Configuration)を使って、カーネルコマンドラインを設定する場所を変えています。この情報は、/boot/grub2/grubenvか、/boot/loader/entriesにあるファイルを参照する必要があります。

26.4 ブートプロセスの不具合

ブートプロセスの不具合

- ブートプロセスに従う
 - バイオス
 - グループ
 - カーネル
 - イニット

システムが正しく起動しない、または完全に起動しない場合、各ステージで何が起るかを熟知していることが重要です。**BIOS**の段階を乗り越えたと仮定すると、以下のような状態になる可能性があります：

ブートローダー画面はありません：

GRUBの設定ミスやブートセクターが破損していないか確認する。ブートローダーの再インストールが必要ですか？

カーネルの読み込みに失敗しました：

ブートプロセス中にカーネルが**パニックになる場合**、カーネルの設定ミスや破損、**GRUB**設定ファイルのカーネルブートパラメータが正しくない可能性が高いです。過去にカーネルが正常に起動したことがある場合は、カーネルが破損しているか、不正なパラメータが供給されたかのどちらかです。カーネルを再インストールするか、起動時にインタラクティブな**GRUB**メニューに入り、最小限のコマンドラインパラメータを使用して、その方法で修正することができます。あるいは、レスキューイメージでブートしてみるのもいいでしょう。

カーネルはロードするが、ルートファイルシステムのマウントに失敗した：

ここでの主な原因としては

1. **GRUB**の設定ファイルが間違っている。
2. **etc/fstab**の設定が間違っている。
3. カーネルや**initramfs**イメージに組み込まれたルートファイルシステムタイプをサポートしない。

init 処理中に失敗した。

initが始まると、いろいろなことがうまくいかなくなります。以前は別のカーネルで動作していたのであれば、それは大きな手がかかりとなります。破損したファイルシステム、スタートアップスクリプトのエラーなどにも注意してください。3（グラフィックなし）や1（シングルユーザモード）など、より低いランレベルで起動してみてください。

26.5 シスセクトル

シスセクトル

- **sysctl**で`/proc/sys`の設定を操作することができます。
- 変数名は`/proc/sys`のパスに対応します。
- カーネルパラメータの表示、設定、変更が可能

```
File Edit View Search Terminal Help
[student@CentOS7 ~]$ sysctl -a
.....
kernel.pid_max = 131072
.....
kernel.tainted = 0
kernel.threads-max = 29215
.....
net.ipv4.ip_default_ttl = 64
.....
net.ipv4.ip_forward = 1
.....
vm.nr_hugepages = 0
.....
vm.swappiness = 30
.....
[student@CentOS7 ~]$
c7:/tmp>
```

図26.1 : sysctl

sysctlインターフェイスは、実行時にカーネルパラメータを読み込んで調整するために使用することができます。することで、現在の値を表示することができます：

```
$ sysctl -a
```

次の2つの文は等価である：

```
$ sudo sh -c 'echo 1 > /proc/sys/net/ipv4/ip_forward' です。
```

```
$ sudo sysctl net.ipv4.ip_forward=1
```

(最初の形式では、単純な`sudo`と`echo`の併用はできないので、図のような複雑な方法でコマンドを実行するか、`root`として実行する必要があります。

`etc/sysctl.conf`に設定すれば、起動時に設定を修正することができます。と入力することに注意してください：

```
$ sudo sysctl -p
```

は、ファイルを即座に消化し、見つかったパラメータをすべて設定します。これは、ブートプロセスの一部にもなっています。

i `/usr/lib/sysctl.d` および `/usr/lib/sysctl.d` を使用します。

systemdの登場により、事態は少し複雑になっています。ベンダーは、`/usr/lib/sysctl.d/`ディレクトリにあるファイルに設定を入れます。これらは、`/etc/sysctl.d`に置かれたファイルに追加されたり、代替されたりします。しかし、自己文書化された元のファイル (`/etc/sysctl.conf`) は、まだサポートされています。

26.6 研究室



ビデオデモンストレーションのリソース

`sysctl_demo.mp4`を使用しています。



練習問題26.1 : `sysctl`によるシステムチューナブル

1. 自分のシステムにpingが打てるかどうか確認する。
2. `net.ipv4.icmp_echo_ignore_all`の現在の値を確認します。この値は、システムがpingに応答するかどうかをオン/オフするために使用されます。値が0の場合、システムはpingに応答します。
3. `sysctl`コマンドラインユーティリティで値を1に設定し、Pingが応答するかどうかを確認します。
4. 値を0に戻し、元の動作を復元して表示します。
5. ここで、`/etc/sysctl.conf`を修正して値を変更し、再起動せずにこの設定ファイルをシステムが強制的に有効にします。
6. これが正しく機能したかどうかを確認する。

この作業を終えたら、おそらくシステムをリセットして元の動作に戻したくなるはずです。



解答 26.1

pingの対象は、localhost、127.0.0.1（ループバックアドレス）、または自分の実際のIPアドレスのいずれかを使用します。

1. `$ ping localhost`
2. `$ sysctl net.ipv4.icmp_echo_ignore_all`
3. `$ sudo sysctl net.ipv4.icmp_echo_ignore_all=1`
`$ ping localhost`
4. `$ sudo sysctl net.ipv4.icmp_echo_ignore_all=0`
`$ ping localhost`
5. `/etc/sysctl.conf`に次の行を追加します：



を `/etc/sysctl.conf` に追加します。

```
net.ipv4.icmp_echo_ignore_all=1
```

をしてから行う：

- ```
$ sysctl -p
```
6. `$ sysctl net.ipv4.icmp_echo_ignore_all`  
`$ ping localhost`

`etc/sysctl.conf`の変更は永続的なので、おそらく以前の状態に戻したいのでしょう。



### 練習26.2: 最大プロセスIDを変更する



Linuxシステムの正常な動作は、プロセスIDが、システム上の最初のユーザープロセスであるinitプロセスのPID=1から始まり、新しいプロセスが常に生成される（そして同様に死ぬ）につれて順次上がっていくことです。

しかし、PIDが`/proc/sys/kernel/pid_max`に示される値（従来は32768（32K））に達すると、より小さい数字に折り返されます。これは、PIDの枠がその数しかないため、システム上に32K以上のプロセスを持つことができないことを意味します。

1. 現在のPID最大値を取得する。
2. 現在発行されているPIDを確認する
3. `pid_max`を現在発行されている値より低い値にリセットする。
4. 新しいプロセスを開始し、PIDとして取得されるものを確認します。

## ✓ 解答 26.2



### 非常に重要

1つは`sysctl`を使う方法、もう1つは`/proc/sys/kernel/pid_max`に直接値をエコーする方法です。なお、`echo`の方法はrootであることが必要で、`sudo`ではうまくいきません。`sudo`では動作しないので、その理由を知ることはできません！

1. `sysctl kernel.pid_max`  
`$ cat /proc/sys/kernel/pid_max`

2. タイプです：

```
$ cat &
```

```
[1] 29222
```

```
$ kill -9 29222
```

3. `$ sudo sysctl kernel.pid_max=24000`  
`$ echo 24000 > /proc/sys/kernel/pid_max # これはrootで行う必要があります。`  
`$ cat /proc/sys/kernel/pid_max`

4. `$ cat &`

```
[2] 311
```

```
$ kill -9 311
```



### ご注意ください

やり直す場合、カーネルはPID=300から開始し、それ以下の値ではないことに注意してください。システムが既に終了している可能性があるため、カーネルは新しいPIDを生成する際に、そのPIDが既に使用されていないかどうかを常に確認する必要があります。Linuxカーネルは、システム上のプロセス数に依存しない、非常に効率的な方法でこれを行います。

## 第27章

# カーネルモジュール



|      |                    |     |
|------|--------------------|-----|
| 27.1 | カーネルモジュール .....    | 424 |
| 27.2 | モジュールユーティリティ ..... | 426 |
| 27.3 | モデム情報 .....        | 428 |
| 27.4 | モジュール構成 .....      | 429 |
| 27.5 | 研究室 .....          | 430 |

### 学習目標

このセッションが終わるころには、次のことができるようになっているはずです：

- カーネルモジュールを活用する利点を列挙する。
- カーネルモジュールのロード、アンロードには **insmod**、**rmmod**、**modprobe** を使用します。
- カーネルモジュールに関する情報を調べるには、**modinfo** を使用します。
- モジュールの設定方法について説明します。





## 27.1 カーネルモジュール

# カーネルモジュール

- カーネルへの拡張
- 必要に応じて、動的にロードまたはアンロードすることができる。
- デバイスドライバ、ネットワークプロトコルなどが含まれることが多い
- 場所は/lib/modules/\$(uname -r)です。
- 特定のカーネルバージョン用にコンパイルされています。
- カーネルはマイクロカーネルではなく、モノリシックなもののみです

Linuxカーネルでは、システム起動後に必要に応じてロード/アンロードできる重要なソフトウェアを含む**モジュール**を多用しています。

多くのモジュールは、システム内部や周辺に取り付けられたハードウェアを制御するための**デバイスドライバ**を組み込んでいます。また、ネットワークプロトコルを制御したり、さまざまなファイルシステムをサポートしたりと、さまざまな目的で利用できるモジュールもあります。

モジュールのロード時にパラメータを指定し、その動作を制御することができます。その結果、変化する状況やニーズに柔軟に対応することができます。

## lsmodで使用中のモジュールの一覧を表

```
c8:/tmp>lsmod
Module Size Used by
vmnet 57344 13
vmmon 126976 0
rfcomm 90112 4
nft_fib_inet 16384 1
bnep 28672 2
vmw_vsock_vmci_transport 32768 0
vmw_vmci 81920 1 vmw_vsock_vmci_transport
kvm_intel 270336 0
kvm 958464 1 kvm_intel
irqbypass 16384 1 kvm
ext4 905216 5
mbcache 16384 1 ext4
jbd2 147456 1 ext4
e1000e 290816 0
c8:/tmp>
```

図27.1 : lsmodの使用

ほとんど使われないであろうカーネル機能であっても、そのほとんどがモジュールとして構成することができます。この柔軟性は、開発やデバッグ時のテストにシステム再起動がほとんど必要ないため、新機能の開発にも役立つ。

他のオペレーティングシステムでもモジュール的な手法は使われていますが、Linuxは他のどのオペレーティングシステムよりもはるかに多く使っています。

## 27.2 モジュールユーティリティ

# モジュールユーティリティ

- **エルエスエムオー**  
ロードモジュールリスト
- **インスモード**  
モジュールを直接ロードする
- **アールエムオーディー**  
モジュールを直接取り外す
- **モジュプロブ**  
依存関係や位置情報を含む事前構築されたデータベースを使用して、  
モジュールをロードまたはアンロードします。
- **デップモッド**  
モジュール依存関係データベースを再構築する
- **モデム情報**  
モジュールに関する情報を表示する

モジュールを読み込むには**modprobe**を使用します：

```
$ modprobe e1000e
```

モジュールのアンロードや削除には、**modprobe -r**を使用します：

```
$ modprobe -r e1000e
```

**modprobe**は、モジュールの依存関係データベースを更新することを要求します。

`/lib/modules/$(uname -r)/modules.dep`.

**insmod**を使用してモジュールを直接ロードします（完全修飾モジュール名が必要です）：

```
$ insmod /lib/modules/$(uname -r)/kernel/drivers/net/ethernet/intel/e1000e.ko
```

モジュールを直接取り外すには、**rmmod**を使用します：

```
$ rmmod e1000e
```

ロードされたモジュールを一覧表示するには、**lsmod**を使用します：

```
$ lsmod
```

モジュールに関する情報（パラメータを含む）を表示するには、**modinfo**を使用します：

```
$ modinfo e1000e
```



## モジュールに関するいくつかの注意点

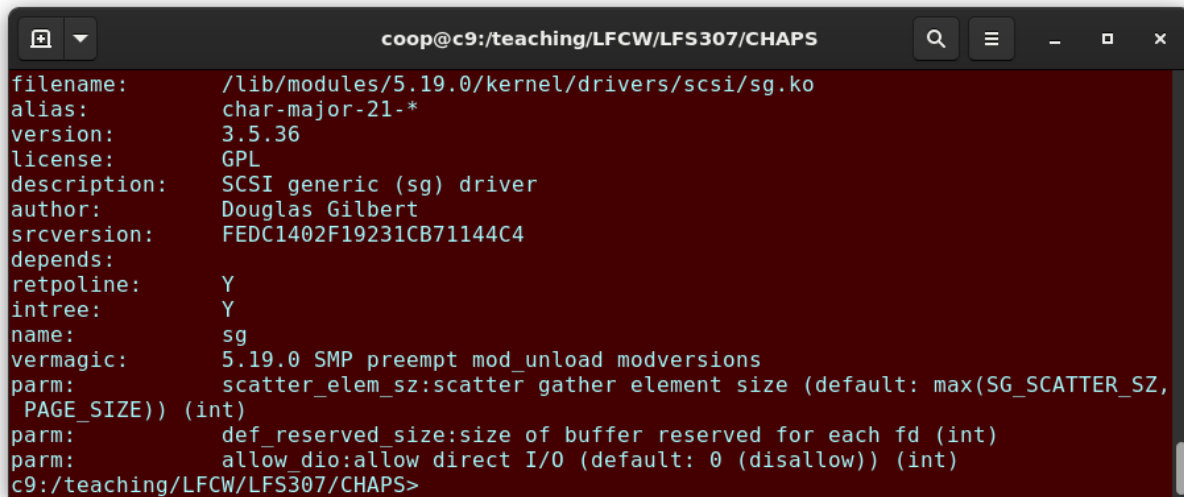
- モジュールのアンロードは不可能です：
  - 他のモジュールで使用中
  - 実行中のプロセスで使用中
- モジュールを読み込むことができない：
  - 異なるカーネル、または異なるオプションでコンパイルされる。
  - 現在実行中のモジュールと競合する
- モジュールは通常、**insmod**ではなく**modprobe**でロードされます。
- オープンソースライセンスが適用されていないモジュールは、カーネルが汚染されていると判断されます。

モジュールのロード・アンロードには、いくつかの重要な注意点があります：

- 他のモジュールに使用されているモジュールをアンロードすることは不可能である。  
をリストアップしています。
- 1つ以上のプロセスで使用されているモジュールをアンロードすることは不可能で、これは**lsmod**のリストからもわかります。しかし、ネットワークデバイスドライバモジュールのように、この参照カウントを追跡しないモジュールもあります。これは、ネットワークスタック全体の多くをシャットダウンして再起動しなければ、モジュールを一時的に交換することが難しくなりすぎるからです。
- **modprobe**でモジュールを読み込むと、最初に読み込む必要のある他のモジュールが自動的に読み込まれます。
- **modprobe -r**でモジュールをアンロードする際、そのモジュールによって使用されている他のモジュールが、同時にロードされた他のモジュールによって使用されていない場合、システムは自動的にアンロードします。

## 27.3 モデム情報

### modinfo 例



```
coop@c9:/teaching/LFCW/LFS307/CHAPS
filename: /lib/modules/5.19.0/kernel/drivers/scsi/sg.ko
alias: char-major-21-*
version: 3.5.36
license: GPL
description: SCSI generic (sg) driver
author: Douglas Gilbert
srcversion: FEDC1402F19231CB71144C4
depends:
retpoline: Y
intree: Y
name: sg
vermagic: 5.19.0 SMP preempt mod_unload modversions
parm: scatter_elem_sz:scatter gather element size (default: max(SG_SCATTER_SZ,
PAGE_SIZE)) (int)
parm: def_reserved_size:size of buffer reserved for each fd (int)
parm: allow_dio:allow direct I/O (default: 0 (disallow)) (int)
c9:/teaching/LFCW/LFS307/CHAPS>
```

図27.2 : modinfo

modinfoは、上のスクリーンショットに示すように、カーネルモジュールに関する情報（現在ロードされているかどうか）を調べるために使用でき、バージョン、ファイル名、デバイスドライバモジュールが扱えるハードウェアデバイス、ロード時に供給できるパラメータに関する情報を表示します。

モジュールに関する多くの情報は、`/sys`疑似ファイルシステムのディレクトリツリーでも見ることができます。上記の例では、`/sys/module/sg`の下を見ることになり、すべてのパラメータではないにしても、いくつかのパラメータは`/sys/module/sg/parameters`で読み書きすることができます。次に、これらの設定方法を説明します。

など、パラメータ値を指定しながら読み込むことができるモジュールが多い：

```
$ sudo /sbin/insmod <pathto>/e1000e.ko debug=2 copybreak=256
```

または、すでに適切なシステム位置にあるモジュールの場合は、より簡単です：

```
$ sudo /sbin/modprobe e1000e debug=2 copybreak=256
```

## 27.4 モジュール構成

**/etc/modprobe.d/ です。**

- **etc/modprobe.d**サブディレクトリのツリーで、末尾が「...」で終わるすべてのファイル。  
モジュールのロード、アンロード時に**modprobe**を使用して.conf拡張子をスキャンします。
- モジュール名のエイリアスを定義する
- 特定のモジュールをブラックリスト化することが可能
- ロード時にモジュールに渡すパラメータを定義する。
- モジュールのロード、アンロード時に実行するコマンドを指定する。

**/etc/modprobe.d** ディレクトリにあるファイルは、**modprobe** でロードする際に使用されるいくつかのパラメータを制御します。これらのパラメータには、モジュール名のエイリアスや自動的に供給されるオプションが含まれます。また、特定のモジュールをブラックリストに登録し、ロードされないようにすることもできます。

設定はモジュールのロード、アンロード時に適用され、ニーズの変化に応じて設定を変更することが可能です。

**etc/modprobe.d**のファイルの形式はシンプルで、1行に1コマンド、空行と#で始まる行は無視されます（コメントを追加するのに便利です）。行末にバックスラッシュを付けると次の行に続くので、ファイルを少しすっきりさせることができます。

## 27.5 研究室



### ビデオデモンストレーションのリソース

カーネルモジュールの使用.mp4

### 練習問題 27.1 : カーネルモジュール

1. システムで現在ロードされているすべてのカーネルモジュールをリストアップします。
2. 現在システムに搭載されていないモジュールをロードする。  
ディストリビューションカーネルを使用している場合、これを見つけるのは簡単です : `/lib/modules/□kernel-version□` を見ればよいのです。  
`/kernel/drivers/net` ディレクトリに移動し、1つ取得します。(ディストリビューションカーネルには、システムが必要とするすべてのデバイス、ファイルシステム、ネットワークプロトコルなどのドライバが付属しています)。しかし、カスタムカーネルを実行している場合、多くのアンロードされたモジュールがコンパイルされていないかもしれません。  
これらのギガビットイーサネットドライバは非常に一般的ですが、両方が一度にロードされる可能性は非常に低いいため、通常は `e1000.ko` または `e1000e.ko` を選択することが有効です。
3. ロードされたすべてのカーネルモジュールを再リストアップし、あなたのモジュールが本当にロードされているかどうかを確認します。
4. ロードされたモジュールをシステムから取り外す。
5. 再度リストアップし、モジュールが適切に削除されたかどうかを確認します。

### 解説 27.1

1. `$ lsmod`
2. 以下では、`e1000e` に使用したモジュール名を代入してください。どちらの方法でも構いませんが、もちろん、2番目の方法の方が簡単です。  

```
$ sudo insmod /lib/modules/$(uname -r)/kernel/drivers/net/ethernet/intel/e1000e.ko
$ sudo /sbin/modprobe e1000e
```
3. `$ lsmod | grep e1000e`
4. もう一度言いますが、どちらの方法でも大丈夫です。  

```
$ sudo rmmod e1000e
$ sudo modprobe -r e1000e
```
5. `$ lsmod | grep e1000e`

## 第28章

# デバイスとudev



|      |                   |     |
|------|-------------------|-----|
| 28.1 | udevとデバイス管理 ..... | 432 |
| 28.2 | デバイスノード .....     | 433 |
| 28.3 | ルール .....         | 436 |
| 28.4 | 研究室 .....         | 439 |

### 学習目標

このセッションが終わるころには、次のことができるようになっているはずです：

- デバイスノードの役割と、メジャー番号とマイナー番号の使い方を説明する。
- udevメソッドの必要性を理解し、その主要なコンポーネントをリストアップする。
- udevデバイスマネージャの機能を説明する。
- udevルールファイルを確認し、カスタムルールを作成する方法を学ぶ。

431



## 28.1 udevとデバイス管理

# ウデヴ

- **udev**は、User Device Managementの略です。
- **udev**は、内蔵されたハードウェアや周辺機器を動的に検出します：
  - システムブート時
  - いつでもホットプラグ時
- **udev**は、デバイスドライバのロードとアンロードを、必要に応じて適切な設定で処理します。
- デバイスノードは自動的に作成され、アプリケーションやオペレーティングシステムのサブシステムによって使用される
- システム管理者は、**udev**の操作方法を制御し、特別な細工をすることができます。  
**udevのルールです。**

**udev**のアクションは以下の通りです：

- デバイスの命名
- デバイスファイルおよびシンボリックリンクの作成
- ファイル属性の設定
- 必要なアクションを起こす

デバイスが追加されたり削除されたりすると、udevはカーネルからメッセージを受け取ります。そして、以下のルールを解析します。  
[etc/udev/rules.d](#)ディレクトリの.filesに、追加または削除されたデバイスに対するルールがあるかどうかを確認します。  
これらのルールは完全にカスタマイズ可能で、デバイスファイル名の指定、デバイスファイルの作成、作成するシンボリックリンクの指定、デバイスファイルに設定するファイル属性（ユーザーやグループの所有権を含む）の指定、さらにアクション（実行するプログラム）の指定まで可能です。



## 28.2 デバイスノード

# デバイスノード、ハードウェアを接続するソフトウェア

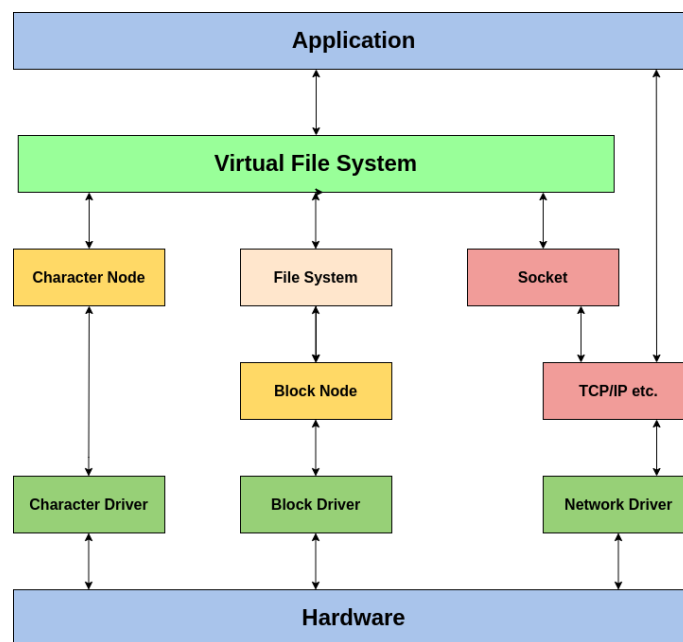


図28.1 : デバイスノード

## デバイスノード

- デバイスノードは/devディレクトリに配置される
- キャラクタデバイスとブロックデバイスにはデバイスノードがあり、ネットワークデバイスにはありません。
- プログラムは、これらのノードを通じて、通常のI/O方式でデバイスと通信します。
- 1つのデバイスドライバで複数のデバイスノードを使用することができる
- でデバイスノードを作成することができます :  
`$ sudo mknod [-m mode] /dev/name <タイプ> <メジャー> <マイナー>.`  
 を例として挙げます、  
`$ sudo mknod -m 666 /dev/mycdrv c 254 1`

`$ ls -l /dev`

```
合計0
crw----- 1 coop audio 14, 4 7月9日 01:54 オーディオ
crw----- 1 root root 10, 62 7月9日 01:54 autofs
lrwxrwx 1 root root 4 7月9日 01:54 cdrom -> scd0
lrwxrwx 1 root root 4 7月9日 01:54 cdrw -> scd0
crw----- 1 こう 根 5, 1 ジ 9 06:54 コンソール
 じょ
 うじ
 っし
 ゆう

lrwxrwxrwx 1 根 根 4 ジ 9 01:54 dvd -> scd0
 ュル

lrwxrwxrwx 1 根 根 4 ジ 9 01:54 dvdwriter -> scd0
 ュル

brw-r----- 1 根 ディス 8, 0 ジ 9 01:53 エスディー
---。 ク ュル

brw-r----- 1 根 ディス 8, 1 ジ 9 01:53 エスディーエーワン
---。 ク ュル

brw-r----- 1 根 ディス 8, 2 ジ 9 06:54 エスディーエーツー
---。 ク ュル

brw-r----- 1 根 ディス 8, 16 ジ 9 01:53 データバンク
```

```

---。 ク ュル
brw-r----- 1 根 ディス 8, 17 ジ 9 01:53 セイジ1
---。 ク ュル
brw-r----- 1 根 ディス 8, 18 ジ 9 01:53 セキリュウ2
---。 ク ュル

crw-rw-rw- 1 根 ティ 5, 0 ジ 9 01:54 ティ
---。 ク ュル
crw-rw-rw- 1 根 根 4, 0 ジ 9 14:54 ティティ0
---。 ク ュル
crw-rw-rw- 1 根 根 4, 1 ジ 9 06:54 ティティワン
---。 ク ュル
cr-r--r-- 1 根 根 1, 9 ジ 9 01:53 ランダム
---。 ク ュル

crw-rw-rw- 1 根 根 1, 5 ジ 9 01:54 ゼロ
---。 ク ュル

```

# udevコンポーネント

## • libudev ライブラリ

を可能にします。      アクセス  
                                  まで      デバイス情報

## • udevdまたはsystemd-udev dae-mon

devディレクトリを管理する

## • ウデバッドーム

制御・診断のためのユーティリティ

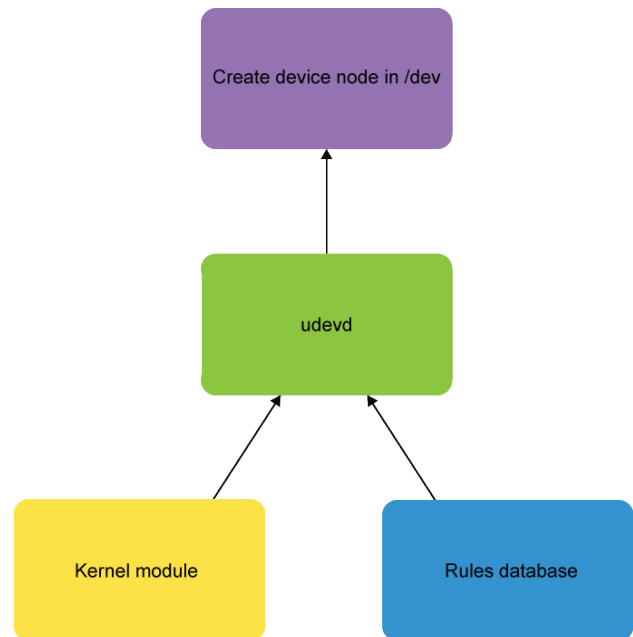


図28.2 : udevのルール

**udev**はデーモン（**udev**dまたは**systemd-udev**d）として実行され、ネットリンクソケットを監視する。新しいデバイスが初期化または削除されると、**uevent**カーネルファシリティがソケットを通じてメッセージを送信し、**udev**はこれを受信して、ルールに従って正しい名前とプロパティのデバイスノードを作成または削除する適切なアクションを実行します。

**udev**の構成要素は以下の3つです：

1. デバイスに関する情報へのアクセスを可能にする**libudev**ライブラリ。
2. devディレクトリを管理する**udev**dデーモン
3. 制御と診断のための**udev**admユーティリティです。

最初のカーネル起動時に /dev ディレクトリは空で、必要なときにデバイスノードが配置されます。この方法で使用する場合、**initramfs** イメージを使用して起動する必要があり、このイメージには **udev** インフラストラクチャだけでなく、予備デバイスノードのセットも含まれている可能性があります。

## 28.3 ルール

# udev ルールファイル

- ファイル名の場所と形式  
`/etc/udev/rules.d/<rulename>.rules`  
`/usr/lib/udev/rules.d/<rulename>.rules`
- 例を挙げます :  
`30-usb.ルール`  
`90-mycustom.ルール`  
`70-mouse.ルール`  
`60-persistent-storage.ルール`

1本の線に2つの別々のパーツが定義されています :

- 最初の部分は、`==`で示される1つ以上のマッチペアから構成される。これらは、デバイスの属性や特性を何らかの値にマッチさせようとするものである。
- 第2部は、ファイル名、グループ割り当て、さらにファイルのパーミッションなど、名前に値を割り当てる1つ以上の割り当てキーと値のペアで構成されます。

一致するルールが見つからない場合、デフォルトのデバイスノード名とその他の属性を使用します。

`$ cat /etc/udev/rules.d/99-fitbit.rules`

```
SUBSYSTEM=="usb", ATTR{idVendor}=="2687", ATTR{idProduct}=="fb01", SYMLINK+="fitbit", MODE="0666"
```

`$ cat /etc/udev/rules.d/60-vboxdrv.rules`

```
KERNEL=="vboxdrv", NAME="vboxdrv", OWNER="root", GROUP="vboxusers", MODE="0660"
KERNEL=="vboxdrv", NAME="vboxdrv", OWNER="root", GROUP="root", MODE="0666"
KERNEL=="vboxnetctl", NAME="vboxnetctl", OWNER="root", GROUP="vboxusers", MODE="0660"
SUBSYSTEM=="usb_device", ACTION=="add", RUN+="/usr/lib/virtualbox/VBoxCreateUSBNode.sh $major $minor"
→ $attr{bDeviceClass}」となります。
SUBSYSTEM=="usb", ACTION=="add", ENV{DEVTYPE}=="usb_device", RUN+="/usr/lib/virtualbox/VBoxCreateUSBNode.sh $major"
→ $minor $attr{bDeviceClass}」となります。
SUBSYSTEM=="usb_device", ACTION=="remove", RUN+="/usr/lib/virtualbox/VBoxCreateUSBNode.sh --remove $major $minor"
SUBSYSTEM=="usb", ACTION=="remove", ENV{DEVTYPE}=="usb_device", RUN+="/usr/lib/virtualbox/VBoxCreateUSBNode.sh"
→ --remove $major $minor" です。
```



## udevルールの作成

- ルールのためのフォーマット :

```
<match><op>value [, ...] <assignment><op>value [, ...]。
```

- サンプル

```
KERNEL=="sdb", NAME="my-spare-disk"
```

```
KERNEL=="sdb", DRIVER=="usb-disk", SYMLINK+="sparedisk"
```

```
KERNEL=="sdb", RUN+="usr/bin/my-program"
```

```
KERNEL=="sdb", MODE="0660", GROUP="mygroup"
```

1. `/etc/udev/rules.d`
2. `/run/udev/rules.d`
3. `/usr/lib/udev/rules.d`

## ルールファイルの例

- Fitbitのデバイスです：

```
$ cat /etc/udev/rules.d/99-fitbit.rules
```

```
SUBSYSTEM=="usb", ATTR{idVendor}=="2687", ATTR{idProduct}=="fb01",
→ SYMLINK+="fitbit", MODE="0666"
```

- クラッシュダンプの作成とkdump/kexecによる高速カーネルロードのため  
：

```
$ cat /usr/lib/udev/rules.d/98-kexec.rules
```

```
SUBSYSTEM=="cpu", ACTION=="add", PROGRAM="/bin/systemctl try-restart kdump.service"
SUBSYSTEM=="cpu", ACTION=="remove", PROGRAM="/bin/systemctl try-restart
→ kdump.service"
SUBSYSTEM=="memory", ACTION=="online", PROGRAM="/bin/systemctl try-restart
→ kdump.service"
SUBSYSTEM=="memory", ACTION=="offline", PROGRAM="/bin/systemctl try-restart
→ kdump.service"
```

- 仮想マシンハイパーバイザー「kvm」の場合：

```
KERNEL=="kvm", GROUP="kvm", MODE="0666"
```





## ビデオデモンストレーションのリソース

[using\\_udev\\_demo.mp4](#)。



1.

2.

3. `/dev`

4.

5. `/dev`



1. `/etc/udev/rules.d/75-myusb.rules`

```
$ cat /etc/udev/rules.d/75-myusb.rules
```

```
SUBSYSTEM=="usb", SYMLINK+="myusb"
```

2.

3. `$ ls -lF /dev | grep myusb`

4.

```
$ umount /media/whatever
```

```
/media/whatever
```

5. `$ ls -lF /dev | grep myusb`



- 29.1 IPアドレス
- 29.2 IPv4アドレスの種類
- 29.3 IPv6アドレスの種類
- 29.4 IPアドレスクラス
- 29.5 ネットマスク
- 29.6 ホスト名
- 29.7 エヌティーピー
- 29.8 研究室

## 学習目標

このセッションが終わるころには、次のことができるようになっているはずです：

- IPv4アドレスとIPv6アドレスの種類を区別する。
- IPアドレスクラスについて理解する。
- ネットマスクの役割を理解する。
- システムのホスト名を取得、設定、変更する。
- **NTP**を使用して、時刻の設定、同期、補正を行います。

# IPアドレス

- IPアドレスは、インターネット上のノードを一意に識別するために使用されます。
- IPアドレスはISPを通じて登録される
- **IPv4** 32ビットアドレスは、4オクテットから構成される。
- 予約したアド・ドレスのセットがある
- ここでは、現在も最も多く使われている**IPv4**に焦点を当てます。

148.114.252.10

(オクテットはちょうど1バイト、8ビットです)。

- **IPv6** 128ビットアドレスは、8つの16ビットオクテットペアで構成されます。

2003:0db5:6123:0000:1f4f:0000:5529:fe23

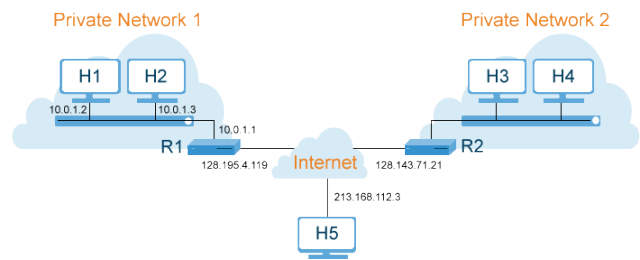


図29.1 : IPアドレス

## IPv4アドレスの種類

- ユニキャスト
- ネットワーク
- ブロードキャスト
- マルチキャスト

-

-

-

-

## リザーブド・アドレス

- ループバックインターフェースです :

127.x.x.x

- 住所は未定です :

0.0.0.0

- 汎用ブロードキャストプライベートアドレス

255.255.255.255

- その他

10.0.0.0 - 10.255.255.255

172.16.0.0 - 172.31.255.255

192.168.0.0 - 192.168.255.255

- 127.x.x.x

- 0.0.0.0

- 255.255.255.255

- 

10.0.0.0 - 10.255.255.255

172.16.0.0 - 172.31.255.255

192.168.0.0 - 192.168.255.255

<https://en.wikipedia.org/wiki/Reserved-IPアドレス>

## IPv6アドレスの種類

- ユニキャスト
- マルチキャスト
- エニーキャスト
- IPv4マップ

- 

- 

- 

- 

- 

- 

-

## IPv4アドレスクラス

- クラスA - 最上位オクテット範囲1-127
  - 128ネットワーク、1ネットワークあたり16,772,214ホスト、127.x.x.xはループバックのために予約されている
- クラスB - 最上位オクテット範囲128-191
  - 16,384ネットワーク、65,534ホスト/ネットワーク
- クラスC - 最上位オクテット範囲192～223
  - 2,097,152ネットワーク、1ネットワークあたり254ホスト
- クラスD - 最上位オクテット範囲224-239
  - マルチキャストアドレス
- クラスE - 最上位オクテット範囲240～254
  - 予約アドレス範囲



## ネットマスク

- クラスAネットマスク
  - 10進数で255.0.0.0
  - ff:00:00:00:00(16進数)
  - バイナリで11111111 00000000 00000000 00000000
- クラスBネットマスク
  - 10進数で255.255.0.0
  - ff:ff:00:00を16進数で
  - バイナリで11111111 111111 00000000 00000000
- クラスCネットマスク
  - 10進数で255.255.255.0
  - 16進数でff:ff:ff:00
  - バイナリで11111111 111111 111111 00000000

•  
•  
•  
•  
•

例172.16.2.17 ipアドレス  
    &255.255.0.0 netmask

-----

172.16.0.0      ネットワークアドレス

## ホスト名の取得と設定

- ネットワーク上の機器を他のノードと区別するためのラベルです。
- 歴史的には **nodename** とも呼ばれる
- インストール時に設定するが、後でいつでも変更可能
- どのユーザーでもホスト名を取得することができます：

\$ ホスト名

けんけんごうごう

- ホスト名を変更するには、root権限が必要です：

\$ sudo hostname lumpy

モコモコ

- 現在の値は、ほとんどのLinuxの **/etc/hostname** に常に格納されています。  
の配信を開始しました。
- 再起動しても変更が持続するようにするには、**systemd** インフラストラクチャの一部である **hostnamectl** を使用します。

\$ sudo hostnamectl set-hostname lumpy

### その他

/etc/sysconfig/network

/etc/hostname

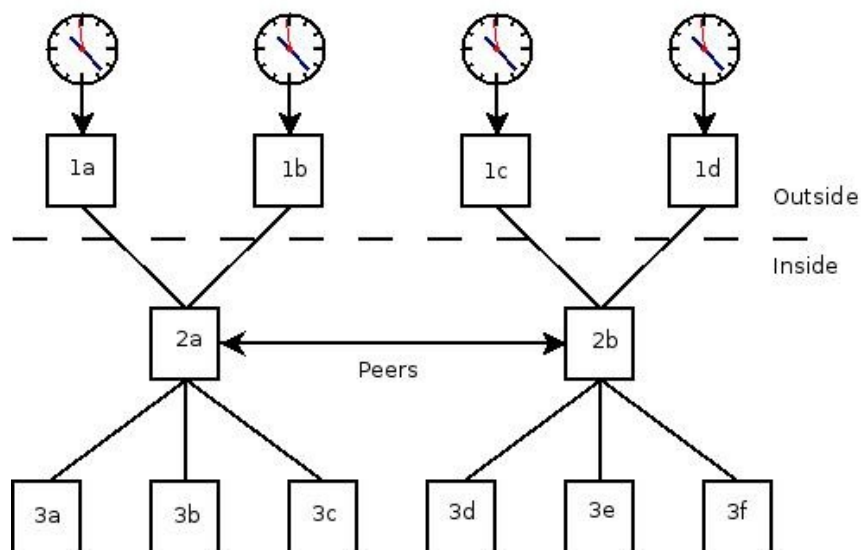
/etc/HOSTNAME

```
$ hostnamectl
 静的ホスト名 : c8
 アイコン名 : computer-desktop 筐体
 : デスクトップ
 マシンID : ce0c82382a8a4c80bbd6931a917a2f1c ブー
 トID : 94207b3fbd9b4891b9a94e21762a47cb
オペレーティングシステムRed Hat Enterprise Linux 8.2
↳ (オオツバ)
 dracut-049-70.git20200228.el8 (Initramfs)
 Kernel : Linux 5.11.6
 アーキテクチャ : x86-64
```

と使用方法のメッセージを見ることができます：

```
$ hostnamectl --help
hostnamectl [OPTIONS...] COMMAND ... システ
ム の ホ ス ト 名 を 照 会 ま た は 変 更 し ま す 。
 -h --help このヘルプを表示する
 --バージョン パッケージのバージョンを表示
 する
 --no-ask-password パスワードの入力を求めない
 -H --host=[USER@]HOST リモートホストで動作する。
 --儂い トランジェントホスト名の
 --静的 み設定 スタティックホスト
 名のみ設定 プリティホスト
す：
 地位 現在のホスト名を表示する
 ↳設定
 set-hostname NAME システムホスト名を設定
 set icon-name NAME する ホストのアイコン名
 set-chassis NAME set- を設定する
 deployment NAME ホストのシャーシタイプを
 ↳ホスト用 設定する 配置環境を設定する
 set-location NAME 設定する
 set-location NAME ホストの位置を設定す
```

# ネットワークタイムプロトコル



Key: 1 = stratum-1, 2 = stratum-2, 3 = stratum-3

図29.2 : ネットワークタイムプロトコル

- 
- 
- 
- 
- 
- 
- 
-

# NTPアプリケーション

NTPアプリケーションはいくつかありますが、代表的なものをいくつか紹介します：

- **ntp** ,<http://www.ntp.org>、デフォルトです。
- **chrony**、<http://chrony.tuxfamily.org>、断続的なネットワーク接続や仮想マシンなど、幅広い環境で動作するように設計されています。
- **systemd-timesyncd** は、**systemd** pack-age にのみ含まれる ntp クライアントです。

- [/etc/ntp.conf](#)
- [/etc/chrony.conf](#)
- [/etc/systemd/timesyncd.conf](#)

## ntpdクライアントの設定

- NTPソースをサーバーまたはピアとして選択します。
- NTPデーモンの起動
- 時刻が同期していることを確認する

<http://www.pool.ntp.org/en/>

/etc/ntp.conf



### /etc/ntp.conf

ドリフトファイル

/var/lib/ntp/ntp.drift プール

0.pool.ntp.org  
プール 1.pool.ntp.org  
pool 2.pool.ntp.org  
pool 3.pool.ntp.org

### # timedatectl

現地時間2023-03-05 ( 日 ) 16:13:50 UTC  
世界標準時です : 2023-03-05 ( 日 ) 16:13:50 UTC  
RTCの時間です : 2023-03-05 ( 日 ) 16:13:50  
タイムゾーンEtc/UTC (UTC,  
+0000) システムクロックの同期化 : はい  
NTPサービス : アクテ  
ィブ ローカルTZのRTC :  
なし

## ntpdサーバーの設定

ntpサーバーの設定要素は、以下のとおりです。

[/etc/ntp.conf](#) ファイルを参照してください。ここでは、ntpサーバーに関連する項目を紹介します：

- ローカルマシンを時間参照とすることを宣言する
  - サーバーとファッジのディレクション
- **ntpq** および **ntpd** コマンドを使用して、タイムサーバーに問い合わせることができるユーザーを制御する (CVE-2013-5211 を参照)
  - リストリクトディレクティブ
- どのシステムがntpピアであるかの宣言
  - ピアは、[/etc/ntp.conf](#) ファイルに記載されています。
- NTPデーモンの起動

/etc/ntp.conf



### /etc/ntp.conf: アクセスコントロール

```
デフォルトのポリシーでクエリを防ぐ
restrict default nopeer nomodify notrap noquery #
特定のサブネットからのクエリを許可する。
restrict 123.123.x.0 mask 255.255.255.0 nopeer nomodify notrap #
特定のホストからのクエリを許可する。
restrict 131.243.1.42 nopeer nomodify notrap noquery #
unrestrict localhost
127.0.0.1 を制限する
```



### /etc/ntp.conf

```
ピア 128.100.49.12
ピア 192.168.0.1
```



### /etc/ntp.conf

```
サーバ 127.127.1.0
ファッジ 127.127.1.0 ストラタム10
```





1.

```
yum install ntp
zypper install ntp
apt install ntp
```

2.

`/etc/ntp.conf``/etc/ntp.conf`

```
server 0.pool.ntp.org
server 1.pool.ntp.org
server 2.pool.ntp.org
server 3.pool.ntp.org
```

3.

```
systemctl restart ntpd
systemctl restart ntp
```

4.

```
$ ntpq -p
```

**ご注意ください**

出力を得る前に、タイムサーバーが同期するのを数分待つ必要があるかもしれません。



- 30.1 ネットワーク機器
- 30.2 アイピー
- 30.3 イフコンフィグ
- 30.4 予測可能なネットワークインターフェースデバイス名
- 30.5 ネットワーク構成ファイル
- 30.6 ネットワークマネージャー
- 30.7 ルーティング
- 30.8 仮想ネットワークインターフェイス
- 30.9 DNSと名前解決
- 30.10 ネットワークトラブルシューティング
- 30.11 ネットワーク診断
- 30.12 研究室

## 学習目標

このセッションが終わるころには、次のことができるようになっているはずです：

- ネットワークデバイスを特定し、オペレーティングシステムがどのようにデバイスに名前を付け、特定の職務に割り付けるかを理解する。
- **ip** ユーティリティを使用して、デバイス、ルーティング、ポリシーベースルーティング、およびトンネリングを表示および制御します。
- コマンドラインまたはシステム構成スクリプトからネットワークインターフェイスパラメータを設定、制御、照会するには、古い**ifconfig**プログラムを使用します。
- **PNIDN ( Predictable Network Interface Device Names )** 方式を理解する。
- **etc**ディレクトリにある主なネットワーク設定ファイルを知ることができる。
- ネットワークマネージャー ( **nmtui**および**nmcli** ) を使用して、ディストリビューションに依存しない方法でネットワークインターフェイスを設定します。
- デフォルトルートとスタティックルートの設定方法を知っている。

## ネットワーク機器

- 名称は通常、型式識別子の後に数字が付いたものです：
  - イーサネット : eth0、eth1、eno1、eno2、など。
  - 無線 : wlan0、wlan1、wlp3s0、wlp3s2、その他
  - 橋渡しされる : BR0、BR1、など。
  - 仮想 : vmnet1、vmnet8、など。
- 1台の機器に複数のIPアドレスを持たせることができる
  - 古いシステムでは、eth0:4 → eth0 のような関連付けがなされていることが多い。
  - **ifconfig**の代わりに**ip**を使用する新しいシステムでは、1つのインターフェイス名に複数のアドレスを割り当てています。

/dev

# アイピー

- インターフェースパラメーターの設定、制御、問い合わせ、マニピュレーションに使用します：
  - デバイス
  - ルーティング
  - トンネル
- 基本的な構文です：  
`ip [ OPTIONS ] OBJECT { COMMAND | help }.`  
`ip [ -force ] -batch filename`  
ここで、第2形式は、指定されたファイルからコマンドを読み込むことができます。
- `ioctl`システムコールではなく、ネットリンクソケットで動作する。
- `ifconfig`より新しく、より多機能になりました。

```
ip [OPTIONS] OBJECT { COMMAND | help }.
ip [-force] -batch filename
```

表30.1 : ip

| オブジェクト | 機能                        |
|--------|---------------------------|
| 宛先     | IPv4またはIPv6プロトコルのデバイスアドレス |
| リンク    | ネットワーク機器                  |
| アドレス   | マルチキャストアドレス               |
| モニター   | ネットリンクメッセージを見る            |
| ルート    | ルーティングテーブルエントリー           |
| 野線     | ルーティングポリシーデータベース内のルール     |
| トンネル   | Tunnel over IP            |

## ipを使用します：例

- すべてのインターフェイスの情報を表示します：  
`ip link show`
- 統計情報を含む eth0 インタフェースの情報を表示します：  
`$ ip -s link show eth0`
- eth0にipアドレスを設定します：  
`$ sudo ip addr add 192.168.1.7 dev eth0`
- インターフェースeth0をダウンさせる：  
`$ sudo ip link set eth0 down`
- インターフェースeth0のMTUを1480にする  
`$ sudo ip link set eth0 mtu 1480`
- ネットワークへのルートを設定する：  
`$ sudo ip route add 172.16.1.0/24 via 192.168.1.5`

```
student@ubuntu: ~
student@ubuntu:~$ ip -s link show ens33
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT group default qlen 1000
 link/ether 00:0c:29:c8:bf:ca brd ff:ff:ff:ff:ff:ff
 RX: bytes packets errors dropped overrun mcast
 1582717 1166 0 0 0 0
 TX: bytes packets errors dropped carrier collsns
 59159 746 0 0 0 0
student@ubuntu:~$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
 link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
 inet 127.0.0.1/8 scope host lo
 valid_lft forever preferred_lft forever
 inet6 ::1/128 scope host
 valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
 link/ether 00:0c:29:c8:bf:ca brd ff:ff:ff:ff:ff:ff
 inet 172.16.2.128/24 brd 172.16.2.255 scope global dynamic ens33
 valid_lft 1096sec preferred_lft 1096sec
 inet6 fe80::bf00:6f80:9502:f589/64 scope link
 valid_lft forever preferred_lft forever
student@ubuntu:~$
```

図30.1 : ipの使用

## イフコンフィグ

- インターフェースパラメーターの設定、制御、問い合わせに使用します。
- **ip**に取って代わられること：
  - 最近のLinuxディストリビューションでは、デフォルトでインストールされないものがあります。
- コマンドラインから使用

```
$ ifconfig
$ ifconfig eth0
$ ifconfig eth0 192.168.1.50
$ ifconfig eth0 netmask 255.255.255.0
$ ifconfig eth0 down
$ ifconfig eth0 mtu 1480
```
- ネットワーク起動スクリプトからも呼び出されます

```
File Edit View Search Terminal Help
x7:/tmp>/sbin/ifconfig
enp0s31f6: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
 inet 192.168.1.250 netmask 255.255.255.0 broadcast 192.168.1.255
 ether 54:ee:75:b5:f8:39 txqueuelen 1000 (Ethernet)
 RX packets 251841 bytes 305672016 (291.5 MiB)
 RX errors 0 dropped 0 overruns 0 frame 0
 TX packets 148108 bytes 105257478 (100.3 MiB)
 TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
 device interrupt 16 memory 0xf1200000-f1220000

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
 inet 127.0.0.1 netmask 255.0.0.0
 loop txqueuelen 1000 (Local Loopback)
 RX packets 100 bytes 7820 (7.6 KiB)
 RX errors 0 dropped 0 overruns 0 frame 0
 TX packets 100 bytes 7820 (7.6 KiB)
 TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlp4s0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
 ether 7e:6d:db:1e:16:9c txqueuelen 1000 (Ethernet)
 RX packets 140946 bytes 192702080 (183.7 MiB)
 RX errors 0 dropped 0 overruns 0 frame 0
 TX packets 44664 bytes 7435027 (7.0 MiB)
 TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

x7:/tmp>
```

図30.2 : ifconfigのアウトプット

## ifconfig の例

- すべてのインターフェース、または1つのインターフェースに関する情報を表示します :

```
$ ifconfig
```

```
$ ifconfig eth0
```

- インターフェースeth0のIPアドレスを192.168.1.50に設定します :

```
sudo ifconfig eth0 192.168.1.50
```

- ネットマスクを24ビットに設定する :

```
sudo ifconfig eth0 netmask 255.255.255.0
```

- インターフェースeth0をアップまたはダウンさせる :

```
sudo ifconfig eth0 up
```

```
sudo ifconfig eth0 down
```

- インターフェースeth0の**MTU** ( 最大転送単位 ) を1480バイトに設定します :

```
$ sudo ifconfig eth0 mtu 1480
```



## プレディクタブルネットワークインターフェイス 名称

- **PNIDN ( Predictable Network Interface Device Names )** は、udevと連携し、**systemd**と統合しています。
- デバイスにつけられる名前は、現在5種類あります：
  1. ファームウェアまたは**BIOS**が提供するオンボードデバイスのインデックス番号を組み込んだもの。例：eno1
  2. ファームウェアまたは**BIOS**が提供する**PCI Express**ホットプラグスロットのインデックス番号を組み込んでいます。例：ens1
  3. ハードウェア接続の物理的および/または地理的な位置を組み込むこと。例：enp2s0
  4. MACアドレスを組み込んだもの。例：enx7837d1ea46da
  5. 昔ながらの古典的な方法で。例：eth0

```
ip link show | grep enp
2: enp4s2: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state DOWN mode DEFAULT qlen 1000
3: enp2s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT qlen 1000
```

```
$ ifconfig | grep enp
enp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
enp4s2: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
```

```
lscpi | grep Ethernet
02:00.0 イーサネットコントローラ : Marvell Technology Group Ltd.88E8056 PCI-E Gigabit Ethernet Controller (rev 12)
04:02.0 Ethernet controller : Marvell Technology Group Ltd.88E8001 ギガビットイーサネットコントローラ (rev 14)
```

```
ip link show | grep wl
3: wlp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DORMANT qlen 1000
```

```
$ lspci | grep Centrino
03:00.0 ネットワークコントローラ : Intel Corporation Centrino Advanced-N 6205 [Taylor Peak] (rev 34)
```

## ネットワーク構成ファイル

- 各ディストリビューションは、ファイルやディレクトリのセットを独自に用意しています：
  - レッドハット
    - `/etc/sysconfig/network`
    - `/etc/sysconfig/network-scripts/ifcfg-ethX`
    - `/etc/sysconfig/network-scripts/ifcfg-ethX:Y`
    - `/etc/sysconfig/network-scripts/route-ethX`
  - デビアン
    - `/etc/network/interfaces`
  - SUSE
    - `/etc/sysconfig/network`
- **systemd**の標準化が進んでいるため
- ネットワークマネージャーを使用することが望ましい場合が多い
- 新しいディストリビューションでは、これらのファイルがなくなっているか

## ネットワークマネージャー設定ファ

- **Network Manager**は、いくつかのディストリビューションから、従来のネットワークファイルを使用することができます。ネットワークプロファイルは、プラグインでサポートされています。
- キーバリューは、好ましいファイル形式です。
- 従来の構成と互換性を持たせるためのオプションのプラグインがいくつか用意されています。
  - **ifupdown** for `/etc/network/interfaces`
  - **ifcfg-rh** for `/etc/sysconfig/network-scripts`
  - **ifcfg-suse** は、SUSE と openSUSE の単純な互換性のためです。
  - **key-file**は、システム固有の設定ファイルを汎用的に置き換えるものです。

追加情報については、<https://wiki.gnome.org/Projects/NetworkManager/SystemSettings/> をご覧ください。

`/etc/NetworkManager/NetworkManager.conf`

<https://developer-old.gnome.org/NetworkManager/stable/NetworkManager.conf.html>

## ネットワークマネージャ

- ネットワーク・インターフェースは、ハードウェア、ソフトウェアともに静的なものがほとんどでした。
- 最近のシステムは、動的な構成を持つことが多い：
  - デバイスの移動に伴い、ネットワークが変化する可能性がある
  - ワイヤレスデバイスは、接続するネットワークの選択肢が多いかもしれない
  - 無線機器などのハードウェアを接続したり、電源を入れたり切ったりすると、デバイスが変化することがあります。
- これまで説明してきた設定ファイルは、より静的な状況に対応するために作成されたもので、ディストリビューションに大きく依存します。
- ネットワークマネージャは、異なるシステムでほとんど同じであるべきで

/エトセトラ

# ネットワークマネージャインタフ

- GUI

すべてのLinuxデスクトップ (GNOME、KDE、XFCEなど) には、通常、水平パネルの何かをクリックすることで到達できるものがあります。

- エヌエムトゥイ

ネットワークマネージャテキストユーザーインターフェイスは、**ncurses** ライブラリインターフェイスを使用した、かなりシンプルなナビゲート可能なコマンドラインインターフェイスです。

- エヌエムシーリ

ネットワークマネージャコマンドラインインターフェイスは、上位のインターフェイスが使用できる下位のメソッドです。スクリプトに有効です

# エヌエムトゥイ

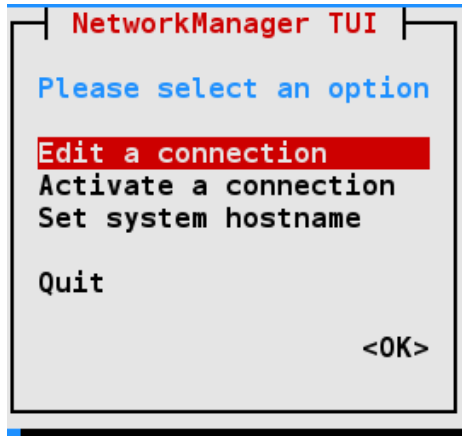


図 30.3 : nmtui メイン画面

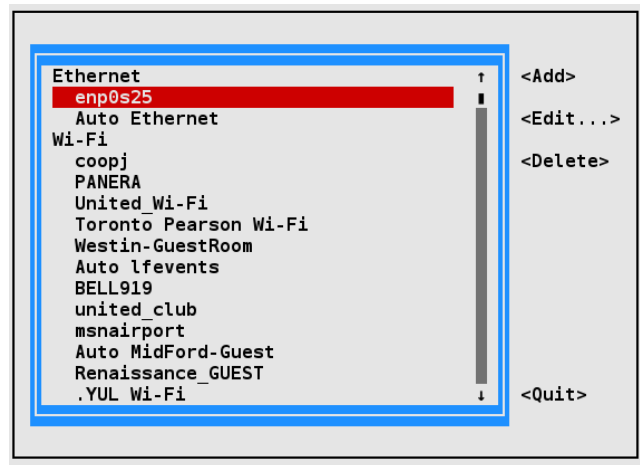


図 30.4 : nmtui 編集画面

# nmtui ワイヤレスコンフィギュレー

The screenshot shows the 'Edit Connection' window in nmtui. At the top, there's a title bar with 'Edit Connection'. Below it, the 'Profile name' is 'Troisieme Etage' and the 'Device' is 'A0:F8:A4:35:E7:73 (wlp3s0)'. A tree view on the left shows 'WI-FI' selected. The main area contains fields for 'SSID' (Monster-that-devoured-cleveland), 'Mode' (<Client>), 'Security' (<WPA & WPA2 Personal>), 'Password' (with a 'Show password' checkbox), 'BSSID', 'Cloned MAC address', and 'MTU' (default). Below these are sections for 'IPv4 CONFIGURATION' and 'IPv6 CONFIGURATION', both set to '<Automatic>' with 'Show' buttons. At the bottom, there are checkboxes for 'Automatically connect' and 'Available to all users', both checked. The bottom right has '<Cancel>' and '<OK>' buttons.

```

Edit Connection

Profile name Troisieme Etage
Device A0:F8:A4:35:E7:73 (wlp3s0)

= WI-FI
 SSID Monster-that-devoured-cleveland
 Mode <Client>
 Security <WPA & WPA2 Personal>
 Password [] Show password
 BSSID []
 Cloned MAC address []
 MTU (default)

= IPv4 CONFIGURATION <Automatic> <Show>
= IPv6 CONFIGURATION <Automatic> <Show>

[X] Automatically connect
[X] Available to all users

<Cancel> <OK>

```

図 30.5 : nmtui ワイヤレス設定



# エヌエムシーリ

- **Network Manager**のコマンドラインインターフェイス
- インタラクティブモードも搭載
- <https://fedoraproject.org/wiki/Networking/CLI> をご参照ください。
- `man nmcli-examples` も参照してください。
- 演習でいくつかの例を試してみる

```
student@ubuntu: ~
student@ubuntu:~$ nmcli --help
Usage: nmcli [OPTIONS] OBJECT { COMMAND | help }

OPTIONS
 -t[erse] terse output
 -p[retty] pretty output
 -m[ode] tabular|multiline output mode
 -c[olors] auto|yes|no whether to use colors in output
 -f[ields] <field1,field2,...>|all|common specify fields to output
 -e[scape] yes|no escape columns separators in val
ues
 -a[sk] ask for missing parameters
 -s[how-secrets] allow displaying passwords
 -w[ait] <seconds> set timeout waiting for finishin
g operations
 -v[ersion] show program version
 -h[elp] print this help

OBJECT
 g[eneral] NetworkManager's general status and operations
 n[etworking] overall networking control
 r[adio] NetworkManager radio switches
 c[onnection] NetworkManager's connections
 d[evice] devices managed by NetworkManager
 a[gent] NetworkManager secret agent or polkit agent
 m[onitor] monitor NetworkManager changes

student@ubuntu:~$
```

図 30.6 : nmcli

# ルーティング

- パケットをネットワーク間で移動させる
- ルーティングテーブル
  - すべてのネットワークとホストへのパスを定義する
  - ローカルパケットを直接送信
  - ルーターに送信されるリモートトラフィック

```
student@ubuntu: /etc
student@ubuntu:/etc$ route -n
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
0.0.0.0 192.168.1.1 0.0.0.0 UG 100 0 0 ens33
169.254.0.0 0.0.0.0 255.255.0.0 U 1000 0 0 ens33
192.168.1.0 0.0.0.0 255.255.255.0 U 100 0 0 ens33
192.168.122.0 0.0.0.0 255.255.255.0 U 0 0 0 virbr0
student@ubuntu:/etc$ ip route
default via 192.168.1.1 dev ens33 proto static metric 100
169.254.0.0/16 dev ens33 scope link metric 1000
192.168.1.0/24 dev ens33 proto kernel scope link src 192.168.1.24 metric 100
192.168.122.0/24 dev virbr0 proto kernel scope link src 192.168.122.1 linkdown
student@ubuntu:/etc$
```

図 30.7 : routeとip routeの使用

## デフォルトルート

- ルーティングテーブルエントリにマッチしない場合に使用します。
- DHCPを使用して動的に取得することが可能
- 手動で設定可能（スタティック）
  - Red Hat 派生システム：GATEWAY=x.x.x.xのどちらかです。  
`/etc/sysconfig/network` または  
`/etc/sysconfig/network-scripts/ifcfg-ethX`
  - Debian 派生システム：gateway=x.x.x.x で。  
`/etc/network/interfaces`
- nmcliを使用することで、`/etc`のファイルを手作業で修正することを避けることができます：
 

```
$ sudo nmcli con mod virbr0 ipv4.routes 192.168.10.0/24 \
+ipv4.gateway 192.168.122.0
$ sudo nmcli con up virbr0
```

```
$ sudo route add default gw 192.168.1.10 enp2s0
```

```
$ ルート
```

| カーネルIPルーティングテーブル<br>デスティネーション | ゲート          | ジェンマスク        | フラグ | メトリック | リフ | 用途       |
|-------------------------------|--------------|---------------|-----|-------|----|----------|
| ウェイ                           |              |               |     |       |    |          |
| デフォルト                         | 192.168.1.10 | 0.0.0.0       |     |       |    |          |
| デフォルト                         | 192.168.1.1  | 0.0.0.0       |     |       |    |          |
|                               |              |               |     |       |    |          |
| 172.16.132.0                  | 0.0.0.0      | 255.255.255.0 | U   | 0     | 0  | 0 vmnet1 |
| 192.168.1.0                   | 0.0.0.0      | 255.255.255.0 | U   | 0     | 0  | 0 enp2s0 |
| 192.168.113.0                 | 0.0.0.0      | 255.255.255.0 | U   | 0     | 0  | 0 vmnet8 |

```
$ sudo route add default gw 192.168.1.1 enp2s0
```

## スタティックルート

- ルーターやルートが複数ある場合に、パケットの流れを制御するために使用します。
- インターフェースごとに定義される
- 永続的または非永続的であることができる
- ipを使用する（非永続的）
- あるいはしつこいくらいに：
  - Red Hatベースのシステムです：
   
`/etc/sysconfig/network-scripts/route-ethX`
  - Debianベースのシステムです：
   
`/etc/network/interfaces`
  - SUSEベースのシステムです：
   
`/etc/sysconfig/network/ifroute-eth0`
- nmcliを使って、`/etc`のファイルを手で変更しないようにする（演習は後述し

```
$ sudo ip route add 10.5.0.0/16 via 192.168.1.100
```

```
/etc/sysconfig/network-scripts/route-ethX
```

```
$ cat /etc/sysconfig/network-scripts/route-eth0
10.5.0.0/16 via 172.17.9.1
```

```
/etc/network/interfaces
```

```
iface eth1 inet dhcp
post-up route add -host 10.1.2.51 eth1
post-up route add -host 10.1.2.52 eth1
```

```
/etc/sysconfig/network/ifroute-eth0
```

```
192.168.1.150 192.168.1.1.255.255.255 eth0
10.1.1.150 192.168.233.1.1 eth0
10.1.1.0/24 192.168.1.1 - eth0
```

## チームと絆 ネットワークのインターフェイス

- TeamingとBondingインターフェースは、動的なネットワークインターフェースの構成を可能にし、ネットワークに高いスループットと追加の堅牢性を提供します。
  - ボンディング  
カーネルモジュールを使用して、ネットワークリンクのアグリゲーションを行い、フェイルオーバー、冗長化、パフォーマンス向上を実現します。
  - チーム編成  
teamingが**teamed**というユーザー空間サービスを使用する以外は**bonding**と同様です。



### チーム編成は非推奨とされています！

- Teaming設定方法は**非推奨**としてリストアップされており、将来的に削除される予定です。
- TeamingからBondingへの変換をスムーズに行うために、設定ファイルを変換する**team2bond**ユーティリティが用意されています。

## ネットワークインターフェイスのボ

**Bonding** インターフェイスの設定には、いくつかの方法があります：

- シスフ

**sys** 疑似ファイルシステムに直接変更を加えます。変更は保存されません。

- アイプルートツ

**ip** コマンドによる **ボンディング** をサポートします。詳細については、`man 8 ip-link` を参照してください。変更内容は保存されません。

- ネットワークマネージャ

**NetworkManager** のすべてのユーザーインターフェイスが **ボンディング** をサポートしています。変更が保存されます。

**NetworkManager** の設定ファイルは、お好みのテキストエディターで編集することも可能です。

## nmcliでボンディングネットワークインターフェースを作成する

nmcli でボンディングインターフェイスの設定を作成するには、いくつかの手順が必要です。最小限のステップとは

- **アダプタを特定する** : nmcli device status を使用します。
- **ボンディングデバイスを作成します** : nmcli connection addを使用します。
- **ボンドにインターフェイスをアタッチします** : nmcli connection addを使用して、アダプタからボンドデバイスへの接続を作成します。
- **ボンドアダプターをオンラインに設定します** : ボンドアダプターの nmcli connection up com- mand を発行します。
- **を再起動します** : 再起動は絶対に必要というわけではありませんが、強くお勧めします。再起動することで、周囲に残っている設定の断片を一掃することができます。

nmcli の 詳 細 に つ い て は 、  
<https://people.freedesktop.org/~lkundrak/nm-docs/nmcli.html> を  
 ご参照ください。

\$

### コマンドライン上

```
nmcli connection add type bond
 ⊕ con-name bond0 ⊕
 ifname bond0
 bond.options "mode=active-backup,miimon=1000"
nmcli connection add type ethernet slave-type bond
 ↙
 con-name bond0-port1
 | ifname
 enp0s20u1u1 |
 master bond0
nmcli connection add type ethernet slave-type bond ↘
 con-name bond0-port2
 ifname enp0s20u1u2 ⊕
 マスターボンド0
nmcli connection up bond0
nmcli device status reboot
nmcli デバイスの状態
```

<https://www.kernel.org/doc/Documentation/networking/bonding.txt>

## DNSと名前解決

- 名前解決：ホスト名からIPアドレスへの変換
  - 静的な名前解決
  - 動的な名前解決（**DNS**）
- 逆解決：IPアドレスからホスト名への変換
- コマンドラインから
  - \$ `dig linuxfoundation.org`
  - \$ `host linuxfoundation.org`
  - \$ `nslookup linuxfoundation.org`

`/etc/hosts`

•  
•  
•



## /etc/hosts

- ローカルまたは静的なホスト名解決に使用します。
- 孤立した小規模なネットワークに有効
- 通常、DNSの前に確認します。
- 優先順位は/etc/nsswitch.confで設定可能だが、現在ではあまり使われていない

student@ubuntu:/etc\$ ls -l host\* です。

```
-rw-r--r-- 1 root root 92 Oct 22 2015 host.conf
-rw-r--r-- 1 root root 7 4月21日08:46 ホスト名
-rw-r--r-- 1 root root 221 Apr 21 08:46 hosts
-rw-r--r-- 1 root root 411 Apr 20 17:14 hosts.allow
-rw-r--r-- 1 root root 711 Apr 20 17:14 hosts.deny
```

/etc/hosts

/etc/hosts



/etc/hosts

```
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.1.100 hans hans7 hans64
192.168.1.150 bethe bethe7 bethe64
192.168.1.2 hp-printer 192.168.1.10
test32 test64 oldpc
```

/etc/nsswitch.conf

/エトセトラ

/etc/hosts.deny /etc/hosts.allow です。

/etc/host.conf

# DNS

- ダイナミックネームサーバを使用
- `/etc/resolv.conf`
  - 特定のDo-Mainを指定して検索することができます。
  - 問い合わせるネームサーバーの厳密な順序を定義する。
  - 手動で設定することも、**DHCP**（Dynamic Host Configuration Protocol）などのサービスから更新することもできます。

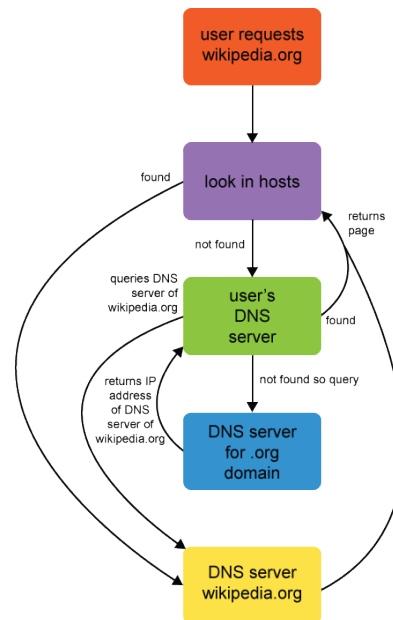


図30.8 : DNS

`/etc/hosts`

`/etc/resolv.conf`



## `/etc/resolv.conf`

検索 example.com aps.org ネームサーバー 192.168.1.1  
ネームサーバー 8.8.8.8

`/etc/resolv.conf`

# NetworkManagerによって生成される

192.168.1.1

## ネットワーク問題のトラブルシューティング

- IPコンフィグレーション
- ネットワークドライバ
- コネクティビティ
- デフォルトゲートウェイとルーティングの設定
- ホスト名解決

-

-

/sys/class/net

-

-

-

/proc    /sys    /proc/interrupts

## ネットワーク診断

- ビュッ
- トレースルート
- マートウル
- 掘る

```
$ ping -c 10 linuxfoundation.org
$ traceroute linuxfoundation.org です。
$ mtr linuxfoundation.org
```

## pingの例

```
[student@fedora ~]$ ping -c 10 linuxfoundation.org
PING linuxfoundation.org (23.185.0.4) 56(84) bytes of data.
64 bytes from 23.185.0.4 (23.185.0.4): icmp_seq=1 ttl=128 time=21.3 ms
64 bytes from 23.185.0.4 (23.185.0.4): icmp_seq=2 ttl=128 time=20.2 ms
64 bytes from 23.185.0.4 (23.185.0.4): icmp_seq=3 ttl=128 time=19.2 ms
64 bytes from 23.185.0.4 (23.185.0.4): icmp_seq=4 ttl=128 time=20.7 ms
64 bytes from 23.185.0.4 (23.185.0.4): icmp_seq=5 ttl=128 time=19.8 ms
64 bytes from 23.185.0.4 (23.185.0.4): icmp_seq=6 ttl=128 time=19.6 ms
64 bytes from 23.185.0.4 (23.185.0.4): icmp_seq=7 ttl=128 time=19.5 ms
64 bytes from 23.185.0.4 (23.185.0.4): icmp_seq=8 ttl=128 time=20.3 ms
64 bytes from 23.185.0.4 (23.185.0.4): icmp_seq=9 ttl=128 time=21.0 ms
64 bytes from 23.185.0.4 (23.185.0.4): icmp_seq=10 ttl=128 time=20.9 ms

--- linuxfoundation.org ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9012ms
rtt min/avg/max/mdev = 19.247/20.259/21.282/0.670 ms
[student@fedora ~]$
```

図30.9 : ping

## traceroute 例

```
c8:/tmp>traceroute google.com
traceroute to google.com (172.217.9.78), 30 hops max, 60 byte packets
 1 router (192.168.1.1) 0.340 ms 0.425 ms 0.544 ms
 2 * * *
 3 096-034-031-180.biz.spectrum.com (96.34.31.180) 19.900 ms 19.553 ms 18.011 ms
 4 acr06ftbgwi-gbe-8-46.ftbg.wi.charter.com (96.34.30.244) 19.682 ms 20.144 ms 19.848 ms
 5 cts03alxnmn-tge-3-0-0.alxn.mn.charter.com (96.34.25.137) 26.237 ms 26.465 ms 26.214 ms
 6 crr01rochmn-bue-2.roch.mn.charter.com (96.34.25.166) 26.445 ms 23.845 ms 25.653 ms
 7 crr01stcdmn-bue-6.stcd.mn.charter.com (96.34.25.0) 27.146 ms 21.003 ms 28.613 ms
 8 bbr02slidla-tge-0-1-0-6.slid.la.charter.com (96.34.1.188) 35.569 ms 33.988 ms 31.121 ms
 9 bbr02chcgil-bue-1.chcg.il.charter.com (96.34.1.149) 46.788 ms 46.397 ms 39.472 ms
10 prr01chcgil-bue-4.chcg.il.charter.com (96.34.3.11) 44.474 ms 44.081 ms 43.731 ms
11 096-034-152-159.biz.spectrum.com (96.34.152.159) 42.068 ms 096-034-152-155.biz.spectrum.co
m (96.34.152.155) 42.878 ms 096-034-152-159.biz.spectrum.com (96.34.152.159) 44.781 ms
12 108.170.243.193 (108.170.243.193) 40.572 ms 108.170.243.174 (108.170.243.174) 39.112 ms 1
08.170.243.193 (108.170.243.193) 42.198 ms
13 72.14.239.123 (72.14.239.123) 42.209 ms 39.873 ms 72.14.232.192 (72.14.232.192) 45.645 m
s
14 72.14.239.123 (72.14.239.123) 43.850 ms 108.170.244.2 (108.170.244.2) 43.267 ms 108.170.2
44.15 (108.170.244.15) 42.362 ms
15 ord38s09-in-f14.1e100.net (172.217.9.78) 42.122 ms 41.888 ms 40.801 ms
c8:/tmp>
```

図30.10 : トレースルート

## mtrの例

```

My traceroute [v0.92]
c8 (192.168.1.200) 2021-03-26T09:11:33-0500
Keys: Help Display mode Restart statistics Order of fields quit
 Packets Pings
 Host Loss% Snt Last Avg Best Wrst StDev
1. router 0.0% 76 0.3 0.3 0.2 0.4 0.0
2. ???
3. 096-034-031-180.biz.spectrum.com 0.0% 76 9.5 9.9 8.2 27.5 2.4
4. acr06ftbgwi-gbe-8-46.ftbg.wi.charter.com 0.0% 76 19.9 11.5 8.8 21.8 3.3
5. crr01onlswi-bue-402.onls.wi.charter.com 0.0% 76 12.8 14.0 11.9 31.9 3.0
6. crr01euclwi-bue-400.eucl.wi.charter.com 0.0% 76 14.2 16.0 13.4 32.6 3.2
7. bbr01euclwi-bue-100.eucl.wi.charter.com 0.0% 76 18.1 18.9 14.2 27.4 2.8
8. prr01mplsmn-bue-801.mpls.mn.charter.com 0.0% 75 20.8 28.8 19.7 505.9 55.9
9. 10ge5-18.core1.msp1.he.net 0.0% 75 20.9 26.3 19.3 44.5 7.6
10. 100ge11-2.core1.sea1.he.net 0.0% 75 89.3 77.4 71.3 101.7 6.6
11. 100ge15-1.core1.pdx1.he.net 0.0% 75 74.4 88.1 74.4 104.2 8.6
12. infinity-internet-inc.gigabitethernet2-11.core1 0.0% 75 74.5 77.0 74.5 89.2 2.9
13. xe-6-0-0.r-1.linuxfoundation.org 0.0% 75 77.1 77.0 74.5 88.1 3.0
14. kernel.org 0.0% 75 74.8 79.3 74.2 94.1 5.2

```

図30.11 : mtr



### ご注意ください

eth0とは異なるネットワークインターフェイス名を使用する必要があるかもしれません。この演習は、**nmtui**またはシステムのグラフィカルインターフェイスを使用して最も簡単に行うことができます。ここでは、コマンドラインによる解決策を紹介しますが、詳細については、あなたのディストリビューションのフレーバーやファッショ

- 1.
- 2.
- 3.
- 4.
- 5.



1. `$ ip addr show eth0`  
`$ ip route`  
`$ cp /etc/resolv.conf resolv.conf.keep`

```
$ ifconfig eth0
$ route -n
$ cp /etc/resolv.conf resolv.conf.keep
```

2. `sudo ip link set eth0 down`

```
sudo ifconfig eth0 down
```



### Red Hat / CentOSの場合



Red Hatベースのシステムでは、`/etc/sysconfig/network-scripts/ifcfg-eth0`に以下の記述があることを確認してください。

`etc/sysconfig/network-scripts/ifcfg-eth0` にあります。

```
DEVICE=eth0
BOOTPROTO=static
ONBOOT=yes
IPADDR=ステップ1から省略
NETMASK=ステップ1から省略
GATEWAY=ステップ1から省略
```





openSUSE, SLES, Debianベースのシステムで。

SUSE系では/etc/sysconfig/networkのファイルを同様に編集し、Debian系では/etc/network/interfacesをincludeに編集します。

**etc/sysconfig/network または /etc/network/interfaces** がある。

```
iface eth0 inet static
address not noted from step
1 netmask noted from step
1 gateway not noted from
step 1
```

### 3. \$ sudo ip link set eth0 up

```
sudo ifconfig eth0 up
```

```
$ sudo cp resolv.conf.keep /etc/resolv.conf
$ cat /etc/sysconfig/network
$ cat /etc/hosts
$ ping yourhostname
```

### 4. sudo reboot

```
$ ping ホスト名
```



### 1. /etc/hosts

### 2.

### 3. <http://winhelp2002.mvps.org/hosts2.htm> <http://winhelp2002.mvps.org/hosts.txt>



### 1. \$ sudo sh -c "echo 192.168.1.180 msystem.mydomain >> /etc/hosts" です。 \$ ping msystem.mydomain

### 2. \$ sudo sh -c "echo 127.0.0.1 ad.doubleclick.net >> /etc/hosts " ad.doubleclick.net に ping を送信します。

### 3. \$ wget http://winhelp2002.mvps.org/hosts.txt

```
-2014-11-01 08:57:12-- http://winhelp2002.mvps.org/hosts.txt 解決中
winhelp2002.mvps.org (winhelp2002.mvps.org)... 216.155.126.40
winhelp2002.mvps.org (winhelp2002.mvps.org)[216.155.126.40]:80 に接続中...接続済みです。HTTP
リクエストを送信しました、応答を待っています... 200 OK
長さ : 514744 (503K) [text/plain] 保存
先 : hosts.txt
```

```
100%[=====] 514,744 977KB/秒 0.5秒
```

2014-11-01 08:57:13 (977 KB/s) - hosts.txt を保存しました [514744/514744]。

```
$ sudo sh -c "cat hosts.txt >> /etc/hosts"
```



/dev

1.

```
$ sudo nmcli con
```

| 名称       | ユー                                         | TYPE           | DEVICE |
|----------|--------------------------------------------|----------------|--------|
| 自動イーサネット | 1c46bf37-2e4c-460d-8b20-421540f7d0e2       | 802-3-ethernet | ens33  |
|          | virbr0a84a332f-38e3-445a-a377-4363a8eb963f | bridge         | ヴィル    |

```
$ sudo nmcli con show "Auto Ethernet" | grep IP4.ADDRESS
```

|                 |                 |
|-----------------|-----------------|
| ip4.address[1]で | 172.16.2.135/24 |
|-----------------|-----------------|

```
$ nmcli con show 1c46bf37-2e4c-460d-8b20-421540f7d0e2
```

2.

```
$ sudo nmcli con modify "Auto Ethernet" +ipv4.addresses 172.16.2.140/24
```

3.

```
$ sudo nmcli con up "オートイーサネット"
```

接続が正常に起動した ( D-Busアクティブパス :  
 ↳ /org/freedesktop/NetworkManager/ActiveConnection/6) です。

```
$ ping -c 3 172.16.2.140
PING 172.16.2.140 (172.16.2.140) 56(84) バイトのデータです。
172.16.2.140 から 64 バイト: icmp_seq=1 ttl=64 time=0.038 ms
172.16.2.140 から 64 バイト: icmp_seq=2 ttl=64 time=0.034 ms
172.16.2.140 から 64 バイト: icmp_seq=3 ttl=64 time=0.032 ms

--- 172.16.2.140 ping の統計 ---
3/パケット送信、3/パケット受信、パケットロス0%、時間1998ms
rtt min/avg/max/mdev = 0.032/0.034/0.038/0.007ms
```

4.

```
$ sudo nmcli con modify "Auto Ethernet" -ipv4.addresses 172.16.2.140/24
```

```

```

```
$ sudo nmcli con up "オートイーサネット"
```

```

```



/dev

1.

\$ ルート

| カーネルIPルーティングテーブル<br>デスティネーション<br>ウェイ<br>デフォルト | ゲート<br>172.16.2.2 | ジェンマスク<br>0.0.0.0 | フラグ<br>ユニバーサル<br>リック | メト<br>100 | レフ<br>0 | ifaceを使用<br>する<br>0 ens33 |
|-----------------------------------------------|-------------------|-------------------|----------------------|-----------|---------|---------------------------|
| リンクローカル<br>172.16.2.0                         | *                 | 255.255.0.0       | U                    | 1000      | 0       | 0 ens33                   |
| 172.16.2.0                                    | *                 | 255.255.255.0     | U                    | 100       | 0       | 0 ens33                   |
| 192.168.122.0                                 | *                 | 255.255.255.0     | U                    | 0         | 0       | 0 virbr0                  |

\$ ip route

```
default via 172.16.2.2 dev ens33 proto static metric 100
169.254.0.0/16 dev ens33 scope link metric 1000
172.16.2.0/24 dev ens33 proto kernel scope link src 172.16.2.135 metric 100
192.168.122.0/24 dev virbr0 proto kernel scope link src 192.168.122.1 linkdown
```

2.

\$ sudo nmcli conn mod "Auto Ethernet" +ipv4.routes "192.168.100.0/24 172.16.2.1"

3.

\$ ルート

| カーネルIPルーティングテーブル<br>デスティネーション<br>ウェイ<br>デフォルト | ゲートウ<br>172.16.2.2 | ジェンマスク<br>0.0.0.0 | フラ<br>グ | メー<br>トル<br>100 | レフ<br>0 | 使用<br>する<br>0 | イフ<br>アス<br>三十三 |
|-----------------------------------------------|--------------------|-------------------|---------|-----------------|---------|---------------|-----------------|
| リンクローカル<br>172.16.2.0                         | *                  | 255.255.0.0       | U       | 1000            | 0       | 0             | 三十三             |
| 172.16.2.0                                    | *                  | 255.255.255.0     | U       | 100             | 0       | 0             | 三十三             |
| 192.168.122.0                                 | *                  | 255.255.255.0     | U       | 0               | 0       | 0             | ウィ<br>ルブ<br>ラス0 |

4.

\$ sudo nmcli conn up "オートイーサネット"

```
接続が正常に起動した (D-Busアクティブパス :
↳ /org/freedesktop/NetworkManager/ActiveConnection/25) です。
```

\$ ルート

|  |
|--|
|  |
|--|

| カーネルIPルーティングテーブル<br>デスティネーション      ゲート<br>ウェイ<br>デフォルト |  | ジェンマスク        | フラグ        | メトリック | レフ  | 使用する | インタフェース         |
|--------------------------------------------------------|--|---------------|------------|-------|-----|------|-----------------|
| 172.16.2.2                                             |  | 0.0.0.0       | ユニバーサルデザイン | 0     | 100 | 0    | 三十              |
| *<br>リンクローカル<br>172.16.2.0                             |  | 255.255.0.0   | U          | 1000  | 0   | 0    | 三十              |
| *<br>172.16.2.0                                        |  | 255.255.255.0 | U          | 100   | 0   | 0    | 三十              |
| 192.168.100.0      172.16.2.1                          |  | 255.255.255.0 | ユニバーサルデザイン | 100   | 0   | 0    | 三十              |
| 192.168.122.0      *                                   |  | 255.255.255.0 | U          | 0     | 0   | 0    | ヴィ<br>ルブ<br>ラス0 |

5.

\$ ルート

| カーネルIPルーティングテーブル<br>デスティネーション      ゲート<br>ウェイ<br>デフォルト |            | ジェンマスク        | フラグ        | メトリック       | lfaceを使用<br>する |
|--------------------------------------------------------|------------|---------------|------------|-------------|----------------|
| 172.16.2.2                                             | *          | 0.0.0.0       | ユニバーサルデザイン | 100      0  | 0 ens33        |
| リンクローカル<br>172.16.2.0                                  | *          | 255.255.0.0   | U          | 1000      0 | 0 ens33        |
| 192.168.100.0                                          | 172.16.2.1 | 255.255.255.0 | U          | 100      0  | 0 ens33        |
| 192.168.122.0                                          | *          | 255.255.255.0 | U          | 0      0    | 0 virbr0       |

```
$ sudo nmcli conn mod "Auto Ethernet" -ipv4.routes "192.168.100.0/24 172.16.2.1"
```

```
$ sudo nmcli conn up "オートイーサネット"
```

接続が正常に起動した ( D-Busアクティブパス :  
↳ /org/freedesktop/NetworkManager/ActiveConnection/3) で  
す。

\$ ルート

| カーネルIPルーティング     | ゲンマスク         | フラグ      | メトリッ   | lfaceを使用 |
|------------------|---------------|----------|--------|----------|
| デフォルト 172.16.2.2 | 0.0.0.0       | ユニバーサルデザ |        | する       |
| リンクローカル *        | 255.255.0.0   | U        | 1000 0 | 0 ens33  |
| 172.16.2.0 *     | 255.255.255.0 | U        | 100 0  | 0 ens33  |
| 192.168.122.0 *  | 255.255.255.0 | U        | 0 0    | 0 ens33  |

6.

```
$ sudo ip route add 192.168.100.0/24 via 172.16.2.1
```

```
$ sudo route
```

----

7.

```
$ nmcli con sho -a
```

| 名称            | ユー                                   | TYPE     | DEVICE |
|---------------|--------------------------------------|----------|--------|
| centos9stream | beb244d0-fb0e-3f4a-83a4-c700fd0fca7a | ethernet | ens33  |

```
$ nmcli con sho centos9stream| grep "'DHCP4"
```

|                      |                                          |
|----------------------|------------------------------------------|
| dhcp4.option[1] です   | ブロードキャストアドレス =                           |
| : dhcp4.option[2] です | 192.168.156.255                          |
| : dhcp4.option[3] です | dhcp_lease_time = 1800                   |
| : dhcp4.option[4] です | dhcp_server_identifier = 192.168.156.254 |
| : dhcp4.option[5] です | domain_name = localdomain                |
| : dhcp4.option[6] です | domain_name_servers = 192.168.156.2      |
| : dhcp4.option[7] です | 期限切れ = 1677701018                        |
| : dhcp4.option[8] です | ip_address = 192.168.156.128             |

|                      |                                               |
|----------------------|-----------------------------------------------|
| dhcp4.option[9]です    | requested_broadcast_address = 1               |
| : dhcp4.option[10]で  | requested_domain_name = 1                     |
| す : dhcp4.option[11] | requested_domain_name_servers = 1             |
| : dhcp4.option[12] : | requested_domain_search = 1                   |
| dhcp4.option[13] :   | requested_host_name = 1                       |
| dhcp4.option[14] :   | requested_interface_mtu = 1                   |
| dhcp4.option[15] :   | requested_ms_classless_static_routes = 1      |
| dhcp4.option[16]です   | requested_nis_domain = 1                      |
| : dhcp4.option[17] : | requested_nis_servers = 1                     |
| dhcp4.option[18] :   | requested_ntp_servers = 1                     |
| dhcp4.option[19] :   | requested_rfc3442_classless_static_routes = 1 |
| dhcp4.option[20] :   | requested_root_path = 1                       |
| dhcp4.option[21] :   | requested_routers = 1                         |
| dhcp4.option[22] :   | requested_static_routes = 1                   |
| dhcp4.option[23] :   | requested_subnet_mask = 1                     |
| dhcp4.option[24]です   | requested_time_offset = 1                     |
| : dhcp4.option[25]で  | requested_wpad = 1                            |
| す : dhcp4.option[26] | ルーター = 192.168.156.2                          |
| です :                 | subnet_mask = 255.255.255.0                   |

```
$ nmcli con down id centos9stream
$ nmcli con sho centos9stream| grep "DHCP4"
```

```
$ nmcli con up id centos9stream
```

接続が正常に起動した ( D-Busアクティブパス :  
/org/freedesktop/NetworkManager/ActiveConnection/5) です。

```
nmcli con sho centos9stream| grep "DHCP4"|grep expiry
```

|                   |        |
|-------------------|--------|
| dhcp4.option[6]です | 有効期限 = |
|-------------------|--------|





31.1 LDAP認証

31.2 研究室 \*\*



### リソース要件

この演習では、追加の仮想マシンをインストールする必要があります。これは、クライアント構成をテストするために使用される小型の**LDAPサーバー**です。LDAPアプライアンスは、約512MBのメモリと2GBのディスクスペースを使用します。この演習は、授業外で行ってください。

### 学習目標

このセッションが終わるころには、次のことができるようになっているはずです：

- LDAP Authenticationの目的を理解する。
- LDAPに関わる/etcの重要なファイルを知っておこう
- これらのファイルを操作するために利用できるユーティリティを知る。
- LDAPサーバーアプライアンスを使用した演習に取り組む。



### ご注意ください

\*\* これらのセクションは、一部または全部をオプションとして考慮することができる。これらのセクションには、背景となる参考資料、専門的なトピック、または高度なテーマが含まれています。インストラクターは、教室での経験や時間的制約に応じて、これらをカバーするかしないかを選択することができます。

## エルダップ

- ライトウェイトディレクトリアクセスプロトコル
- 認証サービス、ユーザー情報サービスを提供する
- ユーザー情報サービスのために **Kerberos** と一緒に使われることが多い
- 多くのシングルサインオンとアイデンティティ管理ソリューションで使用されています。

# LDAP認証

- 認証の一元化
- マルチドメイン
- マルチプラットフォーム
- マルチベンダー
- シングルサインオン
- PAMとsssdを使用

# LDAP認証の流れ

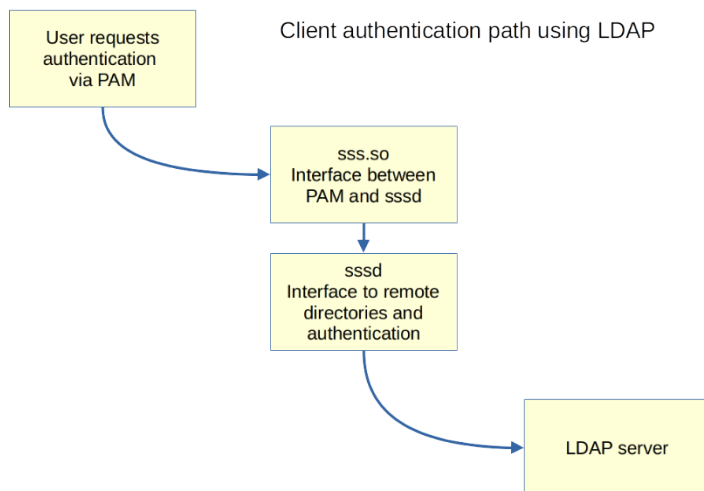


図 31.1 : LDAP クライアント認証プロセス

## /etc/sss/conf.d/00-sss.conf

```

[sss]
config_file_version = 2
domains = example.com
サービス = nss、pam、autofs

[domain/example.com]
enumerate = true
id_provider = ldap
autofs_provider = ldap
auth_provider = ldap
chpass_provider = ldap
ldap_uri = ldap://192.168.122.154/
ldap_search_base = dc=example,dc=com
ldap_id_use_start_tls = true
cache_credentials = True ldap_tls_reqcert
=allow

```

## LDAPとsssdのパッケージ

- **sssd** システムセキュリティサービスデーモン
- **sssd-ldap**は、LDAPと併用する場合のsssdの設定機構です。
- **sssd-tools** sssdで使用する各種ツールです。
- **oddjob-mkhomedir** ホームディレクトリを作成する PAM モジュール
- **ldap-utils** ユーザースペースユーティリティ - ldapから ユーザを追加・削除するツールが含まれており、**Ubuntu**に特化しています。
- **openldap-clients** ldap-utilsと同じで、以下の場合のみ。**CentOS**

/etc/sss/conf.d/00-sss.conf  
/etc/pam.d/common-session.conf

- 

- 



### リソース要件

この演習では、追加の仮想マシンをインストールする必要があります。これは、クライアント構成をテストするために使用される小型の**LDAPサーバー**です。LDAPアプライアンスは、約512MBのメモリと2GBのディスクスペースを使用します。この演習は、授業外で行ってください。

<https://training.linuxfoundation.org/cm/LFS207>

<https://www.turnkeylinux.org/openldap>

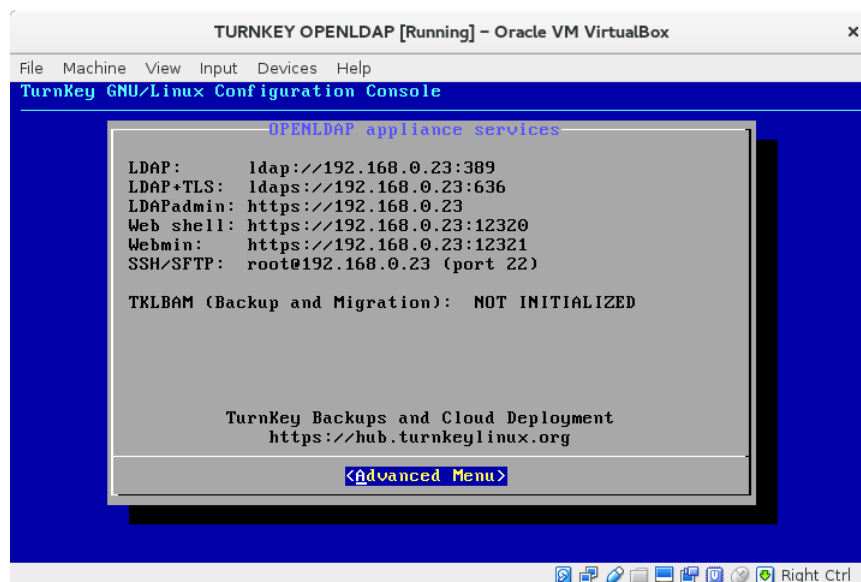


図31.2 : ターンキー型OpenLDAPアプライアンスサービス

1.
  - [turnkey-openldap-14.1-jessie-amd64.ova](#)
  - 
  - 
  - 
  -
- 2.
- 3.
-

-  
-  
-



### 非常に重要

このラボでは、LDAP サーバーとの通信にクリアテキストを使用します。暗号化キーの使用は、このコースの範囲外です。クリアテキストは、どのような種類の本番環境でも**推奨されるものではありません**。ラボ環境に対応するため、証明書の検証を無効にします。キーを検証しない機能は、ラボでは便利ですが、本番システムでこの技術を使用しないでください。

### ready-for.sh

1.



```
apt install sssd sssd-ldap sssd-tools oddjob-mkhomedir ldap-utils
```



```
yum install sssd sssd-ldap sssd-tools oddjob-mkhomedir openldap-clients
```

2.

```
ldapsearch -x -H ldap://192.168.0.23 -b "dc=example,dc=com" -s sub "objectclass=*"
```

```
extended LDIF
#
一、LDAPv3
スコープサブツリーを持つベース
<dc=example,dc=com> # filter: (objectclass=*)
request : ALL

example.com
dn: dc=example,dc=com
objectClass: top
objectClass: dcObject
objectClass: organization
o: example.com
dc: 例

admin, example.com
dn: cn=admin,dc=example,dc=com
objectClass: simpleSecurityObject
objectClass: organizationalRole
```

```
cn: admin
を記述する : LDAP管理者

グループ、example.com
dn: ou=Groups,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou : グループ

ユーザー、グループ、example.com
dn: cn=users,ou=Groups,dc=example,dc=com
cn: users
gidNumber: 100
objectClass: posixGroup
objectClass: top

ユーザー、example.com
dn: ou=Users,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou : ユーザー

ホスト、example.com
dn: ou=Hosts,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou : ホスト

ldmaps, example.com
dn: ou=ldmaps,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou : イドマップ

samba, example.com
dn: cn=samba,dc=example,dc=com
cn: samba
objectClass: simpleSecurityObject
objectClass: organizationalRole
description : SAMBAアクセスアカウント

エイリアス、example.com
dn: ou=Aliases,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou : エイリアス

nssspam, example.com
dn: cn=nssspam,dc=example,dc=com
cn: nssspam
objectClass: simpleSecurityObject
objectClass: organizationalRole
description : NSS/PAMアクセスアカウント

検索結果検索2
結果 : 0 成功

numResponses : 11
numEntries : 10
```



## 3. groups.ldif

```
ldapadd -x -D "cn=admin,dc=example,dc=com" -W -H ldap://192.168.0.23 -f groups.ldif
```

LDAP Passwordを入力します :

新しいエントリ "cn=user1,ou=Groups,dc=example,dc=com" を追加する。

## 4.

```
ldapsearch -x -H ldap://192.168.0.23 -b "dc=example,dc=com" -s sub "objectclass=*"
```

```
拡張LDIF
#
一、LDAPv3
スコープサブツリーを持つベース
<dc=example,dc=com> # filter: (objectclass=*)
request : ALL

example.com
dn: dc=example,dc=com
objectClass: top
objectClass: dcObject
objectClass: organization
o: example.com
dc: 例

admin, example.com
dn: cn=admin,dc=example,dc=com
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
を記述する : LDAP管理者

グループ、example.com
dn: ou=Groups,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou : グループ

ユーザー、グループ、example.com
dn: cn=users,ou=Groups,dc=example,dc=com
cn: users
gidNumber: 100
objectClass: posixGroup
objectClass: top

ユーザー、example.com
dn: ou=Users,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou : ユーザー

ホスト、example.com
dn: ou=Hosts,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou : ホスト

ldmaps, example.com
dn: ou=ldmaps,dc=example,dc=com
objectClass: organizationalUnit
```

```

objectClass: top
ou : Idmaps

samba, example.com
dn: cn=samba,dc=example,dc=com
cn: samba
objectClass: simpleSecurityObject
objectClass: organizationalRole
description : SAMBAアクセスアカウント

エイリアス、example.com
dn: ou=Aliases,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou : エイリアス

nssspam, example.com
dn: cn=nssspam,dc=example,dc=com
cn: nssspam
objectClass: simpleSecurityObject
objectClass: organizationalRole description
: NSS/PAMアクセスアカウント

luser1, Groups, example.com
dn: cn=luser1,ou=Groups,dc=example,dc=com
cn: luser1
objectClass: posixGroup
objectClass: top
gidNumber: 999001
memberUid: luser1

検索結果検索2
結果 : 0 成功

numResponses : 12
numEntries : 11

```

## 5. ユーザー.ldif

```
ldapadd -x -D "cn=admin,dc=example,dc=com" -W -H ldap://192.168.0.23 -f users.ldif
```

LDAP Passwordを入力します :  
新しいエントリ "cn=luser1,dc=example,dc=com" の追加

## 6.

```
ldapsearch -x -H ldap://192.168.0.23 -b "dc=example,dc=com" -s sub "objectclass=*"
```

```

extended LDIF
#
ー、LDAPv3
スコープサブツリーを持つベース
<dc=example,dc=com> # filter: (objectclass=*)
request : ALL

example.com
dn: dc=example,dc=com

```

```
objectClass : top
objectClass : dcObject
objectClass :
organization o :
example.com
dc: 例

admin, example.com
dn: cn=admin,dc=example,dc=com
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
を記述する : LDAP管理者

グループ、example.com
dn: ou=Groups,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou : グループ

ユーザー、グループ、example.com
dn: cn=users,ou=Groups,dc=example,dc=com
cn: users
gidNumber: 100
objectClass: posixGroup
objectClass: top

ユーザー、example.com
dn: ou=Users,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou : ユーザー

ホスト、example.com
dn: ou=Hosts,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou : ホスト

ldmaps, example.com
dn: ou=ldmaps,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou : イドマップ

samba, example.com
dn: cn=samba,dc=example,dc=com
cn: samba
objectClass: simpleSecurityObject
objectClass: organizationalRole
description : SAMBAアクセスアカウント

エイリアス、example.com
dn: ou=Aliases,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou : エイリアス

nssspam, example.com
dn: cn=nssspam,dc=example,dc=com
cn: nssspam
objectClass: simpleSecurityObject
```

objectClass: organizationalRole  
description : NSS/PAMアクセスアカウン  
ト

```
luser1, Groups, example.com
dn: cn=luser1,ou=Groups,dc=example,dc=com
cn: luser1
objectClass: posixGroup
objectClass: top
gidNumber: 999001
memberUid: luser1

luser1, example.com
dn: cn=luser1,dc=example,dc=com
uid: luser1
cn: luser1
givenName: luser1
sn: linux
homeDirectory :
/home/users/luser1 objectClass:
inetOrgPerson objectClass:
posixAccount objectClass: top
uidNumber: 999001
gidNumber: 999001

検索結果検索2
結果 : 0 成功

numResponses : 13
numEntries : 12
```

7.

```
yum install wireshark-gnome
wireshark
```

8.

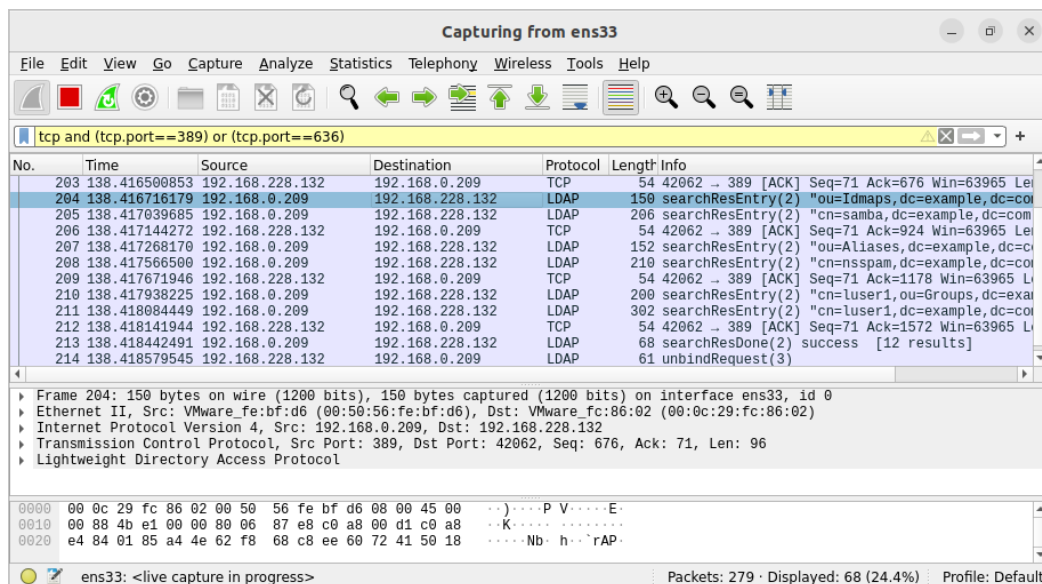


図31.3 : WiresharkのLDAPパケットトレース



### 非常に重要

CentOS-8やUbuntu 20.04から、リモートディレクトリへのアクセスや認証の管理に**System Security Services Daemon ( sssd )**が使用されるようになりました。**sssd**の使い方や設定方法は、参加ディストリビューションで共通となっています。

1.

```
apt install sssd sssd-ldap ldap-utils oddjob-mkhomedir
```

2.



### /etc/sss/conf.d/00-sss.conf

```
[sss]
config_file_version = 2
domains = example.com
サービス = nss, pam, autofs

[domain/example.com]
enumerate = true
id_provider = ldap
autofs_provider = ldap
auth_provider = ldap
chpass_provider = ldap
ldap_uri = ldap://192.168.122.154/
ldap_search_base = dc=example,dc=com
ldap_id_use_start_tls = true
cache_credentials = True
ldap_tls_reqcert =allow
```

3.

```
chmod 600 /etc/sss/conf.d/00-sss.conf
chown root.root /etc/sss/conf.d/00-sss.conf
```

4.

```
/etc/pam.d/common-session
```



### /etc/pam.d/common-session.conf

```
セッション必須 pam_unix.so
セッションオブ pam_oddjob_mkhomedir.so
セッション セッション pam_sss.so
```

5.

```
systemctl restart sssd oddjobd
systemctl enable sssd oddjobd
```

6.

7.

```
getent passwd luser1
```

```
luser1*:999001:999001:luser1:/home/users/luser1:
```

8.

```
ssh luser1@localhost
```

9.

```
-
-
-
```

10.

**非常に重要**

証明書の検証を無効にすることは、本番システムにおいて推奨されません ( NOT RECOMMENDED )。

```
/etc/sss/sss.conf
```

```
サスド
```

```
systemctl restart sssd
```

11.

```
getent passwd luser1
```

12.

```
ssh luser1@localhost
```

13.

```
-
-
-
```

14.



- 32.1 ファイアウォール
- 32.2 インターフェイス
- 32.3 ファイアウォール
- 32.4 ゾーン
- 32.5 ソース管理
- 32.6 サービス・ポート管理
- 32.8 研究室

### 学習目標

このセッションが終わるころには、次のことができるようになっているはずです：

- ファイアウォールとは何か、なぜ必要なのかを理解する。
- コマンドラインとグラフィカルインターフェースの両方で利用できるツールについて知っている。
- `firewalld`と`firewall-cmd`プログラムについて説明する。
- ゾーン、ソース、サービス、ポートの扱い方を知っている。





## ファイアウォールとは？

- 現代のネットワークに欠かせないセキュリティ設備
- システムおよびネットワークへの入退室を管理する。
- 侵入などの攻撃ベクトルから守る
- 特定のインターフェースやアドレスの信頼度を制御する
- 信頼度やネットワークトポグラフィーに応じた柔軟なバリアーをルールで制御

## パケットフィルタリング

- ファイアウォール構築の基礎となる
- ネットワークストリームの各パケットを検査する
- システムで定義されたルールに従う
- アクションには、受け入れる、ドロップする、マングルする、リダイレクトする、などがあります。

-  
-  
-

-  
-  
-  
-  
-

## ファイアウォールの世代

- **パケットフィルタリング**：各パケットを検査し、ドロップ、リジェクト、送信を行う。
- **ステートフルフィルタ**：接続の状態を調べ、新しい接続の一部か、既存の接続か、あるいは何もないかを考慮する。
- **アプリケーションレイヤーファイアウォール**：アプリケーションとプロトコルのパケットを認識し、関連付けます。

## ファイアウォールインターフェイス

- コマンドラインツール :
  - アイピーテーブル
  - ファイアウォールコマンド
  - ウフ
- グラフィカルなインターフェイス
  - システムコンフィグファイアーウォール
  - ファイアウォール設定
  - グフ
  - ヤスト
- その他、shorewallなどのユーティリティが存在します。

•

•

•

•

•

/エトセト  
ラ

## iptablesと連携しない理由

- 現在、ファイアウォールの導入の多くは、実はユーザー側で**iptables**パッケージを使用しています。
- これは現在、**firewalld**と同じカーネルファイアウォール実装コードをインターフェイスにしています。
- **iptables**は、有用な機能を得るまでに多くの時間を必要とするため、教えないことにしました。
- ただし、**iptables**については、**Linux Foundation**のシステム管理者向け講座の次のコースで詳しく説明します：

### LFS311システムエンジニア向けLinux

- 見る <https://training.linuxfoundation.org/linux-courses/>  
システム管理者研修／システムエンジニア向けリナックス

## firewalldとfirewall-cmd

- ダイナミックファイアウォールマネージャー(**firewalld**)
- ゾーンで動作する
- IPv4およびIPv6に対応
- ランタイムコンフィグレーションとパーマネントコンフィグレーションの分離
- 主なツールは**firewall-cmd**
- **iptables**ユーティリティを置き換えます。
- **systemd**の世界の一部
- 主要ディストリビューションで徐々に採用されつつあり、ソースからインストールすることもできる

`/etc/firewalld``/usr/lib/firewalld``/etc/firewalld``$ firewall-cmd --help`

使用法 : `firewall-cmd [OPTIONS...]`  
 を使用します。

....

|                        |                      |
|------------------------|----------------------|
| ステータスオプション             | firewalldの状態を返す・表示する |
| --state                | ファイアウォールを再読み込みし、状態情  |
| -完全な再読み込み              | ファイアウォールを再読み込みし、状態情  |
| 報を緩める                  |                      |
| --runtime-to-permanent | ランタイム設定からパーマネントを作成する |

## firewalld サービスステータス

- **firewalld**サービスは、他のサービスと同様に、有効/無効、開始/停止を行う必要があります :  
`sudo systemctl [enable/disable] firewalld`  
`sudo systemctl [start/stop] firewalld`
- ステータスを表示します :  
`$ sudo systemctl status firewalld`  
`$ sudo firewall-cmd --state`
- 複数のネットワークインターフェイスを使用する場合は、**IP Forwarding**をオンにする必要があります :  
`$ sudo sysctl net.ipv4.ip_forward=1`  
`$ echo 1 > /proc/sys/net/ipv4/ip_forward`



```
$ sudo systemctl status firewalld
```

```
firewalld.service - firewalld - ダイナミックファイアウォールデーモン
Loaded: ロード (/usr/lib/systemd/system/firewalld.service; enabled)
Active: アクティブ (running) since Tue 2015-04-28 12:00:59 CDT; 5分前
メインPID : 777 (firewalld)
...
```

```
$ sudo firewall-cmd --state
```

```
ランニ
```

```
$ sudo sysctl net.ipv4.ip_forward=1
```

```
$ echo 1 > /proc/sys/net/ipv4/ip_forward
```

[/etc/sysctl.conf](#)

## ゾーン

- 各ゾーンには信頼度やパケット管理動作が定義されています：
  - **drop** すべての受信パケットをドロップ
  - **ブロック** すべての受信ネットワーク接続を拒否する
  - **public** デフォルトで信頼なし
  - マスカレード付き外部公開
  - **dmz** (Demilitarized Zone) 一部のサービスを一部の接続に許可する。
  - **work** 信頼できる（やや）特定のネットワーク接続。
  - **home** 特定のネットワーク接続を信頼する（ほとんど）。
  - **インターナル ワークゾーン** に似ている。
  - **trusted** すべての接続を許可する。

•  
•  
•  
•  
•  
•  
•  
•  
•  
•

## ゾーンマネジメントの事例 I

- デフォルトゾーンを取得する：

```
sudo firewall-cmd --get-default-zone
```

おもてだった

- 現在使用されているゾーンのリストを入手する：

```
$ sudo firewall-cmd --get-active-zones
```

公開

インターフェース：eno16777736

- 利用可能なすべてのゾーンを一覧表示します：

```
$ sudo firewall-cmd --get-zones
```

ブロック DMZ ドロップ 外部 ホーム 内部 パブリック 信頼できる 仕事

```
$ firewall-cmd --help
```

-----  
ゾーンオプション

--get-default-zone コネクションおよびインターフェースのデフォルトゾーンを印刷する  
--set-default-zone=<zone>とする。

デフォルトゾーンを設定する

--get-active-zones 現在アクティブなゾーンを印刷する

--get-zones あらかじめ設定されたゾーンを印刷する [ P ]。

--get-services あらかじめ設定されたサービスを印刷する [ P ]。

--get-icmptypes 定義済みのicmptypesを印刷する [ P ]。

--get-zone-of-interface=<interface> です。

インターフェイスがバインドされているゾーンのプリント名 [ P ]。

--get-zone-of-source=<source>[/<mask>] です。

ソース[/mask]がバインドされているゾーンの名前を印刷する [ P ]。

--list-all-zones すべてのゾーンに追加されたもの、または有効なものをリストアップする [ P ]。

--new-zone=<zone> 新しいゾーンを追加する [ Pのみ ]。

--delete-zone=<zone> 既存のゾーンを削除する [ Pのみ ]。

--zone=<zone> オプションの設定や問い合わせにこのゾーンを使用、それ以外はデフォルトゾーン [ Z ] で作成されたオプションに使用可能

--get-target ゾーンターゲット [ P ] [ Z ] を取得する

--set-target=<target>とする。

ゾーンターゲットを設定する [ P ] [ Z ]

## ゾーンマネジメント事例 II

- デフォルトゾーンを**trusted**に変更し、その後元に戻す場合：

```
sudo firewall-cmd --set-default-zone=trusted
```

成功

```
sudo firewall-cmd --set-default-zone=public 成
```

功。

- インターフェイスを特定のゾーンに一時的に割り当てるには、次のようにします：

```
$ sudo firewall-cmd --zone=internal --change-interface=en01。
```

成功

- 恒久的に行うこと：

```
$ sudo firewall-cmd --permanent --zone=internal --change-interface=en01
```

成功

で、`/etc/firewalld/zones/internal.xml`というファイルが作成されます。

`man firewalld-cmd`

## ゾーンマネジメント事例集III

- 特定のインターフェースに関連するゾーンを確認する：

```
$ sudo firewall-cmd --get-zone-of-interface=eno1
public
```

- 特定のゾーンに関するすべての詳細を取得する：

```
sudo firewall-cmd --zone=public --list-all
public (アクティブ)
 target: default
 icmp-block-inversion: no
 interface: eno1 sources
 :
 services: chromecast libvirt libvirt-tls nfs nfs3 rsyncd ssh
 ports :
 protocols:
 masquerade: no
 forward-ports:
 source-ports:
 icmp-blocks:
 rich rules :
```

## ソース管理

- ゾーンをインターフェースだけでなく、特定のネットワークアドレスにバインドする。
- 関連するアドレスからのパケットでないパケットは、デフォルトゾーンに移動します。
- **firewall-cmd**でアサインします：

```
$ sudo firewall-cmd --permanent --zone=trusted --add-source=192.168.1.0/24
```

成功

•  
•

```
$ sudo firewall-cmd --permanent --zone=trusted --add-source=192.168.1.0/24
```

成功

```
$ sudo firewall-cmd --permanent --zone=trusted --list-sources 192.168.1.0/24
```

## サービスマネジメント

- 利用可能なサービスをご覧ください：

```
$ sudo firewall-cmd --get-services
```

- ゾーンでそれらを見る：

```
$ sudo firewall-cmd --list-services --zone=public
```

- ゾーンにサービスを追加する：

```
$ sudo firewall-cmd --permanent --zone=home --add-service=dhcp
```

成功

```
$ sudo firewall-cmd --reload
```

```
$ sudo firewall-cmd --get-services
```

```
RH-Satellite-6 amanda-client bacula bacula-client dhcp dhcpv6 dhcpv6-client dns ftp high-availability
→ http https imaps ipp ipp-client ipsec kerberos kpasswd ldap ldaps libvirt libvirt-tls mdns mountd
→ ms-wbt mysql nfs ntp openvpn pmcd pmproxy pmwebapi pmwebapis pop3s postgresql proxy-dhcp radius
→ rpc-bind samba samba-client smtp ssh telnet tftp tftp-client transmission-client vnc-server
→ wbem-https
```

```
$ sudo firewall-cmd --list-services --zone=public
```

```
dhcpv6-client ssh
```

```
$ sudo firewall-cmd --permanent --zone=home --add-service=dhcp
```

```
成功
```

```
$ sudo firewall-cmd --reload
```

```
/etc/firewalld/services
```

## ポートマネジメント

- 港湾管理です :

```
sudo firewall-cmd --zone=home --list-ports
```

```
$ sudo firewall-cmd --zone=home --add-port=21/tcp
```



```
$ sudo firewall-cmd --zone=home --add-port=21/tcp
```

```
success
```

```
$ sudo firewall-cmd --zone=home --list-ports
```

```
21/tcp
```

ここで、`/etc/services`を見ると、ポート21が**ftp**に対応していることが確認できる：

```
$ grep " 21/tcp" /etc/services
```

```
ftp 21/tcp
```

## ネットワークアドレス変換 ( NAT )

ネットワークアドレス変換 ( NAT ) は、ファイアウォールが、パケットがファイアウォールを出入りする際に、パケットの送信元アドレスまたは宛先アドレスを変更するために使用する方法である。NATにはいくつかの種類があります：

- DNAT (宛先ネットワークアドレス変換)
- SNAT (ソースネットワークアドレストランスレーション)
- SNATのバリエーションであるマスカレード

DNAT、SNAT、Masqueradeについては、次ページ以降で説明します。

## ディーエヌエーティー

**DNAT ( Destination Network Address Translation )** は、パケットがインターネットからシステムに入る際に、エンドデスティネーション ( 「宛先」 IPアドレス )、場合によっては宛先ポートを変更する。

- パケットを送信する際には、そのパケットに提供されたパブリックIPアドレスが付与されます。

### DNS

- パブリックIPアドレスは、ファイアウォールのものになります
- パケットの宛先を変更することで、パケットがファイアウォールを通過して内部ネットワークに移動し、目的のサーバーに接続できるようにする必要があります。

**DNAT**ファイアウォールルールは、通常、**NAT**テーブルのプリルートチェーンに配置されます。これは、インターネットに面したアダプターや、ウェブサーバーなどのサービスで使用されるパブリックアドレスになります。

ファイアウォールは通常パススルー機器 ( ローカルサービスはほとんどない ) であるため、パケットを検査し、宛先ポートを調べて、適用できるルールがあるかどうかを確認します。私たちのルールは、パケットを以下のように検査します :

- 宛先IPアドレス : ファイアウォールのIPアドレスである必要があります。
- 宛先ポート番号 : サービスのポート番号になります ( http=80 )。
- **DNAT**ルールにより、宛先IPアドレスはファイアウォール内のWebサーバーのアドレスに変更されます。
- パケットはファイアウォールの向こうのウェブサーバーに**転送**されます
- Webサーバーはパケットを受け入れて処理し、返信を送信します。

## DNATの例

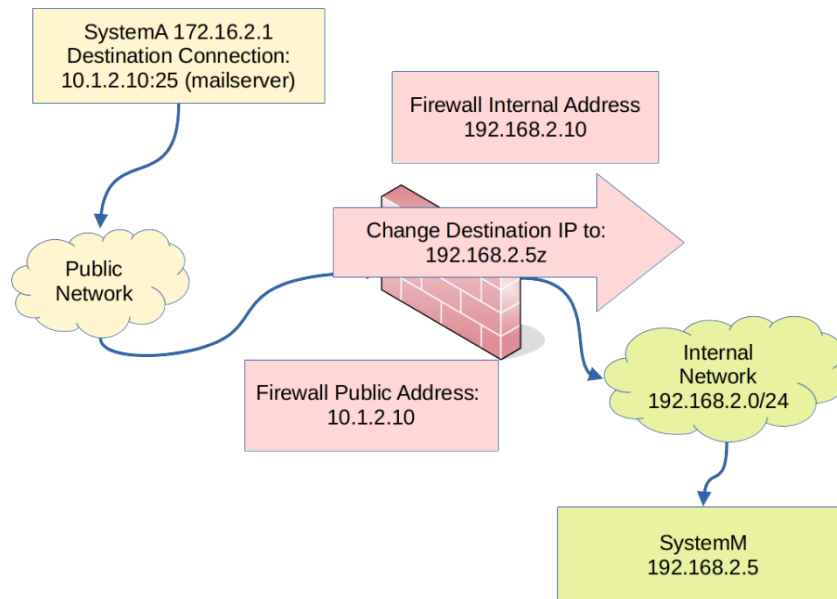


図32.1 : デスティネーションNAT

この図は、パケットがユーザーシステムからファイアウォールで保護されたネットワーク内のシステム上のサービスに至る典型的な経路を表しています。

- SystemA ( IP172.16.2.1 ) は、メール配信のためにsystemM ( IP10.1.2.25 ) のアドレスを調べます。
- 宛先ポート番号は、サービスのポート番号 ( smtp=25 ) になります。
- ファイアウォールのDNATルールは、宛先をファイアウォールの背後にあるメールサーバーのアドレス ( IP 192.168.2.5 ) に変更します。
- パケットはファイアウォールの向こうのメールサーバーに転送されます。
- メールサーバーは、パケットを受け入れて処理し、返信を送信します。

## SNATとマスカレード

ソースネットワークアドレス変換（**SNAT**）は、パケットがインターネットに向かう途中でシステムを離れる際に、パケット上のソース（「From」）IPアドレスを変更します。

- 内部ネットワークで使用するアドレスがインターネットに公開されないようにするため、トランスレーションが必要
- ほとんどの社内ネットワークはプライベートネットワークであり、ルータブルではない

- アウトバウンドインターフェースのIPアドレスがDHCPで提供されている場合、**SNAT**を実装するために**Masquerade**オプションが使用されます。
- masqueradeオプションは、**SNATよりも**多くの計算資源を必要とします。
- アウトバウンドインターフェースが静的な場合は、SNATを使用することができ、パブリックIPアドレスが指定されます。
- SNATとmasqueradeは、**NAT**テーブルのポストルーティングチェーンにあります。

## SNATの例

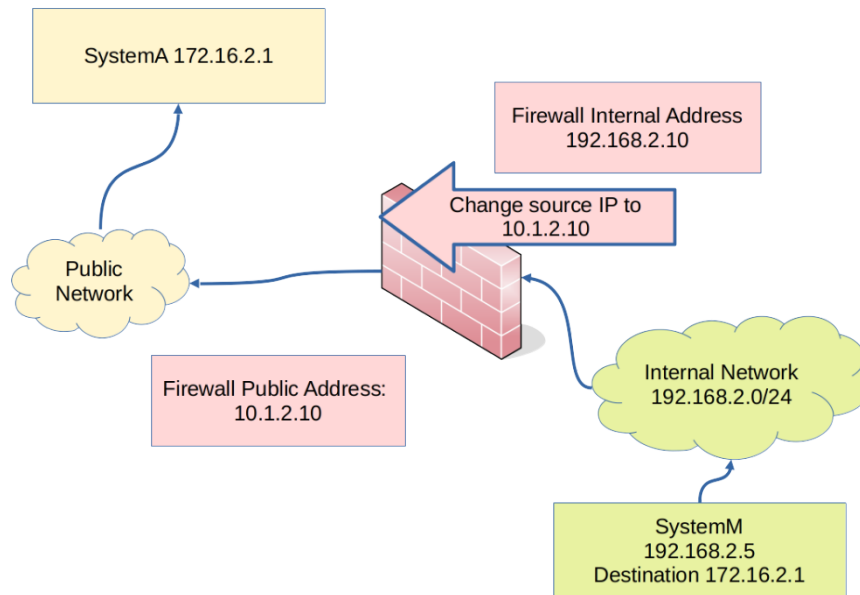


図32.2 : ソースNATとマスカレード

**SNAT**または**Masquerade**ファイアウォールルールは、通常、**NAT**テーブルのポストルーティングチェーンに配置されます。通常、ポストルーティングは、パケットがインターネットに向かう途中でシステムを離れる際の最後の検査または修正ポイントです。

- パケットの宛先アドレスは、接続を開始したリモートシステムのアドレスになります。
- の場合、パケットの送信元アドレスは、サービスの内部アドレスからファイアウォールのパブリックフェイスアダプターのアドレスに変更されます。

## 32.8 研究室

### 練習問題32.1 : firewalldのインストール

現在、ほとんどのLinuxディストリビューションでfirewalldパッケージ（多目的ユーティリティのfirewall-cmdを含む）が利用可能ですが、お使いのシステムにはインストールされていないかもしれません。

まず、すでにインストールされているかどうかを、以下のように確認します。

```
$ which firewalld firewall-cmd
```

```
/usr/sbin/firewalld
/usr/bin/firewall-cmd
```

プログラムを見つけることができなかった場合は、通常の方法でディストリビューションに応じて、次のいずれかを使用してインストールする必要があります：

```
$ sudo dnf install firewalld
sudo zypper install firewalld
$ sudo apt install firewalld
```

これが失敗した場合、firewalldパッケージはあなたのディストリビューションで利用できません。この場合、ソースからインストールする必要があります。

これを行うには、<https://fedorahosted.org/firewalld/>、gitソースリポジトリを取得するか、最新のtarballを簡単にダウンロードすることができます。

そして、ソースからインストールするための一般的な手順（現在のバージョンが何であれ使用する）を踏む必要があります：

```
$ tar xvf firewalld-0.3.13.tar.bz2
$ cd firewalld-0.3.13
$./configure
$ make
sudo make install
```

このソースはアンインストールターゲットも持っていることに注意してください：

```
sudo make uninstall
```

後悔のないように

./configure ステップで出てくる、ライブラリの不足などの不備に対処する必要があります。パッケージングシステムからインストールする場合は、ディストリビューションがこの問題を解決してくれますが、ソースからインストールする場合は、問題が発生することがあります。Linux Foundationのready-for.shスクリプトをシステムで実行したのであれば、問題が発生することはまずありません。

### 練習問題32.2: firewall-cmdの検証

firewalldパッケージの使い方については、まだ表面しか見ていません。ほとんど全てはfirewall-cmdは、非常に明確な名前のオプションを使用して、多種多様なタスクを実行する権限を与えられています。それを実感するためには、本当にただやることに越したことはない：

```
$ firewall-cmd --help
```

使用法 : firewall-cmd [OPTIONS...]を使用します。

-----

サービスオプション

--new-service=<service> です。

新しいサービスを追加する [ Pのみ ]。

--delete-service=<service> ( サービス削除 )。

削除して既存のサービス [ Pのみ ] を

-----



とありますが、**RHEL 8**のシステムで409行になるので、ここでは再現しません。

また、`/etc`にある様々な設定ファイルの説明や、ゾーンやサービスといった概念の説明など、他の **man** ページのリストもあります。

### 練習問題32.3: ゾーンにサービスを追加する

httpとhttpsのサービスをパブリックゾーンに追加し、現在表示されていることを確認する。

#### 解答 32.3

```
$ sudo firewall-cmd --zone=public --add-service=http
```

成功

```
$ sudo firewall-cmd --zone=public --add-service=https
```

成功

```
$ sudo firewall-cmd --list-services --zone=public
```

dhcpv6-client http https ssh

を実行していた場合は注意が必要です。

```
$ sudo firewall-cmd --reload
```

```
$ sudo firewall-cmd --list-services --zone=public
```

dhcpv6-client ssh

を追加した後、リストから消えてしまうのです！この不思議な動作は、新しいサービスを追加した後に、そのサービスがリストから消えてしまうためです。

サービスを追加する際に`--permanent`フラグを付け、`--reload`オプションは既知の永続的なサービスのみを再読み込みします。

### 練習32.4: ファイアウォールのGUIを使う

各ディストリビューションには、ファイアウォール管理のための独自のグラフィカルインターフェースがあります。**Red Hat** ベースのシステムでは **firewall-config** を実行し、**Ubuntu** では **gufw** と呼ばれ、**openSUSE** ではグラフィカル・メニュー・システムの **yast** の一部として見つけることができます。

私たちは、配布に柔軟に対応するため、単にコマンドラインのアプローチに集中してきました。しかし、ほとんどの比較的簡単なファイアウォール設定作業については、おそらくGUIからより少ない暗記で効率的に行うことができます。

ファイアウォールの設定GUIを起動したら、前回行ったパブリックゾーンにhttpとhttpsを追加する演習を行い、効果が出ていることを確認します。

グラフィカルなインターフェイスを理解するために、時間をかけるようにしてください。

### 練習32.5: 自分の本当のIPアドレスを決める

この演習では、**NAT**の最も一般的な使い方として、ファイアウォールの背後にあるデバイスの内部アドレスまたはプライベートアドレスを密閉する方法を説明します。

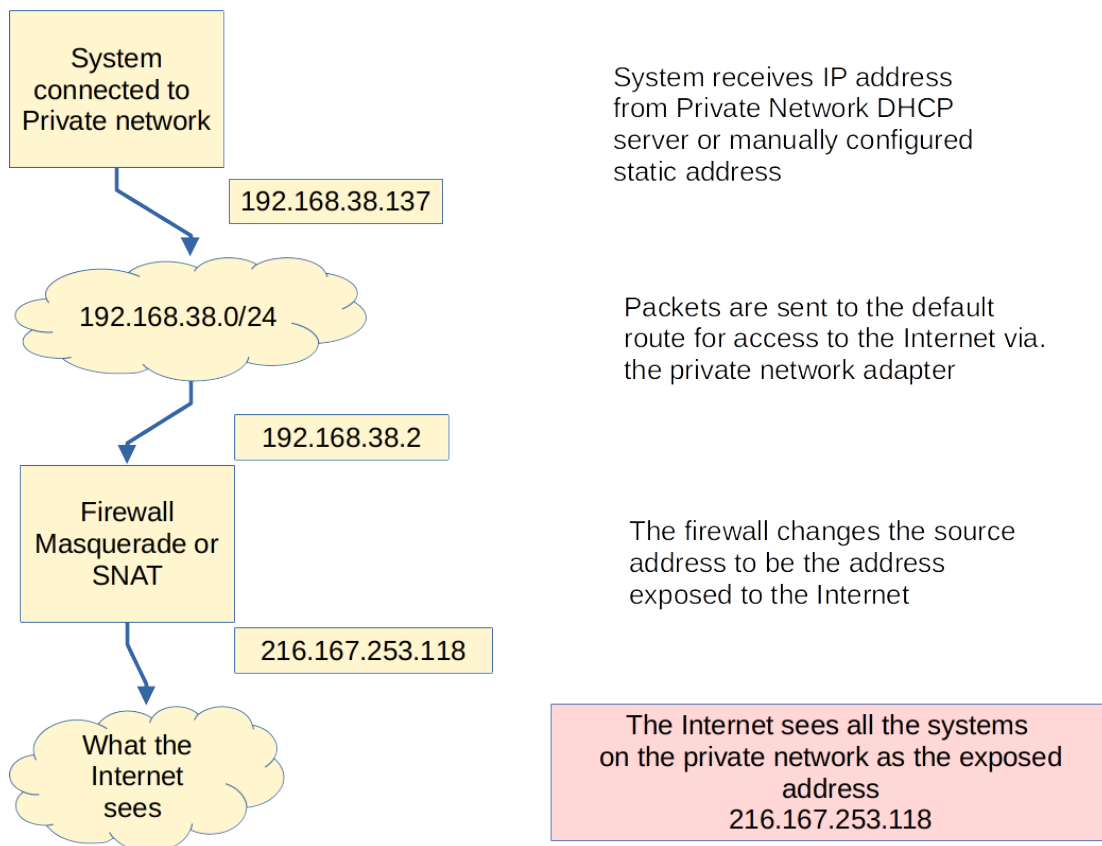


図32.3 : SNATまたはMASQUERADEの例

上図を参考に、システムで使用されている IP アドレスを決定してください。

1. Linux VM上で、ネットワークアドレスを取得します。ipコマンドでうまくいくので、以下のように必要のない余分な情報は排除することができます。

\$ sudo ip a | grep " inet " です。

```
inet 127.0.0.1/8 scope host lo
inet 192.168.38.137/24 brd 192.168.38.255 scope global dynamic noprefixroute ens33
```

表示されるアドレスは、ループバックと仮想マシンのIPアドレスです。

2. VMのアドレスは192.168.24.0/24のサブネットにあり、インターネット上の非ルート可能なアドレスとなります。プライベートインターネットのアドレス割り当てに関する追加情報については、以下のリンクを参照してください  
<https://datatracker.ietf.org/doc/html/rfc1918>  
 クライアントをネクストホップに接続するための次のIPアドレスを見つけるには、デフォルトルート調べます。

\$ sudo ip r  
 デフォルト via 192.168.38.2 dev ens33 proto dhcp metric 100

次のホップは、192.168.38.2というIPアドレスを持つシステムであることがわかります。

3. 192.168.38.2システムは、この環境の本当のファイアウォールであるため、私たちがログインすることはできません。私たちのシステムを世界がどのように見ているかを抽出することができます。

コマンドライン経由：

\$ sudo lynx -dump http://whatismyip.akamai.com | awk '{print \$1}' .

216.167.253.118

または、Googleで検索して、多くのグラフィカルなオプションから1つを選択します：

#### 私のIPアドレスは何ですか

外部との通信に使用されるアドレスは、プライベートアドレスのどれでもなく、プライベートネットワーク上のすべてのマシンが、外部からは1つのIPアドレスに見えることに注意してください。

プライベートIPアドレスを単一アドレスに変更する動作は、**Source Network Translation**の例です。

またはマスカレード

## 第33章

# システム初期化：systemd、SystemV、アップスタート



|      |                   |     |
|------|-------------------|-----|
| 33.1 | init プロセス.....    | 534 |
| 33.2 | スタートアップの代替品 ..... | 535 |
| 33.3 | システムディナー .....    | 536 |
| 33.4 | システムセクター .....    | 538 |
| 33.5 | 研究室.....          | 540 |

### 学習目標

このセッションが終わるころには、次のことができるようになっているはずです：

- イニシャルプロセスの重要性を理解する。
- SysVinitとUpstartに代わってsystemdが誕生した経緯を理解する。
- systemdの主な機能を説明する。
- systemctl を使って、systemd でサービスの設定や制御を行います。



## 33.1 init プロセス

# イニット

- `/sbin/init`
- 全プロセスの親
- ブートプロセスの残りの部分を調整する
- 最も一般的な3つの実装方法 :
  - システムディー
  - アップスタート
  - シスビニット
- すべての主要ディストリビューションが **systemd** に移行しました

`/sbin/init` (通常 `init` と呼ばれる) は、システム上で最初に実行されるユーザーレベルのプロセス (またはタスク) であり、システムがシャットダウンされるまで実行され続ける。伝統的に、このプロセスはすべてのユーザープロセスの親とみなされてきましたが、いくつかのプロセスはカーネルによって直接起動されるため、技術的にはそうではありません。

`init` は、ブートプロセスの後段を調整し、環境のあらゆる側面を設定し、システムへのログインに必要なプロセスを開始します。また、`init` はカーネルと密接に連携して、プロセスが終了したときに後始末をします。

従来、ほぼすべてのディストリビューションは、UNIXの由緒あるソフトウェアである **SysVinit** をベースに `init` プロセスを構築していました。しかし、この方式は数十年前にかなり異なる状況で開発されたものです :

- 対象はマルチユーザーメインフレームシステム (パーソナルコンピュータやノートパソコンなどは対象外) でした。
- ターゲットはシングルプロセッサのシステムでした。
- 起動 (およびシャットダウン) 時間は重要な問題ではなく、物事を正しく理解することよりもはるかに重要なことだったのです。

スタートアップは、一連の連続したステージ (ランレベル) に分けられ、各ステージが完了しないと次のステージに進めないシリアルプロセスとして捉えられていた。そのため、複数のプロセッサやコアで行う並列処理を利用することは容易でなかった。

第二に、シャットダウンや再起動は比較的稀な出来事であり、それにかかる正確な時間は重要視されませんでした。今日、Linuxシステムは通常数秒で起動します。

現代のシステムは、より高度な機能を備えた新しい手法を必要としています。

## 33.2 スタートアップの代替品

# スタートアップの代替品

- アップスタート
  - Ubuntuによって開発され、2006年に初めて搭載された
  - Fedora 9 (2008年)、RHEL 6とそのクローンで採用されました。
  - 様々な組み込み機器やモバイル機器にも採用された
- システムデーモン
  - Fedoraは2011年に主要ディストリビューションで初めて採用された
  - RHEL、SUSEに続く
  - Ubuntu 16.04ではUpstartがsystemdに置き換わっています。
  - 重要なLinuxディストリビューションはすべてsystemdをベースとするようになった

SysVinit の本質的な制限に対処するため、システムの起動を制御する新しい方式が開発されました。他にもありますが、エンタープライズのディストリビューターが採用したのは主に2つの方式、Upstartとsystemdでした。

他のすべての主要なLinuxディストリビューションは、今やsystemdを採用し、デフォルトとして構成しています。Ubuntuでさえ、段階的に

有利にアップスタート。

systemdへの移行は非自明であり、バグや機能不足は非常に大きな障害となるため、レガシーソフトウェアのために不可欠な互換性レイヤーが採用され、現在も存在しています。そのため、SysVinitユーティリティの互換性ラッパーはまだ残っています。

systemdの開発と採用の歴史はかなり複雑で、カラフルな個性があるため、すべての議論が友好的かつ技術的なものであったわけではありません。しかし、この聖戦は、私たちがここで関心を持つ限りでは、もう終わったことです。

以下では、systemdに集中し、SysVinitや、もはや重要な方法では使用されていないUpstartはほとんど無視することになります。

### 33.3 システムデー

## systemdの特徴

- 従来のイニシャルシステムより高速に起動
- 積極的な並列化機能を提供
- サービスの起動にソケットとD-Busの起動を使用します。
- シェルスクリプトをプログラムに置き換える
- デーモンのオンデマンド起動を提供します。
- **cgroups**を使用してプロセスを追跡する
- マウントポイント、オートマウントポイントを維持する
- 精巧なトランザクション依存ベースのサービス制御ロジックを実装しています。
- **SysVinit**のドロップイン代替品として動作し、互換性があります。  
**SysVinit**のスクリプトです。

Linuxのシステムおよびセッションマネージャである**systemd**は、現在、すべての主要なディストリビューションで主流となっています。

**systemd** は **SysVinit** と下位互換性があり、ランレベルの概念はランレベルターゲットを介してサポートされていることに注意してください。そのため

**telinit** プログラムは、ランレベルとともに動作するようにエミュレートされています。

**bash** スクリプトの代わりに、**systemd** は **.service** ファイルを使用します。さらに、**systemd** はすべてのデーモンをそれぞれの Linux **cgroup** に振り分けます。

(対照群)とした。



## systemd 設定ファイル

- `/etc/hostname` を置き換えた :
  - Red Hatベースのシステムでは、`/etc/sysconfig/network`。
  - SUSEベースのシステムでは、`/etc/HOSTNAME`。
  - Debianベースのシステムでは、`/etc/hostname` (すでに標準として採用されている) が使用されています。
- その他のファイルとしては、以下のようなものが考えられます :
  - `/etc/vconsole.conf`: デフォルトのキーボードマッピングとコンソールフォントを使用します。
  - `/etc/sysctl.d/*.conf`: カーネル `sysctl` パラメータのドロップインディレクトリです。
  - `/etc/os-release`: ディストリビューションIDファイル

**systemd** は標準化された設定ファイルのセットを使用することを好みますが、ディストリビューションに依存するレガシーな設定ファイルをフォールバックとして使用することができます。

例えば、仮想端末のデフォルトを設定する `/etc/vconsole.conf` は、Ubuntu システムでは表示されないなど、どの設定ファイルを使用するかは、各ディストリビューションの設定方法によって異なります。



### SysVinitとの互換性

**systemd** は **SysVinit** とほぼ後方互換性を持つように構成されているため、古いコマンドを使っても概ね動作します。ランレベルターゲットのメカニズムにより、ランレベルの使用を概念的にサポートします。また、**telinit** はランレベルとともに動作するようにエミュレートされています。

## 33.4 システムセクター

# システムセクター

\$ `systemctl` [オプション] コマンド [名前]。

- **systemd**が制御するすべてのものの状態を表示すること :

\$ `systemctl`

- 利用可能なサービスをすべて表示します :

\$ `systemctl list-units -t service --all`

- アクティブなサービスのみを表示する :

\$ `systemctl list-units -t サービス`

- 1台以上のユニットを起動 (アクティベート) させること :

`sudo systemctl start foo`

\$ `sudo systemctl start foo.service`

\$ `sudo systemctl start /path/to/foo.service`

ここで、ユニットは、サービスまたはソケットであることができる。

`systemctl`は、サービスを管理するための主要なユーティリティです。**systemctl**コマンドの中には、非rootユーザーで実行できるものもあれば、rootまたは**sudo**で実行する必要があるものもあります。

ほとんどのコマンドで、サービス名に付いている.serviceを省略することができます。

SysVinitから**systemd**に移行する方法については、[httpsのSysVinit to systemd Cheatsheet](https://fedoraproject.org/wiki/SysVinit_to_systemd_Cheatsheet)を参照してください :

[//fedoraproject.org/wiki/SysVinitから Systemd Cheatsheet](https://fedoraproject.org/wiki/SysVinit_to_systemd_Cheatsheet) へ。

## systemctl (続き)

- 停止（休止）させること：  
`$ sudo systemctl stop foo.service`
- サービスを有効化／無効化する：  
`$ sudo systemctl enable sshd.service`  
`$ sudo systemctl disable sshd.service`

これらのコマンドは、実際にサービスを開始または停止するのではなく、システム起動時にサービスを開始するかどうかを制御します。

ほとんどのコマンドで、サービス名に付いている.serviceを省略することができます。

## 33.5 研究室



### ビデオデモンストレーションのリソース

`systemctl_demo.mp4`を使用しています。

#### 練習問題33.1 : *systemd*で新しいスタートアップサービスを追加する

新しいスタートアップサービスを追加するには、`/etc/systemd/system`の直下に（`root`として）ファイルを作成するか、そのディレクトリツリーのどこかにファイルを作成する必要があります。例えば、`/etc/systemd/system/fake2.service`という非常に小さなファイル（ダウンロードしたSOLUTIONSファイルから`fake2.service`として解凍できます）には、次の内容が含まれています：

#### fake2. サービス

##### [ユニット]

```
Description=fake2
After=network.target
```

##### [サービス] のご案内 ]

```
ExecStart=/bin/sh -c '/bin/echo I am starting fake2 service ; /bin/sleep 30' です。
ExecStop=/bin/echo fake2サービスを停止しています。
```

##### [インストール] してください。 ]

```
WantedBy=multi-user.targetの略です。
```

さて、このユニットファイルにはいろいろなことが書かれています。After=network.targetは、ネットワークが起動してからサービスを開始することを意味し、WantedBy=multi-user.targetは、マルチユーザモードになったときに開始することを意味します。これは、**SysVinit**のランレベル2、3に相当します。なお、graphical.targetはランレベル5に相当する。

あとは、コマンドを発行するだけで、サービスの開始、停止、状態の確認ができます：

```
$ sudo systemctl start fake2.service
$ sudo systemctl status fake2.service
$ sudo systemctl stop fake2.service
```

もし、この作業中にユニットファイルをいじっていたら、リロードする必要があります：

```
sudo systemctl daemon-reload
```

という警告が表示されます。

出力に直接目を向けるには、こうすることができます：

```
$ sudo tail -f /var/log/messages
```

(**Ubuntuでは/var/log/syslogを使用**) サービスが実行されている間は、バックグラウンドまたは別のウィンドウで表示されます。システム起動時にサービスがON/OFFするように設定する：

```
$ sudo systemctl enable fake2.service
$ sudo systemctl disable fake2.service
```

もう一度、本当に再起動しないと効果が発揮されない。

## 第34章

# バックアップとリカバリーの方法



|      |                                 |     |
|------|---------------------------------|-----|
| 34.1 | バックアップの基本 .....                 | 542 |
| 34.2 | バックアップとアーカイブ .....              | 544 |
| 34.3 | バックアップの方法と戦略 .....              | 546 |
| 34.4 | タール .....                       | 549 |
| 34.5 | 圧縮 : gzip、bzip2、xz、バックアップ ..... | 552 |
| 34.6 | ディーディー .....                    | 553 |
| 34.7 | アールエスアイエヌシー .....               | 554 |
| 34.8 | バックアッププログラム**について .....         | 555 |
| 34.9 | 研究室 .....                       | 556 |

### 学習目標

このセッションが終わるころには、次のことができるようになっているはずです：

- バックアップが必要なデータを特定し、優先順位をつける。
- データのバックアップとアーカイブの違いを理解する。
- 状況に応じて、さまざまな種類のバックアップとリストアの方法を効率的に採用する。
- xzやgzipなど様々な圧縮方式を使い、アーカイブプログラムであるtarと効率よく組み合わせる。
- 生のデータをコピーする場合はddを、複数のマシンにまたがるバックアップや同期を効率的に行う場合はrsyncを使用します。
- 最も有名なバックアッププログラムについて説明する。



### ご注意ください

\*\* これらのセクションは、一部または全部をオプションとして考慮することができる。これらのセクションには、背景となる参考資料、専門的なトピック、または高度なテーマが含まれています。インストラクターは、教室での経験や時間的制約に応じて、これらをカバーするかしないかを選択することができます。

541



## 34.1 バックアップの基本

### なぜバックアップなのか？

- データは価値あるもの
- ハードウェアの故障
- ソフトウェアの不具合
- 人は過ちを犯すもの
- 悪意のある人が意図的に被害を与えることがある
- 不可解な出来事が起こる
- 巻き戻しは便利です

個人で管理しているシステムでも、多数のマシンで構成されるネットワークでも、システムのバックアップは非常に重要です。ディスク上のデータは、重要な仕事の成果物であり、それゆえに保護したい商品です。失われたデータの再作成には、時間とコストがかかります。また、データによってはユニークなものもあり、再作成する方法がない場合もあります。

記憶媒体の信頼性が向上する一方で、ドライブ容量も増加しています。バイトあたりの故障率が下がっても、予測できない故障は発生する。故障したドライブと故障するドライブの2種類しかない、というのは悲観的かもしれないが、本質的にはそうなのだ。**RAID**の利用は有効ですが、バックアップは必要です。

どんなソフトウェアも完璧ではありません。バグによってデータが破壊されたり、破損したりすることがあります。長く使われている安定したプログラムにも問題がある場合があります。

誰もが一度は隣の部屋から（あるいは自分の口から）**OOPS!**（あるいはもっとひどいもの）を聞いたことがあるはずです。単純な入力ミスが、ファイルやデータの大規模な破壊を引き起こすこともあります。

それは、典型的な不満を抱えた従業員であったり、主張のある外部のハッカーであったりします。セキュリティの懸念とバックアップ機能は、非常に強く関連しています。

ファイルは、いつ、誰が、どのようにして消したのか、わからないまま消えてしまうことがあります。システムの全部または一部の以前のスナップショットに復元することが必要な場合があります。



## 何をバックアップする必要があるの

- 間違いなく :
  - ビジネス関連のデータです。
  - システム構成ファイル。
  - ユーザーファイル ( 通常は `/home` 以下 ) 。
- そうかもしれませんね :
  - スプーリング・ディレクトリ ( 印刷用、メール用など )
  - ログファイル ( `/var/log` などにあります ) 。
- おそらく、そうではないでしょう :
  - 再インストールが容易なソフトウェア。よく管理されたシステムでは、これはほとんどすべてであるべきです。
  - `tmp` ディレクトリ
- 絶対にダメです :
  - `proc`、`/dev`、`/sys` などの疑似ファイルシステム、スワップ

当然ながら、組織にとって不可欠なファイルはバックアップが必要です。設定ファイルは頻繁に変更される可能性があり、各ユーザーのファイルと同様にバックアップが必要です。

ログファイルは、システムの履歴を調査する必要がある場合、特に侵入やその他のセキュリティ侵害を検出するために重要な役割を果たします。

簡単に再インストールできるものは、バックアップする必要はありません。また、スワップパーティション ( またはファイル ) や `proc` ファイルシステムは、これらの領域のデータは基本的に一時的なもの ( `/tmp` ディレクトリと同じ ) であるため、一般的にバックアップする有用性や必要性はありません。

## 34.2 バックアップとアーカイブ

# バックアップとアーカイブ

- すべてのバックアップメディアは、読み取り不可能になるまでの寿命が有限である
- 従来の推定値です：
  - 磁気テープ：10～30年
  - CD、/DVD：3～10年
  - ハードディスク：2～5年
- ライフタイムに非常に敏感
  - 環境条件（温度、湿度など）。
  - メディアの質
  - 現在のOSやハードウェアでデータを読み取ることができる、動作するソフトウェアを持つこと
- バックアップには十分な寿命であり、永久的なデジタルアーカイブには適さ

通常のバックアップ期間よりも長い寿命の場合、複数のコピーと新しいメディアへのコピーでデータを保存することができます。

非常に長い時間（つまり何十年、何百年など）、すべてが陳腐化する可能性があるため、標準的な方法は簡単に機能しない：

- ハードウェア
- ソフトウェアと文書フォーマット
- メディア

安価なデジタルフォーマットのどれも、実は紙やフィルムに長期間対抗できるものではありません（適切に保管され、継続的に手入れがされている場合-ワインのように）。

この問題は、深刻な人々が考えている問題であり、すべてが失われる前に、良い解決策が用意されているはずです。

## テープドライブ

- 比較的遅い
- シーケンシャルアクセスのみを許可する
- かつてのような一般的なものではありません
- 現在、プライマリバックアップに使用することは稀です
- オフサイトストレージやアーカイブのためにまだ時々有用である

テープドライブは、以前ほど一般的ではありません。テープドライブは比較的遅く、シーケンシャルなアクセスしかできません。最新のセットアップでは、プライマリバックアップに使用されることはほとんどありません。しかし、磁気テープドライブの寿命は、物理的な劣化がない限り有限である。しかし、磁気テープドライブは、物理的な劣化やデータの損失がない限り、常に有限の寿命しか持ちません。

最近のテープドライブは通常**LTO**（Linear Tape Open）タイプで、その最初のバージョンは1990年代後半にオープンスタンダードの代替品として登場し、初期のフォーマットはほとんどが独自仕様でした。初期の形式はほとんどが独自のものであった。

2.5TB以上を同サイズのカートリッジに入れた場合。

日々のバックアップは、通常**NAS**（Network Attached Storage）やクラウドベースのソリューションで行われるため、テープベースの新規インストールはあまり魅力的ではありません。しかし、それらはまだ見つかっており、システム管理者はそれらに対処する必要がある場合があります。

以下では、バックアップメディアの物理的な形態にこだわらず、より抽象的な話をすることにします。

### 34.3 バックアップの方法と戦略

## バックアップの方法

- **フルです：**  
システム上のすべてのファイルをバックアップする。
- **インクリメンタルです：**  
前回の増分バックアップまたは完全バックアップ以降に変更があったファイルをすべてバックアップする。
- **ディファレンシャルです：**  
前回のフルバックアップ以降に変更されたファイルをすべてバックアップします。
- **複数レベルのインクリメンタル：**  
前回のバックアップ以降に変更されたすべてのファイルを、同じレベルまたは前のレベルでバックアップする。
- **ユーザーです：**  
特定のユーザーのディレクトリにあるファイルのみをバックアップする。

いくつかの異なる種類のバックアップ方法を使用することができ、多くの場合、リストにあるように互いに協調して使用することができます。

すべてのバックアップを保護するシステムと同じ物理的な場所に置くことは避けなければならない。そうしないと、火災やその他の物理的なダメージで全損する可能性があります。以前は、磁気テープを安全な場所に物理的に輸送することが一般的でした。現在では、バックアップファイルをインターネット経由で別の物理的な場所に転送することが多くなっています。もちろん、これは暗号化やその他のセキュリティ対策が適切に施された安全な方法で行われる必要があります。

## バックアップ戦略

テープを使った便利な戦略です（説明の中で他のメディアで代用することも可能です）：

1. テープ1を金曜日のフルバックアップに使用する。
2. テープ2～5を月～木曜日の増分バックアップに使用します。
3. 第2金曜日のフルバックアップにテープ6を使用する。
4. テープ2～5を第2月曜日～木曜日の増分バックアップに使用する。
5. テープ6のフルバックアップが完了するまで、テープ1を上書きしないでください。
6. テープ6にフルバックアップした後、テープ1をディザスタリカバリ用に外部に移動する。
7. 次回のフルバックアップ（来週の金曜日）には、テープ1をテープ6に交換します。

バックアップの方法は、関連するリストアの方法がなければ意味がないことに注意する必要があります。戦略を選択する際には、両者の堅牢性、明確性、容易性を考慮する必要があります。

目安としては、最低でも2週間分のバックアップを用意しておくといでしょう。

最もシンプルなバックアップ方式は、一度すべてをフルバックアップし、その後に変更があったものについては増分バックアップを行うことです。フルバックアップには多くの時間がかかるが、増分バックアップからの復元はより困難で時間がかかる。したがって、時間と労力を最適化するために、両方を組み合わせて使用することができます。

## バックアップ関連ユーティリティの

- ヒメコマチ
- タール
- gzip、bzip2、xz
- ディーディー
- アールエスアイエヌシー
- ダンプ/リストア
- 麦稈

cpioとtarは、ファイルのアーカイブを作成・抽出します。

アーカイブは、多くの場合、gzip、bzip2、またはxzで圧縮されます。アーカイブファイルは、ディスク、磁気テープ、またはファイルを保持できるその他のデバイスに書き込むことができます。アーカイブは、あるファイルシステムやマシンから別のファイルへファイルを転送するのに非常に便利です。

ddは、メディア間の生データ転送によく使われる強力なユーティリティです。パーティション全体やディスク全体のコピーに使用することができます。

rsyncは、ネットワーク越しにディレクトリのサブツリーやファイルシステム全体を同期させたり、ローカルマシンの異なるファイルシステムの場所間で同期させることができる強力なユーティリティです。

dumpとrestoreは、バックアップのために特別に設計された古いユーティリティです。ファイルシステムから直接読み込むので、より効率的です。しかし、復元はファイルシステムから読み込んだのと同じファイルシステムタイプにのみ行う必要があります。より新しい代替手段があります。

mtは、バックアップやリストアを実行する前に、テープの問い合わせや位置決めを行うために使用します。

## 34.4 タール

# バックアップのためのtarの使用

- アーカイブの作成：-cを使うか、cだけでいい  
`$ tar cvf /dev/st0 /root`  
`tar -cvf /dev/st0 /root`
- マルチボリュームオプションで作成：-Mを使用  
`tar -cMf /dev/st0 /root`  
次のテープを入れるように促される
- compare オプションでファイルを検証する：-d または --compare を使用する。  
`$ tar --compare --verbose --file /dev/st0`  
`$ tar -dvf /dev/st0`
- 各オプションには、ショートフォーム（1文字に-を付けたもの）とロングフォーム（--を付けたもの）があることに注意してください。
- tarはデフォルトで再帰的である

tarアーカイブを作成する場合、引数として与えられた各ディレクトリに対して、すべてのファイルとサブディレクトリがアーカイブに含まれます。リストアする場合は、必要に応じてディレクトリを再構成します。

さらに--newerオプションで、増分バックアップも可能です。

Linuxで使用されているtarのバージョンは、1本のテープや使用するデバイスに収まりきらないバックアップも扱うことができます。fまたは-fileオプションでデバイスやファイルを指定することができます。

バックアップを作成した後、検証オプションを使用して、バックアップが完全で正しいことを確認することができます。デフォルトでは、tarはアーカイブに含まれるすべてのサブディレクトリを再帰的に含めます。

アーカイブを作成する際、tarは絶対パス名から先頭のスラッシュを削除する旨のメッセージを表示します。これにより、どこでもファイルを復元できるようになりますが、デフォルトの動作は変更することができます。

ほとんどのtarオプションは、ダッシュ1つの短い形式と、ダッシュ2つの長い形式で指定できます：-cは-createと完全に同じです。また、ダッシュをいちいち入力する必要がないように、オプションを組み合わせることができることに注意してください（短い表記を使用する場合）。

さらに、一点鎖線のtarオプションは、ダッシュの有無にかかわらず使用できます。例えば、tar cvf file.tar dir1は、tar -cvf file.tar dir1と同じ結果になります。

## ファイルの復元にtarを使用する

- アーカイブから取り出す： -x または --extract を使用します。

```
tar --extract -同じパーミッション --verbose --file /dev/st0
```

```
$ tar -xpvf /dev/st0
```

```
$ tar -xpvf /dev/st0
```

- 復元するファイルに特定の名前を付けることができます

```
$ tar -xvf /dev/st0 somefile
```

- tarバックアップの中身を一覧表示する

```
$ tar --list --file /dev/st0
```

```
$ tar -tf /dev/st0
```

xまたは--extractオプションは、デフォルトでアーカイブからすべてのファイルを抽出します。特定のファイルだけを指定して、抽出するファイルを絞り込むことができます。ディレクトリを指定した場合は、その中に含まれるすべてのファイルとサブディレクトリも抽出されます。

pまたは-same-permissionsオプションは、ファイルが元のパーミッションでリストアされることを保証します。tまたは-lオプションは、アーカイブ内のファイルを一覧表示しますが、抽出はしません。



## tarによる増量バックアップ

- newerまたは--after-dateオプションを使用します。
- 日付に基づく
- 特定の日付以降に変更されたすべてのファイルのバックアップを作成し、圧縮する：

```
$ tar --create --newer '2011-12-1' -vzf backup1.tgz /var/tmp
```

```
$ tar --create --after-date '2011-12-1' -vzf backup1.tgz /var/tmp
```

どちらの形式でも、`/var/tmp`にある2011年12月1日以降に変更されたすべてのファイルのバックアップアーカイブを作成します。

**tar**では、`-N`（または同等の`--newer`）オプション、または`--after-date`オプションを使用して、インクリメンタルバックアップを行うことができます。いずれのオプションも、日付または修飾された（参照）ファイル名のいずれかを指定する必要があります。

**tar**はファイルの日付しか見ないため、パーミッションやファイル名など、ファイルに対する他の変更点は考慮されません。これらの変更があるファイルを増分バックアップに含めるには、**find**を使用して、バックアップするファイルのリストを作成します。

**注意：**`--newer`のようなオプションが続く場合、`--vzf`のようなオプションではダッシュを使用しなければなりません。このようなオプション指定の混乱は、`ps`や**tar**のような古いUNIXユーティリティで、異なるファミリーのUNIXを含む複雑な歴史を持つ場合に発生することがあります。

## 34.5 圧縮: gzip、bzip2、xz、バックアップ

### アーカイブ圧縮方式

- ファイルを圧縮することで、ディスク容量やネットワーク通信時間を節約することができます。
- 圧縮効率を上げるためには、圧縮時間が長くなることを代償としています：
  - **gzip**です : Lempel-Ziv Coding (LZ77) を使用し、.gz ファイルを生成する。
  - **bzip2**: Burrows-Wheeler ブロックソートテキスト圧縮アルゴリズムとハフマンコーディングを使用し、.bz2ファイルを生成する。
  - **xz** : .xzファイルを生成します。また、レガシーな.lzmaフォーマットもサポートします。
- 最近の機種では、圧縮→送信→伸張のサイクルが、非圧縮のファイルを送信（コピー）するよりも速くなることが多い。
- <https://www.kernel.org> Linuxのダウンロードには現在**xz**フォーマットしか使用されていません。

カーネルズ

圧縮ユーティリティは、**tar**との組み合わせで非常に簡単に（そして頻繁に）使用することができます：

```
$ tar zcvf source.tar.gz ソース
$ tar jcvf source.tar.bz2 ソース
$ tar Jcvf source.tar.xz ソース
```

は、圧縮されたアーカイブを作成するためのコマンドです。最初のコマンドは、全く同じ効果であることに注意してください：

```
$ tar cvf source.tar source ; gzip -v source.tar
```

が、中間ファイルストレージがなく、アーカイブと圧縮がパイプラインで同時に行われるため、より効率的です。

減圧のため：

```
$ tar xzvf source.tar.gz
$ tar xjvf source.tar.bz2
$ tar xJvf source.tar.xz
```

またはもっとシンプルなものです：

```
$ tar xvf source.tar.gz
```

のように、最近のtarは圧縮の方法を感知して自動的に対処してくれます。

もちろん、画像や.pdfファイルなど、構成ファイルがすでに圧縮されているアーカイブにこれらの方法を使用する価値はない。



## 34.6 ディーディー

# ディーディー

- 生データのコピーに使用
- データの低レベルコピー
- 構文 : dd if=input-file of=output-file options  
\$ dd if=/dev/sda of=/dev/sdb
- ファイル作成に使用できる  
\$ dd if=/dev/zero of=file1 bs=1M count=10
- ハードディスクやパーティションを丸ごとバックアップすることが可能
- CDやDVDのコピーも可能

ddは、生データの低レベルなコピーと変換を主目的とするUNIXベースの一般的なプログラムである。指定されたバイト数またはブロックをコピーし、その場でバイト順序を変換したり、データがある形式から別の形式に変換したりするのに使用します。また、ハードディスクのブートセクタのバックアップなど、生のデバイスファイルの領域をコピーしたり、[/dev/zero](#)や[/dev/random](#)などの特殊なファイルから固定量のデータを読み取るのにも使用される。

ハードディスクを丸ごと別のハードディスクにバックアップする。

```
$ dd if=/dev/sda of=/dev/sdb
```

ハードディスクのイメージを作成します。

```
$ dd if=/dev/sda of=sdadisk.img
```

パーティションをバックアップする。

```
$ dd if=/dev/sda1 of=partition1.img
```

CD ROMをバックアップする。

```
$ dd if=/dev/cdrom of=tgsservice.iso bs=2048
```

## 34.7 アールエスアイエヌシー

# バックアップのためのrsyncの使用

- 使用することです :  
`rsync [オプション] sourcefile destinationfile`
- ローカルマシンとネットワークターゲットとの間 :  
`$ rsync file.tar someone@backup.mydomain:/usr/local`
- dry-runで実行する前にテストしてください :  
`rsync -r --dry-run /usr/local /BACKUP/usr`

**rsync** (remote synchronize) は、ネットワーク越しに（または同じマシン上の異なる場所間で）ファイルを転送するために使用されます。転送元と転送先はtarget:pathの形式で、targetは[user@]hostの形式で指定します。user@はの部分はオプションで、リモート・ユーザーがローカル・ユーザーと異なる場合に使用されます。このように、これらはすべて可能な**rsync**コマンドです :

```
$ rsync file.tar someone@backup.mydomain:/usr/local
rsync -r --dry-run /usr/local /BACKUP/usr
```

**rsync**では、正確な場所の指定には非常に注意が必要です（特に--deleteオプションを使用した場合）ので、最初に--dry-runオプションを使用し、予測される動作が正しいようならコマンドを繰り返すことを強くお勧めします。

**rsync** は非常に巧妙で、ローカルファイルとリモートファイルを小さな塊でチェックし、あるディレクトリを同様のディレクトリにコピーする際に、差分のみを最初のディレクトリと一緒にネットワーク上にコピーするという非常に効率的な機能を備えています。rオプションを使用すると、rsyncはディレクトリツリーを再帰的に歩き、ソースファイルとしてリストされたもの以下のすべてのファイルとディレクトリをコピーするようになります。このように、プロジェクト・ディレクトリをバックアップするのに非常に便利な方法は、次のようなものです :

```
rsync -r project-X archive-machine:archives/project-X
```

シンプルな（そして非常に効果的で高速な）バックアップ戦略は、単にディレクトリやパーティションをネットワーク上で複製することです。

**rsync**コマンドと、それを頻繁に行うこと。

## 34.8 バックアッププログラム\*\*について

# バックアッププログラム

- アマ ندا
- バキュラ
- クロネジラ

Linuxで利用可能なバックアッププログラムには、プロプライエタリなアプリケーションやストレージベンダーが提供するもの、オープンソースのアプリケーションなど、さまざまなものがあります。特によく知られているのは、以下のようなものです：

- **Amanda** (Advanced Maryland Automatic Network Disk Archiver)は、ネイティブユーティリティ（tarやdumpを含む）を使用しますが、はるかに堅牢で制御可能です。

Amandaは、一般的にEnterprise Linuxシステムで、通常のリポジトリを通じて利用可能です。完全な情報は <http://www.amanda.org> で見ることができます。

- **Bacula**は異種ネットワークでの自動バックアップのために設計されている。使い方がやや複雑なため、経験豊富な管理者のみに推奨されている（著者）。

Bacula は一般に、通常のリポジトリを通じて Enterprise Linux システムで利用可能である。完全な情報は、<https://www.bacula.org/7.0.x-manuals/en/main/Main Reference.html>で見ることができる。

- **Clonezilla**は非常に堅牢なディスククローニングプログラムで、ディスクのイメージを作成し、バックアップを復元したり、多くのマシンをインストールするために使用できるイメージを提供するゴーストに使用したりするために、それらを展開することができます。完全な情報は <https://clonezilla.org> で見るすることができます。

このプログラムには、2つのバージョンがあります：**Clonezilla Live**は、シングルマシンのバックアップとリカバリに適しています。

同時に多数のコンピュータにクローンを作成できるサーバー版「**Clonezilla SE**」。

Clonezillaの使い方はそれほど難しくなく、非常に柔軟で、Linuxだけでなく多くのオペレーティングシステム、ファイルシステムタイプ、ブートローダをサポートしています。

## 34.9 研究室

### 練習問題 34.1 : バックアップにtarを使う

1. backupというディレクトリを作成し、その中に/usr/include以下のすべてのファイル（最上位ディレクトリはinclude）を圧縮したtarアーカイブを配置します。圧縮方法は自由です（gzip、bzip2、xz）。
2. アーカイブのファイルを一覧表示します。
3. restoreというディレクトリを作成し、アーカイブを解凍・展開する。
4. アーカイブが作成された元のディレクトリと内容を比較します。

### 解答 34.1

1. 

```
$ mkdir /tmp/backup
$ cd /usr ; tar zcvf /tmp/backup/include.tar.gz include
$ cd /usr ; tar jcvf /tmp/backup/include.tar.bz2 インクルード
$ cd /usr ; tar Jcvf /tmp/backup/include.tar.xz includes
```

または

```
tar -C /usr -zcf include.tar.gz include.
$ tar -C /usr -jcf include.tar.bz2 include.
$ tar -C /usr -Jcf include.tar.xz include.
```

3つの方法間の圧縮の効き目に注目してください：

```
$ du -sh /usr/include
```

```
55M /usr/include
```

2. 

```
$ ls -lh include.tar.*.
```

```
c8:/tmp/backup>ls -lh
total 17M
-rw-rw-r-- 1 coop coop 5.3M Jul 18 08:17 include.tar.bz2
-rw-rw-r-- 1 coop coop 6.7M Jul 18 08:16 include.tar.gz
-rw-rw-r-- 1 coop coop 4.5M Jul 18 08:18 include.tar.xz
```

3. 

```
$ tar tvf include.tar.xz
```

```
qdrwxr-xr-x root/root 0 2014-10-29 07:04 インクルードー
-rw-r--r-- root/root 42780 2014-08-26 12:24 include/unistd.h
-rw-r--r-- root/root 957 2014-08-26 12:24 include/re_comp.h
-rw-r--r-- root/root 22096 2014-08-26 12:24 include/regex.h
-rw-r--r-- root/root 7154 2014-08-26 12:25 include/link.h

```

なお、解凍時にj,J,zオプションを指定する必要はありません。

4. 

```
$ cd ... ; mkdir restore ; cd restore
$ tar xvf ../backup/include.tar.bz2
```

```
インクルード/
インクルード
```



```
include/re_comp.h
include/regex.h
include/link

diff -qr include /usr/include
```

## 練習問題34.2: バックアップにrsyncを使用する

1. **rsync**を使って、再び**/usr/include**の完全なコピーをバックアップディレクトリに作成します：

```
$ rm -rf include
$ rsync -av /usr/include .
```

```
インクリメンタルファイ
ルリストの送信 include/
include/FlexLexer.h
include/_G_config.h
include/a.out.h
include/aio.h

```

2. 2回目にコマンドを実行し、何かが起こるかどうか確認してみましょう：

```
$ rsync -av /usr/include .
```

増分ファイルリスト送信

```
送信 127398 バイト 受信 188 バイト 255172.00 バイト/秒 総サイ
ズ 41239979 速度アップ 323.23
```

3. **rsync**で混乱することの1つは、正しいコマンドを期待していたかもしれないことです：

```
rsync -av /usr/include インクルード
```

増分ファイルリスト送信

```

```

しかし、これを実行すると、実際には**include/include**という新しいディレクトリが作成されることがわかります！

4. 余分なファイルを削除するには、**--delete**オプションを使用します：

```
$ rsync -av --delete /usr/include .
```

```
インクリメンタルファイ
ルリストの送信 include/
include/include/xen/privcmd.h 削除
include/include/xen/evtchn.h 削除

include/include/FlexLexer.h を削除
include/include/ を削除。
```

```
送信 127401 バイト 受信 191 バイト 85061.33 バイト/秒 総サイ
ズ 41239979 速度アップ 323.22
```

5. 別の簡単な練習として、バックアップコピーのサブディレクトリのツリーを削除し、`-dry-run`オプションの有無にかかわらず**rsync**を再度実行します：

```
$ rm -rf include/xen
$ rsync -av --delete --dry-run /usr/include .
```

```
インクリメンタルファイ
ルリストの送信 include/
include/xen/evtchn.h
include/xen/privcmd.h
include/xen/privcmd.h
```

```
送信 127412 バイト 受信 202 バイト 255228.00 バイト/秒 総サイ
ズ 41239979 速度アップ 323.16 (DRY RUN)
```

```
$ rsync -av --delete /usr/include .
```

6. **rsync**を使用するための優れたオプションのセットを備えたシンプルなスクリプトです：

SH

### rsyncを使ったスクリプト

```
#!/bin/sh
セット -x

rsync --progress -avrxH --delete $*.
```

は、ネットワーク上だけでなく、ローカルマシンでも動作します。**rsync**がファイルシステムの境界を越えないようにする重要な`-x`オプションに注意してください。

### エクストラクレジット

複数のコンピュータにアクセスできる場合は、ソースとデスティネーションを別のマシンに置いて、この手順を実行してみると、より楽しくなります。

## 第35章

# Linuxセキュリティモジュール



|      |                       |     |
|------|-----------------------|-----|
| 35.1 | Linuxセキュリティモジュール..... | 560 |
| 35.2 | セリナックス.....           | 562 |
| 35.3 | AppArmor.....         | 577 |
| 35.4 | 研究室.....              | 583 |

### 学習目標

このセッションが終わるころには、次のことができるようになっているはずです：

- Linux Security Moduleのフレームワークの仕組みと導入方法を理解する。
- 利用可能なさまざまなLSMの実装をリストアップします。
- SELinuxの主な機能を説明する。
- さまざまなモードと利用できるポリシーについて説明する。
- コンテキストの重要性とその取得・設定方法を把握する。
- 重要なSELinuxユーティリティプログラムの使い方を知ることができる。
- AppArmorの使い方に慣れる。

559



## 35.1 Linuxセキュリティモジュール

# Linuxのセキュリティモジュールと

- LSM ( Linux Security Module ) は、カーネルへのリクエストに対して強制的なアクセス制御を行うものである。
- LSMを使用する：
  - カーネルへの変更を最小限に抑えます。
  - カーネルにかかるオーバーヘッドを最小限に抑えることができます。
  - 異なる実装を柔軟に選択することができ、それぞれが自己完結した **LSM** ( Linux Security Module ) として提供されます。

現代のコンピュータシステムは安全でなければならないが、データの機密性、アカウントを持つユーザの数、外部ネットワークへの露出、法的要件、その他の要因によってニーズは異なる。優れたセキュリティ制御を可能にする責任は、アプリケーション設計者とLinuxカーネル開発者・保守者の双方にあります。もちろん、ユーザは適切な手順に従わなければなりません。うまく運用されているシステムでは、非特権ユーザがシステムをセキュリティ違反にさらす能力は非常に限られているはずです。このセクションでは、LinuxカーネルがLinux Security Modulesフレームワークの使用、特にSELinuxの導入により、どのようにセキュリティを強化しているかに注目する。

基本的な考え方は、システムコールをフックすることです。強化された能力を必要とする作業を行うために、アプリケーションがカーネル（システム）モードへの移行を要求するたびにコードを挿入します。このコードは、パーミッションが有効であること、悪意から保護されていることなどを確認します。このコードは、システムコールがカーネルによって実行される前や後に、セキュリティ関連の機能ステップを呼び出すことによって、これを実行します。

## LSMの主な選択項目

- SELinux: <https://selinuxproject.org/page/Main> ページ
- AppArmor : <https://apparmor.net>
- スマック : <http://schaufler-ca.com>
- TOMOYO : <https://tomoyo.osdn.jp>
- ヤマ : <https://kernel.org/doc/html/latest/admin-guide/LSM/Yama.html>



### 同時刻のLSMの使用

2019年からは、特定の指定された順序でLSMを組み合わせる（スタックする）ことができるようになりました。

- 見る <https://www.starlab.io/blog/a-brief-tour-of-linux-security-modules>  
メカニズムや選択肢について、素敵なレビューをしていただきました。

長い間、セキュリティ強化モデルとして実装されていたのはSELinuxだけでした。2001年にこのプロジェクトがカーネルに直接組み込まれるよう上流に持ち込まれたとき、セキュリティ強化のアプローチを1つだけにすることに異論がありました。

その結果、SELinuxの代替モジュールが開発されるとそれを利用できるLSM方式が採用され、2003年にLinuxカーネルに組み込まれました。

元々、LSMはLinuxカーネルの同じ部分を変更する可能性があるため、一度に1つしか使用できない。

しかし、現在では、多少の注意を払うことで積み重ねることが可能になっています。LSMの組み合わせを構成する際に、メジャーかマイナーかを考慮するようになりました。

ここでは、使用量の多い順に、主にSELinux、次にAppArmorに絞って説明します。

## 35.2 セリナックス

### SELinuxの概要

- NSAによって開発された
- 追加保護方法
- 使用します：
  - コンテキスト
  - ルール
  - ポリシー
- 明示的に許可されていないアクションは、デフォルトで拒否される
- 追加のオンラインリソース：

**SELinux ユーザーおよび管理者ガイド：**

[https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/7/html/selinux\\_ユーザーおよび管理者ガイド/index](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/selinux_ユーザーおよび管理者ガイド/index)

SELinuxは、もともと米国NSA（国家安全保障局）が開発したもので、RHELに不可欠なものです。

は、非常に長い間、多くのユーザーに愛用されてきました。

運用上、SELinuxは、どのプロセスがシステム上のどのファイル、ディレクトリ、ポート、およびその他のアイテムにアクセスできるかを決定するために使用されるセキュリティルールのセットである。

この3つの概念量に対応しています：

1. **コンテキスト（Contexts）**：ファイル、プロセス、ポートに対するラベルです。コンテキストの例としては、SELinuxのユーザー、ロール、タイプなどがあります。
2. **ルールコンテキスト**、プロセス、ファイル、ポート、ユーザーなどの観点からアクセス制御を記述する。
3. **ポリシーのこと**：SELinuxがどのようなシステム全体のアクセス制御を決定すべきかを記述した**ルールのセット**です。

SELinuxのコンテキストは、ユーザー、プロセス、ファイル、ポートが互いにどのように相互作用するかを定義するためにルールによって使用されるラベルです。デフォルトのポリシーはあらゆるアクセスを拒否することであるため、ルールはシステム上で許可されたアクションを記述するために使用されます。



# SELinux Enforcement Modes I

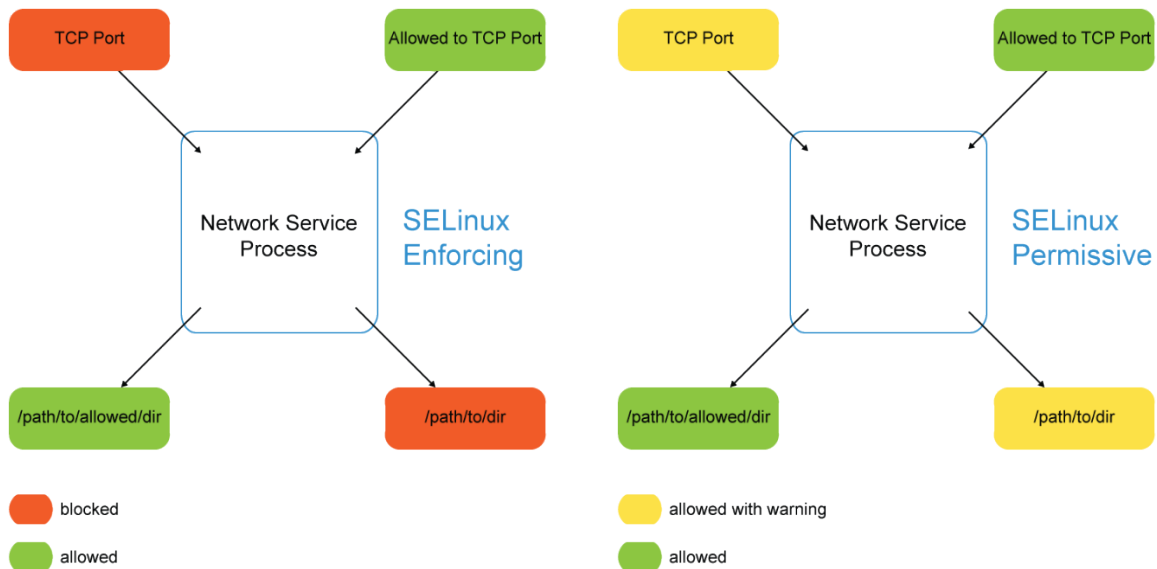


図35.1 : SELinux エンフォースメントモード

図35.2 SELinuxのパーミッシブモード

これらのモードは、ファイル (通常は `/etc/selinux/config`) で選択 (および説明) され、その場所は配布によって異なります (`/etc/sysconfig/selinux` にあるか、そこからリンクされていることが多いです)。このファイルは、よく自己文書化されています。

## SELinux Enforcement Modes II

- **SELinuxの3つのモード**
  - エンフォースメント
  - パーミッシブ
  - 使用不可
- `etc/sysconfig/selinux` (CentOS、openSUSE) で設定する、または `/etc/selinux/config` (Ubuntu)
- 現在のモードを確認または設定するには、`getenforce` および `setenforce` を使用します。

- **エンフォースする**：すべてのSELinuxコードが動作し、ポリシーに従ってアクセスが拒否されます。すべての違反は監査され、ログに記録されます。
  - **Permissive (許可)**：SELinuxコードを有効にしますが、強制モードでは拒否される操作についてのみ監査と警告を行います。
  - **無効にする**：SELinuxカーネルとアプリケーションコードを完全に無効化し、システムを保護しない状態にします
- 。 `sestatus` ユーティリティは、現在のモードとポリシーを表示することができます。

## ゲットエンフォースとセットエンフ

- 現在のモードを調べる、または設定する：

- 勢いづく

```
$ getenforce
```

```
エンフォースメント
```

- セットインフォース

```
sudo setenforce Permissive
```

```
$ getenforce
```

```
パーミッシブ
```

**setenforce** を使用すると、システムの動作中にその場で強制モードと許可モードを切り替えることができます。ただし、無効モードのイン、アウトへの変更は、この方法ではできません。

setenforceでは、**Permissive**モードと**Enforcing**モードを切り替えることができますが、**SELinux**を無効にすることはできません。完全に無効にすることができます。**SELinux**を無効にするには、少なくとも2種類の方法があります：

- **設定ファイル**：**SELinux**の設定ファイル（通常は`/etc/selinux/config`）を編集し、`SELINUX=disabled`に設定します。これはデフォルトの方法であり、**SELinux**を永久に無効にするために使用する必要があります。
- **Kernelのパラメータ**です：リブート時のKernelパラメータリストに`selinux=0`を追加。

しかし、**SELinux**が再有効化されるシステムで**SELinux**を無効にすることは推奨されないことに注意することが重要である。**SELinux**を無効にする代わりに、**Permissive**モードを使用することが望ましいです。

## SELinux ポリシー

- をターゲットにした :

**SELinux**が対象とするプロセスをより厳しく制限するデフォルトのポリシーです。ユーザープロセスやinitプロセスは対象外ですが、ネットワークサービスプロセスは対象となります。**SELinux**はすべてのプロセスに対してメモリ制限を実施し、バッファオーバーフロー攻撃に対する脆弱性を低減します。

- を最低限にしています :

選択されたプロセスのみが保護される、ターゲットポリシーの修正。

- **MLS**です :

マルチレベルセキュリティポリシーは、より制限的で、すべてのプロセスは、特定のポリシーを持つ細かいセキュリティドメインに配置されます。

モードを設定する同じ設定ファイル（通常は`/etc/sysconfig/selinux`）で、**SELinux**ポリシーも設定します。複数のポリシーを設定することができますが、一度にアクティブにできるのは1つだけです。

ポリシーを変更するには、システムの再起動と、ファイルシステムのコンテンツの再ラベル化に時間がかかる場合があります。各ポリシーには、`/etc/selinux/[SELINUXTYPE]`の下にインストールする必要があるファイルがあります。

## コンテキスト・ユーティリティ

- コンテキストとコンテキスト・ユーティリティ
  - ユーザー
  - 役割
  - タイプ
  - レベル
- コンテキストを調べる、または変更する：
  - コンテキストを見るには `-Z` オプションを使用します。  
`$ ls -Z`  
`$ ps auZ`
  - コンテキストを変更するには、**chcon**を使用します。  
`$ chcon -t etc_t somefile`  
`$ chcon --reference somefile someotherfile`

前述のように、コンテキストとは、ファイル、ディレクトリ、ポート、プロセスに適用されるラベルのことである。このラベルを利用してアクセスルールを記述します。**SELinux**のコンテキストは、**User**、**Role**、**Type**、**Level**の4つです。

ここでは、最も一般的に利用されるコンテキストである**type**に焦点を当てます。ラベルの命名規則では、typeコンテキストのラベルはkernel\_tのように\_tで終わるように決められています。

\$ ls -Z

```
-rw-rw-r--. dog dog unconfined_u:object_r:user_home_t:s0 somefile.
```

\$ chcon -t etc\_t somefile

\$ ls -Z

```
-rw-rw-r--. 犬 犬 不自由_u:物_r:等_t:s0 サムファイル
```

\$ ls -Z

```
-rw-rw-r--. 犬 犬 不自由_u:物_r:等_t:s0 サムファイル
-rw-rw-r--. 犬 犬 unconfined_u:object_r:user_home_t:s0 somefile1
```

\$ chcon --reference somefile somefile1

\$ ls -Z

```
-rw-rw-r--. 犬 犬 不自由_u:物_r:等_t:s0 サムファイル
-rw-rw-r--. 犬 犬 unconfined_u:object_r:etc_t:s0 somefile1
```

## SELinuxと標準コマンド

- 多くの基本的なユーティリティ（ls、ps、cpなど）がSELinuxで動作するように拡張されています。
- のように、通常はZオプションを追加します：  
\$ ls -Z ...  
\$ ps axZ
- SELinuxが無効の場合、拡張オプションは何もしません。

lsやpsなどの多くの標準的なコマンドラインコマンドがSELinuxをサポートするように拡張され、詳細を説明する対応するセクションがそれらのマニュアルページに追加されました。のように、標準的なコマンドラインツールにパラメータZが渡されることがよくあります：

\$ ps axZ

```
ラベル pid tty stat time コマンド
system_u:system_r:init_t:s0 1 ?Ss 0:04 /usr/lib/systemd/systemd --switched-root ... system_u:system_r:kernel_t:s0
2 ?S 0:00 [kthreadd] です。
...
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 2305 ?D 0:00 sshd: jimih@pts/0
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 2306 pts/0 Ss 0:00 -bash
...
system_u:system_r:httpd_t:s0 7490 ?Ss 0:00 /usr/sbin/httpd -DFOREGROUND
system_u:system_r:httpd_t:s0 7491 ?S 0:00 /usr/sbin/httpd -DFOREGROUND
...
```

\$ ls -Z /home/ /tmp/...

```
/home/ :
drwx-----. jimih jimih unconfined_u:object_r:user_home_dir_t:s0 jimih
/tmp/ :
-root root system_u:object_r:initrc_tmp_t:s0 ks-script-c4ENhg drwx-----.
root root system_u:object_r:tmp_t:s0 systemd-private-0ofSvO
-rw-----. root root system_u:object_r:initrc_tmp_t:s0 dnf.log
```

SELinuxをサポートするために拡張された他のツールには、cp、mv、mkdirがあります。

SELinuxを無効にしている場合、これらのユーティリティの関連フィールドには有用な情報が表示されないことに注意してください。

## コンテキストの継承

- 新しいファイルは親ディレクトリからコンテキストを継承する
- ファイル移動時、ソースディレクトリのコンテキストを保持
- コンテキストが変更されないとは問題が発生する可能性がある



新しく作成されたファイルは親ディレクトリのコンテキストを引き継ぎますが、ファイルの移動・コピー時には、移動元のディレクトリのコンテキストが保存されることがあり、これが問題になることがあります。

前の例に引き続き、ファイルを/tmpから/home/jimihに移動しても、tmpfileのコンテキストは変更されていないことがわかります：

```
$ cd /tmp/
$ touch tmpfile
$ ls -Z tmpfile
```

```
jimih jimih unconfined_u:object_r:user_tmp_t:s0 tmpfile -rw-rw-r--.
```

```
$ cd
$ touch homefile
$ ls -Z homefile
```

```
jimih jimih unconfined_u:object_r:user_home_t:s0 homefile -rw-rw-r--.
```

```
$ mv /tmp/tmpfile .
$ ls -Z
```

```
jimih jimih unconfined_u:object_r:user_home_t:s0 homefile -rw-rw-r--.
```

```
jimih jimih unconfined_u:object_r:user_tmp_t:s0 tmpfile -rw-rw-r--.
```

ファイルの移動がSELinuxの問題を引き起こす典型的な例は、httpdサーバーのDocumentRootディレクトリにファイルを移動することです。SELinuxが有効なシステムでは、Webサーバーは正しいコンテキストラベルを持つファイルにしかアクセスできない。tmpにファイルを作成し、DocumentRootディレクトリに移動すると、ファイルのSELinuxコンテキストが調整されるまで、httpdサーバーからそのファイルにアクセスできなくなります。

# リストコン

- 親ディレクトリと一致するようにファイルコンテキストをリセットする :  
`$ restorecon -Rv /home/jimih`

**restorecon** は、親ディレクトリの設定に基づき、ファイルコンテキストをリセットします。

次の例では、**restorecon**はホームディレクトリのすべてのファイルに対して再帰的にデフォルトラベルをリセットしています :

`$ ls -Z`

```
jimih jimih unconfined_u:object_r:user_home_t:s0 homedir -rw-rw-r--.
jimih jimih unconfined_u:object_r:user_tmp_t:s0 tmpfile -rw-rw-r--.
```

`$ restorecon -Rv /home/jimih`

```
restorecon reset /home/jimih/tmpfile context ✗。
アンコンファインド_u:オブジェクト_r:ユーザー_tmp_t:s0->アンコンファインド_u:オブジェクト_r:ユ
ーザー_home_t:s0
```

`$ ls -Z`

```
jimih jimih unconfined_u:object_r:user_home_t:s0 homedir -rw-rw-r--.
-jimih jimih unconfined_u:object_r:user_home_t:s0 tmpfile
```

**tmpfile**のコンテキストは、ホームディレクトリに作成されるファイルのデフォルトコンテキストにリセットされたことに注意してください。型はuser\_tmp\_tからuser\_home\_tに変更されました。

## セマナージュ

- 新しいディレクトリのデフォルトコンテキストを設定する
- デフォルトの設定のみを変更し、既存の設定は変更しない
- 既存のコンテキストを変更するには、その後に **restorecon** を呼び出す必要があります。

**semanage fcontext** (**policycoreutils-python** パッケージで提供) は、ファイルやディレクトリのデフォルトコンテキストを変更および表示できます。**semanage fcontext** はデフォルトの設定を変更するだけで、既存のオブジェクトに適用しないことに注意してください。そのため、その後で **restorecon** を呼び出す必要があります。例えば、以下のようになります：

```
[root@rhel7 ~]# mkdir /virtualHosts
[root@rhel7 ~]# ls -Z
```

```

drwxr-xr-x. root root unconfined_u:object_r:default_t:s0 virtualHosts
```

```
[root@rhel7 ~]# semanage fcontext -a -t httpd_sys_content_t /virtualHosts
[root@rhel7 ~]# ls -Z
```

```

drwxr-xr-x. root root unconfined_u:object_r:default_t:s0 virtualHosts
```

```
[root@rhel7 ~]# restorecon -RFv /virtualHosts
```

```
restorecon reset /virtualHosts context ①。
アンファインド_u:オブジェクト_r:デフォルト_t:s0->システム_u:オブジェクト
_r:httpd_sys_content_t:s0
```

```
[root@rhel7 ~]# ls -Z
```

```
drwxr-xr-x. root root system_u:object_r:httpd_sys_content_t:s0 virtualHosts
```

このため、**default\_t**から**httpd\_sys\_content\_t**へのコンテキスト変更は、**restorecon**の呼び出しの後にのみ適用されます。

## SELinuxのブール値の使用

- SELinuxポリシーの動作を変更する
- 有効・無効の切り替えが可能
  - **getsebool**: ブーリアンを見る
  - **setsebool**: ブーリアン値を設定する
  - **semanage boolean -l**: 永続的な boolean 設定を表示します。

```
$ setsebool allow_ftpd_anon_write on.
$ getsebool allow_ftpd_anon_write
allow_ftpd_anon_write -> on
$ semanage boolean -l | grep allow_ftpd_anon_write
$ allow_ftpd_anon_write -> off
```

しつこくないノート。

```
$ setsebool -P allow_ftpd_anon_write on
$ semanage boolean -l | grep
allow_ftpd_anon_write allow_ftpd_anon_write ->
になっています。
```

今、しつこいくらいに。

```
student@CentOS7:tmp
[student@CentOS7 tmp]$ semanage boolean -l
SELinux boolean State Default Description
ftp_home_dir (on , on) Allow ftp to home dir
smartmon_3ware (off , off) Allow smartmon to 3ware
mpd_enable_homedirs (off , off) Allow mpd to enable homedirs
xdm_sysadm_login (off , off) Allow xdm to sysadm login
xen_use_nfs (off , off) Allow xen to use nfs
mozilla_read_content (off , off) Allow mozilla to read content
ssh_chroot_rw_homedirs (off , off) Allow ssh to chroot rw homedirs
mount_anyfile (on , on) Allow mount to anyfile
cron_userdomain_transition (on , on) Allow cron to userdomain transition
icecast_use_any_tcp_ports (off , off) Allow icecast to use any tcp ports
openvpn_can_network_connect (on , on) Allow openvpn to can network connect
zoneminder_anon_write (off , off) Allow zoneminder to anon write
minidlna_read_generic_user_content (off , off) Allow minidlna to read generic user content
spassassin_can_network (off , off) Allow spassassin to can network
gluster_anon_write (off , off) Allow gluster to anon write
deny_ptrace (off , off) Allow deny to ptrace
selinuxuser_execadm (on , on) Allow selinuxuser to execadm
httpd_can_network_relay (off , off) Allow httpd to can network relay
openvpn_enable_homedirs (on , on) Allow openvpn to enable homedirs
glance_use_execadm (off , off) Allow glance to use execadm
telepathy_tcp_connect_generic_network_ports (on , on) Allow telepathy to tcp connect generic network ports
httpd_can_connect_aythiv (off , off) Allow httpd to can connect aythiv
....
[student@CentOS7 tmp]$
```

図35.3 : semanage

## SELinuxのアクセス監視

- SELinuxのエラーを追跡するためにsetroubleshoot-serverを使用します。
  - インストール後、auditdサービスを再起動します。
- 生のメッセージを /var/log/audit/audit.log に保存します。
- メッセージを /var/log/messages に移動します。
- 詳細なメッセージを見るには、sealertを使用します。

SELinuxには、実行時に問題を収集し、その問題を記録し、同じ問題の再発を防ぐための解決策を提案するツールのセットが付属しています。これらのユーティリティは、setroubleshoot-serverパッケージで提供されています。以下はその使用例です：

```
[root@rhel7 ~]# echo 'ファイルは/rootに作成'> rootfile
[root@rhel7 ~]# mv rootfile /var/www/html/
[root@rhel7 ~]# wget -O - localhost/rootfile
-2014-11-21 13:42:04-- http://localhost/rootfile
localhost (localhost) の解決... ::1, 127.0.0.1 localhost
(localhost)::1:80 に接続中... 接続されました。HTTPリ
クエストを送信しました、応答待ちです...403 Forbidden
2014-11-21 13:42:04 ERROR 403 : Forbidden です。

[root@rhel7 ~]# tail /var/log/messages
11/21 13:42:04 rhel7 setroubleshoot : プラグインエクセプション restorecon
11/21 13:42:04 rhel7 setroubleshoot : SELinuxは、 /usr/sbin/httpdがgetattrアクセスでファイル.NETにア
クセスすることを妨げています。SELinuxの完全なメッセージについては、sealert -l d51d34f9-91d5-4219-ad1e-
5531e61a2dc3 を実行してください。
11/21 13:42:04 rhel7 python : SELinuxは、 /usr/sbin/httpdがgetattrアクセスするのを防いでいます。

.....
実行することで、今のところこのアクセスを許可してください：
grep httpd /var/log/audit/audit.log | audit2allow -M mypol
semodule -i mypol.pp

追加情報です：
ソースコンテキスト system_u:system_r:httpd_t:s0
対象コンテキスト
unconfined_u:object_r:admin_home_t:s0 対象オブジェ
クト[ファイル]。
ソース httpd
ソースパス /usr/sbin/httpd
```

## 35.3 AppArmor

# AppArmor

- **MAC ( Mandatory Access Control )** を提供する。
- 管理者は、プログラムにセキュリティプロファイルを関連付け、その機能を制限することができます。
- **SELinux**よりも使いやすいとされています（一部ではありますが、すべてではありません）。
- ファイルシステムニュートラル（セキュリティラベルが不要）とされています。

AppArmorは、**SELinux**に代わる**LSM**である。2006年以降、**Linux**カーネルにサポートが組み込まれています。**SUSE**、**Ubuntu**、その他のディストリビューションで採用されています。

**AppArmor**は、強制アクセス制御（**MAC**）を提供することで、従来の**UNIX**の裁量アクセス制御（**DAC**）モデルを補完しています。

AppArmorには、手動でプロファイルを指定する以外に、学習モードがあり、プロファイルの違反がログに記録されますが、防止はされません。このログは、プログラムの典型的な動作に基づき、プロファイルに変換することができます。

## ステータスを確認する

- **AppArmor**は通常有効で、デフォルトでロードされます。
- **Linux**カーネルは、それがコンパイルされ、オンになっている必要があります。
- スタートとストップなど :  
`sudo systemctl [start|stop|restart|status] apparmor`
- ブート時にロードされる、またはロードされない原因 :  
`sudo systemctl [enable|disable] apparmor`
- 現在の状況をより詳しく説明します :  
`sudo apparmor_status`
- プロファイルとプロセスは、強制モードと苦情モードのいずれかになります。
  - **SELinux**の**enforcing**モードと**permissive**モードに直接類似しています。



AppArmorが付属しているディストリビューションは、これを有効にし、デフォルトでロードする傾向があります。Linuxカーネルもこれをオンにする必要があり、ほとんどの場合、一度に1つのLSMしか実行できないことに注意してください。

AppArmorカーネルモジュールが利用可能であると仮定すると、systemdを搭載したシステム上では、以下のようにすることができます：

```
sudo systemctl [start|stop|restart|status] apparmor
```

を変更したり、現在の動作状態を問い合わせたり、する：

```
sudo systemctl [enable|disable] apparmor
```

で、起動時に読み込ませるか、読み込ませないかを設定します。現在の状態を確認するために

```
sudo apparmor_status
```

```
apparmorモジュールがロードされます。
25個のプロファイルが読み込まれます。
25のプロファイルがエンフォースメントモードになっています。
/sbin/dhclient

```

プロファイルとプロセスは、SELinuxのenforcingとpermissiveに直接類似した、enforceまたはcomplainモードになっています。プロセスのリストでは、PID が与えられていることに注意してください：

```
ps aux | grep libvirtd
```

```
根 787 0.0 0.9 527200 35936 ? Ssl 10:54 0:00 /usr/sbin/libvirtd
スチューデント 3346 0.0 0.0 13696 2204 pts/16 S+ 11:42 0:00 grep --color=auto libvirtd
```

## モードとプロファイル

- モードです :
  - **Enforce Mode** 制限されたアクションを防止し、報告します。**aa-enforce**で設定する
  - 苦情 ポリシーは強制されないが、ポリシー違反は報告される。**aa-complain**とセットで
- **etc/apparmor.d**で設定されたプロファイルは、ユーティリティとソフトウェアパッケージの詳細を示します。

プロファイルは、`/usr/bin/evince` のようなシステム上のパス名を持つ実行プログラムの使用方法を制限します。プロセスは、2つのモードのどちらかで実行できます：

- ・ エンフォースモード

アプリケーションは、制限された方法で動作することを阻止される。違反の試みは、システムのログファイルに報告されます。これはデフォルトのモードです。プロファイルは、**aa-enforce**でこのモードに設定することができます。

- ・ クレーム

ポリシーは強制されませんが、ポリシー違反の試みは報告されます。これは**学習モード**とも呼ばれます。プロファイルをこのモードに設定するには、**aa-complain**を使用します。

Linuxディストリビューションには、あらかじめプロファイルがパッケージングされています。

**apparmor-profiles**のような**AppArmor**パッケージ。これらのプロファイルは、`/etc/apparmor.d`に保存されます。

新しいソフトウェアをインストールする際、パッケージ内の実行ファイルに特化した新しいプロファイルを作成することができます。

システムにインストールされる**AppArmor**プロファイルは、ソフトウェア・パッケージの選択によって異なります。ある特定の**Ubuntu**システムで：

```
student@ubuntu:/etc/apparmor.d$ ls
```

|                        |                                     |                                    |
|------------------------|-------------------------------------|------------------------------------|
| 抽象化                    | <code>usr.lib.dovecot.anvil</code>  | <code>usr.lib.telepathy</code>     |
| <code>apache2.d</code> | <code>usr.lib.dovecot.auth</code>   | <code>usr.sbin.avahi-daemon</code> |
| <code>bin.ping</code>  | <code>usr.lib.dovecot.config</code> | <code>usr.sbin.cups-brows</code>   |
| ----                   |                                     |                                    |

これらのファイルに記述できる内容については、`man apparmor.d`を実行することで完全なドキュメントを得ることができます。

## AppArmor ユーティリティ

表35.1 : AppArmorユーティリ

| プログラム      | 使用する                                                                                                            |
|------------|-----------------------------------------------------------------------------------------------------------------|
| アブローズステータス | すべてのプロファイルとプロファイルを持つプロセスのステータスを表示する                                                                             |
| アポロン通知     | <b>AppArmor</b> のログメッセージのサマリーを表示します。                                                                            |
| 囁らす        | 指定したプロファイルをコンプラモードに設定する。                                                                                        |
| 強要         | 指定したプロファイルをエンフォースメントモードに設定する。                                                                                   |
| 無効にする      | 現在のカーネルから指定されたプロファイルをアンロードし、システム上でロードされないようにする。<br>を起動します。                                                      |
| ログプロフ      | ログファイルをスキャンし、 <b>AppArmor</b> のイベントが記録されている場合、既存のイベントでカバーされていないことがあります。<br>プロファイルを考慮する方法を提案し、承認されれば修正と再装填を行います。 |
| イーjeeプロフ   | プログラムの基本的な <b>AppArmor</b> プロファイルのセットアップを支援します。                                                                 |

AppArmorには、監視と制御のための管理ユーティリティがかなり充実しています。例えば、openSUSEシステムで：

```
rpm -qil apparmor-utils | grep bin
```

```
/usr/bin/aa-easyprof
/usr/sbin/aa-audit
/usr/sbin/aa-autodep
/usr/sbin/aa-cleanprof です。
/usr/sbin/aa-complain
/usr/sbin/aa-decode
/usr/sbin/aa-disable
/usr/sbin/aa-enforce
/usr/sbin/aa-exec

/usr/sbin/complain
/usr/sbin/decode
/usr/sbin/disable
/usr/sbin/enforce

```

これらのユーティリティの多くは、短い名前でも長い名前でも呼び出すことができることに注意してください：

```
linux-llgn:/etc/apparmor.d # ls -l /usr/sbin/*complain
```

```
-rwxr-xr-x 1 root root 1442 Oct 25 07:37 /usr/sbin/aa-complain*.
lrwxrwx 1 root ルート 11 11月11日 13:02 /usr/sbin/complain -> aa-
complain* linux-llgn:/etc/apparmor.d #.
```

## 35.4 研究室

### 練習問題 35.1 : SELinuxのコンテキスト



#### ご注意ください

この演習は、SELinuxがインストールされているシステム（RHELなど）でのみ行うことができます。UbuntuのようなDebianベースのディストリビューションにもインストールは可能ですが、最も簡単な作業ではないため、あまり行われ

1. `getenforce`と`sestatus`を実行し、SELinuxが有効で、かつ強制モードであることを確認する。有効でない場合は、`/etc/selinux/config`を編集し、再起動後、再度確認してください。
2. Apache Webサーバーを提供する`httpd`パッケージ（まだ存在しない場合）をインストールし、動作することを確認します：

```
$ sudo dnf install httpd
sudo systemctl start httpd
エリンクス http://localhost
```

（この手順以降では、ブラウザとしてlynxやelinksなどを使用したり、Firefoxやchromeなどのグラフィカルブラウザを使用することも可能です）

3. スーパーユーザーとして、`/var/www/html`に小さなファイルを作成します：

```
$ sudo sh -c "echo file1 > /var/www/html/file1.html"
```

4. 見えるかどうかを検証する：

```
$ elinks -dump http://localhost/file1.html
```

フ

次に、rootのホームディレクトリに別の小さなファイルを作成し、それを`/var/www/html`に移動します。（コピーしないで、移動してください！）それから、それを見てみてください：

```
$ sudo cd /root
$ sudo sh -c "エコーファイル2 > ファイル2.html"
$ sudo mv file2.html /var/www/html
$ elinks -dump http://localhost/file2.html
```

#### 禁じ手

このサーバーの `/file2.html` にアクセスする権限がありません。

5. セキュリティのコンテキストを調べる：

```
$ cd /var/www/html
$ ls -Z file*.html
```

```
-rw-r--r--. root unfinde_u:object_r:httpd_sys_content_t:s0 file1.html
-rw-r--r--. root unfinde_u:object_r:admin_home_t:s0 file2.html
```

6. 問題のあるコンテキストを変更し、再度表示します：

```
$ sudo chcon -t httpd_sys_content_t file2.html
```

---

```
$ elinks http://localhost/file2.html
```

フ

## 練習問題35.2: apparmorのセキュリティについて調べる



### ご注意ください

この演習は、AppArmorがインストールされているシステム（Ubuntuなど）でのみ実行できます。

以下はUbuntuでテストされましたが、openSUSEなど他のAppArmor対応システムでも動作するはずで、aptコマンドはzypperに置き換える必要があります。

Ubuntuでは、/bin/pingユーティリティはSUIDを有効にして実行されます。この演習では、pingをping-xにコピーして、プログラムが機能するように機能を調整します。

その後、AppArmorプロファイルを構築し、インストールし、何も変化がないことを確認します。AppArmorプロファイルを変更し、機能を追加することで、プログラムがより機能的になります。

1. 必要なパッケージがすべてインストールされているこ

とを確認します：

student@ubuntu:~\$ sudo apt install apparmor\* です。

2. pingのコピー（ping-xと呼ぶ）を作成し、初期の特別な権限や機能を持たないことを確認する。さらに、一般ユーザーであるstudentが実行しても動作しない：

```
student@ubuntu:~$ sudo cp /bin/ping /bin/ping-x
student@ubuntu:~$ sudo ls -l /bin/ping-x
```

```
-rwxr-xr-x 1 root root 64424 Oct 17 10:12 /bin/ping-x
```

```
student@ubuntu:~$ sudo getcap /bin/ping-x
student@ubuntu:~$ です。
```

```
student@ubuntu:~$ ping-x -c3 -4 127.0.0.1
ping: socket : 操作が許可されていない
```

3. 能力を設定し、ping-xを再試行する：

```
student@ubuntu:~$ sudo setcap cap_net_raw+ep /bin/ping-x
```

```
student@ubuntu:~$ ping-x -c3 -4 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) バイトのデータです。
127.0.0.1 から 64 バイト: icmp_seq=1 ttl=64 time=0.092 ms
127.0.0.1 から 64 バイト: icmp_seq=2 ttl=64 time=0.093 ms
127.0.0.1 から 64 バイト: icmp_seq=3 ttl=64 time=0.086 ms
```

```
--- 127.0.0.1 pingの統計 ---
```

```
3パケット送信、3パケット受信、パケットロス0%、時間2034ms rtt
min/avg/max/mdev = 0.086/0.090/0.093/0.008 ms
```

修正したping-xプログラムは正常に機能するようになりました。

4. ping-x用の既存のAppArmorプロファイルがなく、ping用のプロファイルがあることを確認します。現在のpingプログラムのステータスを確認します：

```
student@ubuntu:~$ sudo aa-status
```

aa-statusの出力は長いので、興味深い行をgrepで探すことができます：

```
student@ubuntu:~$ sudo aa-status | grep -e "[[:alnum:]]"-イーピーイング
```

```
apparmorモジュールがロードされます。
87個のプロファイルが読み込まれています。
51のプロファイルがエンフォースメントモードになっている
ping
36のプロファイルがコンプラモードになっています。
17のプロセスでプロファイルを定義しています。
6プロセスがエンフォースメントモードになっている
11プロセスがコンプラモード。
0プロセスはアンファインドだが、プロファイルが定義されている。
```

pingにはプロファイルがロードされており、施行が有効になっていることがわかります。

- 次に、**ping-x**用の新しいプロファイルを構築します。このステップでは、2つのターミナルウィンドウが必要です。最初のウィンドウ (**window1**) は、**aa-genprof**コマンドを実行します。これは、**/var/log/syslog**で**AppArmor**エラーをスキャンして**AppArmor**プロファイルを生成するものです。2番目のウィンドウ (**window2**) で**ping-x**を実行します (詳しくは**aa-genprof**のマニュアルページを参照) :

```
student@ubuntu:~$ sudo aa-genprof /bin/ping-x
/bin/ping-xのプロファイルを更新して書き込みます。
bin/ping-xをコンプラモードに設定する。
```

開始する前に、閉じ込めたいアプリケーションのプロファイルがすでに存在するかどうかを確認するとよいでしょう。詳しくは、以下のWikiページを参照してください :

<http://wiki.apparmor.net/index.php/Profiles>

今すぐ別ウィンドウでプロファイル対象のアプリケーションを起動し、その機能を行使してください。

完了したら、AppArmorイベントのシステム・ログをスキャンするために、以下の「スキャン」オプションを選択します。

AppArmorの各イベントについて、アクセスを許可するか拒否するかを選択する機会が提供されます。

プロファイリングしています : /bin/ping-x

[AppArmorイベントのシステムログを取得する] / (F)inish

**window2にて :**

```
student@ubuntu:~$ ping -c3 -4 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) バイトのデータです。
127.0.0.1 から 64 バイト: icmp_seq=1 ttl=64 time=0.099 ms
127.0.0.1 から 64 バイト: icmp_seq=2 ttl=64 time=0.120 ms
127.0.0.1 から 64 バイト: icmp_seq=3 ttl=64 time=0.114 ms
```

--- 127.0.0.1 pingの統計 ---

```
3パケット送信、3パケット受信、パケットロス0%、時間2041ms rtt
min/avg/max/mdev = 0.099/0.111/0.120/0.008 ms
```

**window1にて :**

ping-xコマンドが完了したので、次は**aa-genprof**にプロファイルに追加する必要な情報をスキャンするよう指示する必要があります。プロファイルに必要な情報をすべて収集するためには、数回の**スキャン**が必要な場合があります。スキャンする場合は**S**を入力します :



var/log/syslogからログエントリを読み込む。  
etc/apparmor.dのAppArmorプロファイルの更新。  
。コンプレインモードの変更：

プロファイルです：  
/bin/ping-x  
Capability: net\_raw  
Severity: 8

[1 - capability net\_raw、]です。  
(A)llow / [(D)eny] / (l)gnore / Audi(t) / Abo(r)t / (F)inish

ケイパビリティを許可する場合はAを入力します：

プロファイルにケイパビリティnet\_rawを追加する。

プロファイルです：  
/bin/ping-x  
ネットワークファミリー  
: inet  
ソケットタイプです： 生

[1 - ネットワーク inet raw、]。  
(A)llow / [(D)eny] / (l)gnore / Audi(t) / Abo(r)t / (F)inish

ネットワークファミリーを許可する場合は、Aを入力します：

プロファイルにネットワーク inet raw、を追加する。

プロファイルです：  
/bin/ping-x  
ネットワークファミリー  
: inet  
ソケットタイプです： ドグラム

[1 - #include <abstractions/nameservice>] です。  
2 - network inet dgram、  
(A)llow / [(D)eny] / (l)gnore / Audi(t) / Abo(r)t / (F)inish

Aを入力すると、ソケットタイプのデータグラムがプロファイルに追加されます：

プロファイルに #include <abstractions/nameservice> を追加。

= 変更されたローカルプロファイル =。

以下のローカルプロファイルを変更しました。保存しますか？[1 -

/bin/ping-x】です。]  
(変更履歴を見る / 選択したプロファイルを保存する / [ 変更履歴を見る ] / プロファイルの変更履歴を見る / 省略する

新しいプロファイルを保存するには、Sを入力します：

bin/ping-xの更新プロファイルを書き込んでいます。

プロファイリングしています：/bin/ping-x

[AppArmorイベントのシステムログを取得する] / (F)inish

Fを入力すると終了します：

/bin/ping-x を enforce モードに設定する。

AppArmorのプロファイルをenforceモードで再

読み込み。

新しいプロフィールの投稿をご検討ください！

詳しくは以下のWikiページをご覧ください：

<http://wiki.apparmor.net/index.php/Profiles>

bin/ping-xのプロファイルの生成を終了しました。

6. `etc/apparmor.d/bin.ping-x`に保存されている、作成したプロファイルを表示します。

```
student@ubuntu:~$ sudo cat /etc/apparmor.d/bin.ping-x #
Last Modified : 火曜日 10月17日 11:30:47 2017
#インクルード <tunables/global>を使用します。
```

```
/bin/ping-x { です。
#include <abstractions/base>
#include <abstractions/nameservice>
です。
```

```
capability net_raw, ネット
```

```
トワーク inet raw、
```

```
/bin/ping-x mr、
/lib/x86_64-linux-gnu/ld-*.so mr、
```

```
}
```

7. **aa-genproc** ユーティリティは、新しいポリシーをインストールしてアクティブにするので、すぐに使用できるはずですが。また、`systemctl reload apparmor` コマンドでポリシーを必要に応じて再ロードすることができます。潜在的な問題を回避し、変更が存続することを確認するために、システムを再起動します。

システムが再起動したら、ユーザーの学生として、新しいプロファイルを有効にして **ping-x** がまだ機能することを確認します。ipアドレスでlocalhostにpingを打つ：

```
student@ubuntu:~$ ping-x -c3 -4 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) バイトのデータです。
127.0.0.1 から 64 バイト: icmp_seq=1 ttl=64 time=0.057 ms
127.0.0.1 から 64 バイト: icmp_seq=2 ttl=64 time=0.043 ms
127.0.0.1 から 64 バイト: icmp_seq=3 ttl=64 time=0.095 ms

--- 127.0.0.1 pingの統計 ---
3パケット送信、3パケット受信、パケットロス0%、時間2027ms rtt
min/avg/max/mdev = 0.043/0.065/0.095/0.021ms
```

8. これは期待通りに動作するはずですが。このプロファイルは非常に特殊で、AppArmorは指定されたパラメータ以外の機能を許可しません。**AppArmor**がこのアプリケーションを保護していることを確認するために、**IPV6** localhostアドレスにpingを送信してみます。

これは失敗するはずですが：

```
student@ubuntu:~$ ping-x -c3 -6 ::1
ping: socket : パーMISSIONが拒否されました
```

(なお、-6オプションは**IPv6のみ**を使用することを意味し、:1は**IPv6におけるローカルホスト**である)。

出力は、ソケットの問題があることを示します。システムログを調べると、**ping-x** プログラムが**AppArmor**内で**IPv6**にアクセスできないことが判明します：

```
766:104): apparmor="DENIED" operation="create" profile="/bin/ping-x"
pid=2709 comm="ping-x" family="inet6" sock_type="raw" protocol=58
requested_mask="create" denied_mask="create
```

9. この欠陥を修正するには、先ほどと同様に**aa-genprof**を再実行し、**window2**で**IPv6**ループバックをpingし、追加オプションを追加します。



## 第36章

# システムレスキュー



|      |                             |     |
|------|-----------------------------|-----|
| 36.1 | レスキューメディアとトラブルシューティング ..... | 590 |
| 36.2 | レスキュー/リカバリーメディアの使用 .....    | 591 |
| 36.3 | システムレスキューとリカバリー .....       | 592 |
| 36.4 | エマージェンシーブートメディア .....       | 593 |
| 36.5 | レスキューメディアの利用 .....          | 594 |
| 36.6 | エマージェンシーモード .....           | 596 |
| 36.7 | シングルユーザーモード .....           | 597 |
| 36.8 | 研究室 .....                   | 598 |

### 学習目標

このセッションが終わるころには、次のことができるようになっているはずです：

- システムレスキューメディアはどのような形態で提供され、どのように用意されるのか、または用意できるのかを説明する。
- レスキューメディアを使用して、システムの修復を支援します。
- 緊急モードへの移行方法とそこでできることを知る。

589



## 36.1 レスキューメディアとトラブルシューティング

# レスキューメディアとトラブルシューティ

- レスキュー/リカバリーメディアはトラブルシューティングにも使える
- 光学メディア（**CD**、**DVD**）、**USB**メモリで登場。
- トラブルシューティング（修理）のための便利なユーティリティを収録しています：
  - ディスクメンテナンスユーティリティ
  - ネットワーキング・ユーティリティ
  - その他ユーティリティ
  - ロギングファイル

システムのレスキューとリカバリについては後ほど説明しますが、ここではトラブルシューティングにおけるレスキューメディアの使用に焦点を当てましょう。正確なツールの選択はLinuxディストリビューションによって異なりますが、インストールまたはライブ**CD/DVD**、**USB**から起動する場合

ドライブで、「Rescue Installed System」のような名前のオプションを選択できるようになります。

レスキューディスクには、以下のような多くの便利なプログラムが含まれています：

- パーティションの作成、**RAID**デバイスの管理、論理ボリュームの管理、ファイルシステムの作成などを行うディスクユーティリティです：  
**fdisk**、**mdadm**、**pvccreate**、**vgcreate**、**lvcreate**、**mkfs**、その他多数。
- ネットワークのデバッグやネットワーク接続のためのネットワーク・ユーティリティです：  
**ifconfig**、**route**、**traceroute**、**mtr**、**host**、**ftp**、**scp**、**ssh**。
- その他、多数のコマンドを用意しています：  
**bash**、**chroot**、**ps**、**kill**、**vi**、**dd**、**tar**、**cpio**、**gzip**、**rpm**、**mkdir**、**ls**、**cp**、**mv**、**rm**などなどです。



## 36.2 レスキュー／リカバリーメディアの使用

# レスキュー/リカバリーメディアイメージの

- 可能であれば、ファイルシステムのマウントを提案します（通常は `/mnt/sysimage`）。
- その環境に **chroot** することを提案する場合があります。
- ローカルディスクやネットワークからのソフトウェアのインストールをお手伝いすることがあります。

レスキューイメージは、起動時にいくつかの質問をします。そのうちのひとつが、ファイルシステムをマウントするかどうかです（マウントできる場合）。

もしそうなら、それらはどこかにマウントされており、通常は `/mnt/sysimage` にあります。そのディレクトリに移動してファイルにアクセスするか、あるいはその環境に

```
$ sudo chroot /mnt/sysimage
```

ネットワークベースのレスキューの場合、`/mnt/source` をマウントするように指示されることがあります。

**chroot** 化された環境の中からソフトウェアパッケージをインストールすることができます。また、chroot環境の外からインストールすることも可能です。例えば、**RPM** ベースのシステムでは、`--root` オプションでルート・ディレクトリの場所を指定することでインストールできます：

```
$ sudo rpm -ivh --force --root=/mnt/sysimage /mnt/source/Packages/vsftpd-2*.rpm
```

### 36.3 システムレスキューとリカバリー

## システムレスキューとリカバリー

- システムレスキューメディアは複数の形態で提供される
- エマージェンシーモードに移行して、便利な操作を行うことができます。
- また、エマージェンシーモードとは異なるシングルユーザーモードへの移行方法も知っておくと便利です。

システムには遅かれ早かれ、起動しない、ファイルシステムがマウントできない、デスクトップが表示されない、などの重大な障害が発生する可能性があります。このような場合、光ディスクやポータブルUSBドライブなどのシステムレスキューメディアを使用することで、状況を修復することができます。緊急モードまたはシングルユーザーモードで起動すると、Linuxの全ツールを使用してシステムを正常な機能に修復することができます。

## 36.4 エマージェンシーブートメディア

# エマージェンシーブートメディア

- CD
- DVD
- USB
- スタンドアロンのレスキュー/リカバリツール、ディストリビューションのインストールディスク、レスキュー/リカバリ機能を備えたライブLinuxディストリビューションのいずれかである。

緊急ブートメディアは、ファイルの紛失、設定ミス、破損、サービスの設定ミスなど、何らかの問題でシステムが起動しない場合に有効です。

また、rootパスワードが何らかの原因で失われたり、スクランブルされたりしてリセットが必要な場合にも、レスキューメディアは有効です。

ほとんどのLinuxディストリビューションでは、インストールメディア（CD、DVD、USB）やLiveメディアをレスキューディスクと兼用することができ、非常に便利です。また、特殊な用途のレスキューディスクも用意されています。

ライブメディアは、ディスクからロードするのではなく、メモリ上で動作する完全で起動可能なオペレーティングシステムを提供します（あらゆる形式）。ユーザーは、オペレーティングシステムやLinuxディストリビューションを実際にインストールしたり、コンピュータ上の既存のオペレーティングシステムに変更を加えたりすることなく、体験・評価することができます。

ライブリムーバブルメディアは、ハードディスクドライブなどの2次記憶装置がないパソコンや、ハードディスクドライブやファイルシステムが壊れたパソコンでも動作し、データを救出することができる点が特徴です。

## 36.5 レスキューメディアの利用

# レスキューメディアの利用

- 事実上すべてのLinuxディストリビューションは、**GRUB**メニューの選択肢としてレスキューイメージを用意しています。もしあなたのシステムがここまで来て、レスキューオプションが機能するのであれば、レスキューメディアをマウントする前に試してみるべきでしょう。
- システム起動時に、以下のメニューを選択する必要があります。のようなレスキューやタイピングを行います。

### リナックスレスキュー

ブート：プロンプトで

- ネットワークを開始するかどうかなど、言語やその他の選択肢について質問されることがあります。
- また、どのファイルシステムをマウントするか等も聞かれるでしょう。

Liveメディア、インストールメディア、レスキューメディアのいずれを使用しても、レスキューやリカバリーのための特別なOSに入る手順は同じで、これまで指摘してきたように、1つのメディアが3つの目的すべてに対応します。

レスキュー/リカバリーモードは、リムーバブルメディアからシステムを起動する際に、ブートメニューのオプションからアクセスすることができます。多くの場合、このような行でrescueと入力する必要があるかもしれません：

### ブートLinuxレスキュー

各ディストリビューションは、多少異なるものの、確認しやすい手順を採用しているため、すべての可能性をお伝えすることはできません。

次に、どの言語で続けるかなどの質問と、ディストリビューションに依存した選択をすることが予想されます。次に、有効なレスキューイメージがある場所を選択するように促されます：**CD/DVD**、**ハードディスク**、**NFS**、**FTP**、**HTTP**のいずれかを選択します。

選択した場所には有効なインストールツリーが含まれている必要があり、インストールツリーはレスキューディスクと同じLinuxバージョンである必要があります。ペンダーからダウンロードしたboot.isoイメージを使用する場合は、ネットワークベースのインストールツリーも必要です。

ファイルシステムのマウントについて追加の質問があります。ファイルシステムが見つければ、`/mnt/sysimage`の下にマウントされます。その後、シェルプロンプトが表示され、様々なユーティリティにアクセスし、システムを適切に修正することができます。

**chroot** は、ルート (`/`) ファイルシステムによりよくアクセスするために使用されます。

## レスキューUSBキー

- **USB**インストール/ライブ/レスキューディスクイメージは、最近のすべてのディストリビューションで利用可能になっています。
- ほとんどの場合、**USB**ドライブに転送することができます：  
`$ dd if=boot.iso of=/dev/sdX`  
`dev/sdX`がリムーバブルドライブの場合
- **BIOS**が**USB**ドライブから起動するように設定されていることを確認してください。
- **livecd-tools**や**liveusb-creator**などのツールは、このプロセスをより簡単にし、ディスクをカスタマイズすることも可能です。

多くのディストリビューションでは、`boot.iso`のイメージファイルをダウンロードできるようになっています（名称は異なる場合があります）。これを`dd`を使って**USB**のようにキードライブ：

```
$ dd if=boot.iso of=/dev/sdX
```

リムーバブルドライブが`/dev/sdX`として認識されていることを前提に、この操作を行います。この操作を行うと、ドライブ上の既存のコンテンツが消去されますのでご注意ください！

お使いのシステムが**USB**メディアから起動する機能を持ち、**BIOS**が設定されている場合、この**USB**ドライブから起動することができます。この**USB**ドライブは、レスキューCDやDVDと同じように機能します。ただし、インストールツリーは**USB**ドライブに存在しないので、この方法では、ネットワークベースのインストールツリーが必要な場合は、それが必要です。

**livecd-tools**や**liveusb-creator**などの便利なユーティリティは、インストールイメージを取得する場所としてローカルドライブかインターネットを指定でき、ブータブルイメージを構築してリムーバブルドライブに書き込むという難しい作業をすべて行ってくれる。これは非常に便利で、事実上すべてのLinuxディストリビューションで動作します。

## 36.6 エマージェンシーモード

# エマージェンシーモード

- 緊急モードは、最も最小限の環境で起動します。
- ルートファイルシステムを読み取り専用にする
- `init` スクリプトが実行されないなので、ほとんどセットアップが行われません。
- `init` が失敗しているときに便利
- ファイルシステムがマウントされたままなので、データ復旧に役立つ
- 緊急モードを選択するために、**GRUB** インタラクティブメニューと対話する必要があります。
- コマンドプロンプトを表示する前に、`root` のパスワードが要求されます。
- ファイルシステムが破損している場合など、起動に失敗すると、システムが自動的に緊急モードに入ることがあります。

**緊急モードでは**、可能な限り最小限の環境に起動されます。ルートファイルシステムは読み取り専用マウントされ、`init` スクリプトは実行されず、ほとんど何もセットアップされません。

シングルユーザーモード（次に説明します）に対する緊急モードの主な利点は、`init` が破損したり動作しない場合でも、再インストール時に失われるかもしれないデータを回復するためにファイルシステムをマウントできることです。

緊急モードに入るには、**GRUB** ブートメニューからエントリーを選択し、`e` キーを押して編集する必要があります。そして、システムの起動を指示する前に、カーネルコマンドラインに `emergency` という単語を追加してください。シェルプロンプトが表示される前に、`root` のパスワードが要求されます。

ファイルシステムの破損など、さまざまな理由で起動に失敗した場合にも、エマージェンシーモードに入ることがあります。

## 36.7 シングルユーザーモード

# シングルユーザーモード

- システムは起動するが、ログインを許可しない場合に使用される。
- システムはランレベル1に起動します。
  - initが開始されます。
  - サービスが開始されない。
  - ネットワークは起動していません。
  - 可能なすべてのファイルシステムがマウントされます。
  - rootアクセスはパスワードなしで許可されます。
  - システムメンテナンス用のコマンドラインシェルが起動します。

システムは起動するが、起動が完了してもログインできない場合は、シングルユーザーモードを試してみてください。

このモードでは、システムはランレベル1（**SysVinit**言語）で起動します。シングルユーザーモードは、自動的にファイルシステムのマウントを試みるため、ルートファイルシステムが正常にマウントできない場合や、**init**の設定が壊れている場合は、使用することができません。

シングルユーザーモードで起動するには、緊急モードと同じ方法を使用しますが、1つの例外として、キーワード **emergency** をキーワード **single** に置き換えます。

## 36.8 研究室

### 練習問題36.1レスキュー／リカバリーメディアを使うための準備



#### 非常に重要

以下の演習では、意図的にシステムにダメージを与え、レスキューメディアを使用して回復させる予定です。したがって、より野心的なことをする前に、レスキューメディアから本当に起動できることを確認することが賢明です。

そのため、まずレスキューメディア（専用のレスキュー／リカバリーイメージ、または光学ディスクかUSBドライブにインストールまたはライブイメージ）を用意する必要があります。

それを起動し、レスキューメディアからシステムを強制的に起動させる方法を確認し（**BIOS設定をいじらなければ**ならない可能性が高い）、システムが起動したらレスキューモードを選択します。



#### ご注意ください

仮想マシンを使用する場合、2つの違いを除いて論理的には同じ手順です：

- **BIOSへのアクセス**は、使用するハイパーバイザーによって難しいかもしれません。中には非常に素早いキー入力が必要なものもあるので、ドキュメントを読んでやり方を確認してください。
- 物理的な光ディスクやドライブを使用することもできますが、仮想マシンの設定でマウントされていることを確認し、**USBの場合は**、仮想マシンが物理デバイスを主張できるようにするためのいくつかのハードルがあるかもしれません。通常、.isoイメージファイルを仮想マシンに直接接続の方が簡単です。

仮想マシンで作業している場合、明らかに物事の危険性は低くなります。もし、修正不可能な方法でシステムを破損することを恐れるなら、これらの演習を行う前に仮想マシンイメージのバックアップコピーを作成するだけで、後でいつでもそのイメージと置き換えることができます。



#### 非常に重要

レスキュー／リカバリーメディアからシステムを起動できることを確認しない限り、以下の演習を行わないでください！

### 練習36.2：破損したGRUBの設定から回復する

1. **GRUB設定ファイル**（`/boot/grub/grub.cfg`、`/boot/grub2/grub.cfg`、`/boot/grub/grub.conf`）を編集し、UUIDというフィールドの値の最初の文字を削除してカーネルラインを修正します。どの文字を削除したかは、レスキューモードで置き換えるので、メモしておいてください。（ルートファイルシステムがラベルまたはハードディスクデバイスノードで識別されている場合、同様の簡単な変更を行います）。オリジナルのバックアップコピーを取っておきます。



#### BLSCFGシステム上

代わりに `/etc/grub2/grubenv` を編集することで、コマンドラインを破壊することができます。

2. マシンを再起動します。No root device was found などと表示され、起動に失敗します。また、パニックが発生したことも確認できます。
3. インストールまたはライブDVD、CD、USBドライブ（またはネットワークブートメディア）をマシンに挿入します（機能するインストールサーバーにアクセスできる場合は、ネットワークブートメディア）。もう一度再起動します



。ブートメニューが表示されたら、レスキューモードへの移行を選択します。

4. 代替案として、**GRUB**メニューからレスキューイメージを選択することができます；ほとんどのディストリビューションで提供されています。ほとんどのディストリビューションで提供されています。

レスキューメディアを使用するのと同じような体験ができますが、必ずうまくいくわけではありません。例えば、ルートファイルシステムが損傷している場合、何もできなくなります。

5. レスキューモードでは、ファイルシステムの検索を求められたら同意します。プロンプトが表示されたら、シェルを開き、`mount`や`ps`などのユーティリティを実行して、レスキューシステムを探索します。
6. **GRUB**設定ファイルを編集するか、バックアップコピーから復元して、壊れたシステムを修復してください。
7. `exit`と入力してインストーラーに戻り、ブートメディアを取り出し、再起動方法の説明に従ってください。マシンを再起動してください。正常に起動するはずですが。

### 練習問題36.3: パスワードの失敗から回復する

1. `root`で（**`sudo`ではありません**）、`root`のパスワードを変更します。新しいパスワードが何であるか知らないふりをします。
2. ログアウトして、古いパスワードを使って`root`で再ログインしてみます。明らかに失敗します。
3. レスキューメディアを使用して起動し、オプションでレスキューを選択します。ファイルシステムをマウントさせ、コマンドラインシェルに移行させる。
4. `chroot`化された環境（システムに普通にアクセスできる状態）に入る：  

```
$ chroot /mnt/sysimage
```

  
と、`root`のパスワードをリセットして元の値に戻す。
5. 終了してレスキューメディアを取り出し、再起動すると、これで正常にログインできるはずです。

### 練習問題36.4: パーティションテーブルの破損から回復する



#### 非常に重要

この作業は危険であり、使用不可能なシステムを残すことになりかねません。必ず理解してから行ってください。



#### ご注意ください

**MBRシステム**の場合は以下の手順で、**GPT**の場合は`sgdisk`に`-backup-file`を付けて使用する必要があります。  
`--load-backup` オプションはパーティションの章で説明したとおりです。

1. `root`でログインし、**MBR**を保存します：

```
$ dd if=/dev/sda of=/root/mbrsave bs=446 count=1
```

```
で1+0レコード。
1+0レコードアウト
446/バイト (446B) コピー、0.00976759秒、45.7kB/秒
```

注意：上記のコマンドを正確に発行し、保存されるファイルの長さが正しいことを確認してください：

```
$ sudo ls -l /root/mbrsave
```

```
-rw-r--r-- 1 root root 446 Nov 12 07:54 mbrsave
```

2. では、**MBR**を一緒に消していきます：

```
$ dd if=/dev/zero of=/dev/sda bs=446 count=1
```

で1+0レコード。  
1+0レコードアウト  
446バイト ( 446B ) コピー、0.000124091秒、3.6MB/秒

3. システムを再起動すると、失敗するはずです。
4. レスキュー環境に再起動し、**MBR**を復元する :  
`$ dd if=/mnt/sysimage/root/mbrsave of=/dev/sda bs=446 count=1`
5. レスキュー環境から終了し、再起動します。これでシステムは正常に起動するはずです。

## 練習36.5: インストールイメージを使ったりカバリー



### ご注意ください

この演習は、Red Hatベースのシステム向けに特別に作成されています。他のディストリビューションファミリーのために適切な置換を簡単に構築できるはずでず。

1. **zsh** パッケージを削除します (インストールされている場合!) :

```
s$ dnf remove zsh
```

または

```
rpm -e zsh
```

ここでは、問題を簡単にするために、一般的に依存関係のないパッケージを選択したことに注意してください。もし、依存関係があるものを選んだ場合は、以下のステップに注意し、他のものを削除した場合は、必要に応じて再インストールする必要があります。

2. レスキュー環境にブートする。
3. レスキュー環境内から**zsh**を再インストール (またはインストール) します。まず、インストールメディアを/mnt/sourceにマウントします :

```
マウント /dev/cdrom /mnt/source
```

その後、パッケージを再インストールしてください :

```
rpm -ivh --force --root /mnt/sysimage /mnt/source/Packages/zsh*.rpm
```

forceオプションは、依存関係の情報などを決定する際にソースディレクトリを使用するようrpmに指示します。インストールイメージがあなたのシステムよりずっと古く、何度も更新されている場合、すべての手順が破綻する可能性があることに注意してください !

4. 終了して再起動します。
5. **zsh**が再インストールされていることを確認します :

```
$ rpm -q zsh
```

```
zsh-5.0.2-7.el7.x86_64
```

6. `$ zsh`

```

[coop@q7]/tmp/LFS201%
```

## 第37章

# クロージングと評価アンケート



|      |               |     |
|------|---------------|-----|
| 37.1 | 評価アンケート ..... | 601 |
|------|---------------|-----|

601



## 37.1 評価アンケート

### 評価アンケート

この度は、**Linux Foundation**コースをご受講いただきありがとうございます。  
お客様のご要望にどれだけお応えできたか、また今後のセッションの改善に役立てるため、お客様のご意見は私たちにとって重要なものです。

- インストラクターが提供するリンクから、トレーニングの評価をしてください。
- 証明書にお名前が記載されますので、必ずお名前のスペルを確認し、正しい大文字を使用してください。
- この情報は、あなたの**Cer-**を生成するために使用されます。  
**tificate of Completion**、記入したメールアドレスに送信されますので、間違いがないことを確認してください。



図 37.1 : コースサーベイの様子