



LFS307

Linuxシステム 管理

バージョン1.4



バージョン1.4

© Copyright the Linux Foundation 2023.無断複写・転載を禁じます。

© Copyright the Linux Foundation 2023.無断複写・転載を禁じます。

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教材は、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・コードには、他の著作権者が存在する場合があります、適用されるオープン・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るものです。The Linux Foundation の書面による事前の承諾なしに、これらを複製、変更、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、書面による明示的な事前の同意なしに、複製、複写、検索システムへの保存、または送信することを禁じます。

出版社

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、明確に否認されます。

本書では、サードパーティのアプリケーション・ソフトウェア・パッケージが参照されることがありますが、これはデモンストレーションのみ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。本コースに含まれるその他の商標は、それぞれの所有者に帰属します。

ここに掲載されている資料の適切かつ公正な使用についてご質問がある場合は、<https://trainingsupport.linuxfoundation.org>。

機密情報の非開示

「(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する情報、(c)オープンプロジェクトに関わる一般的なビジネス事項に関連する情報。

このコースには機密情報は含まれません。また、機密情報を授業中に漏らしてはなりません。

V 1.

4© Copyright the Linux Foundation 2023.無断複写・転載を禁じます。

内容

1 はじめに	1
1.1 リナックス・ファンデーション	2
1.2 Linuxファウンデーショントレーニング	4
1.3 Linux Foundation認定資格	8
1.4 Linux Foundation デジタルバッジ	11
1.5 実験演習、解答とリソース	12
1.6 Linuxとオープンソースプロジェクトの変化	14
1.7 EラーニングコースLF207	15
1.8 ディストリビューション詳細	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 データの区別	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 /ビン	32
2.6 /ブート	34
2.7 /dev	35
2.8 /etc	36
2.9 /ホーム	37
2.10 /libおよび/lib64	39
2.11 /メディア	40
2.12 /mnt	41
2.13 /オプト	42
2.14 /プログラム	43
2.15 /シス	45
2.16 /ルート	46
2.17 /sbin	47
2.18 /srv	49
2.19 /tmp	50
2.20 /usr	51

2.21	/var	52
2.22	/ラン	53
2.23	研究室	54

3 ユーザー環境 57

3.1	環境変数	58
3.2	コマンドの歴史	62
3.3	コマンド・エイリアス	66
3.4	研究室	67

4 ユーザーアカウント管理 71

4.1	ユーザーアカウント	72
4.2	シェル起動ファイル	75
4.3	ユーザーアカウントの管理	78

iii

4.4	ロックされたアカウント	80
4.5	パスワード	81
4.6	/etc/shadow	82
4.7	パスワード管理	84
4.8	パスワードのエージング	85
4.9	ルートアカウント	86
4.10	SSH	88
4.11	研究室	95

5 グループ経営 99

5.1	グループ	100
5.2	団体会員	101
5.3	グループ経営	102
5.4	ユーザー・プライベート・グループ	103
5.5	研究室	104

6 ファイルのパーミッションと所有権 107

6.1	ファイルのパーミッションと所有権	108
6.2	ファイルアクセス権	109
6.3	chmod、chown、chgrp	110
6.4	ウマスク	113
6.5	ファイルシステムACL	115
6.6	研究室	116

7 パッケージ管理システム 119

7.1	なぜパッケージを使うのか？	120
7.2	ソフトウェアパッケージングコンセプト	121
7.3	パッケージの種類	122
7.4	利用可能なパッケージ管理システム	123
7.5	パッケージングツールのレベルと種類	124
7.6	パッケージ・ソース	125
7.7	ソフトウェア・パッケージの作成	126
7.8	リビジョン管理システム	127
7.9	利用可能なソース管理システム	128
7.10	Linuxカーネルとgit	129
7.11	研究室	131
8	dpkg	135
8.1	DPKG (Debian パッケージ)	136
8.2	パッケージのファイル名とソース	137
8.3	DPKGクエリ	138
8.4	インストール/アップグレード/アンインストール	139
8.5	研究室	140
9	APT	141
9.1	APT	142
9.2	APT ユーティリティーズ	143
9.3	クエリー	144
9.4	パッケージのインストール/移動/アップグレード	145
9.5	清掃	146
9.6	研究室	147
10	回転数	149
10.1	RPM (Red Hat Package Manager)	150
10.2	パッケージのファイル名	151
10.3	RPM データベースとヘルパープログラム	152
10.4	クエリー	154
10.5	パッケージの検証	155
10.6	パッケージのインストールと削除	157
10.7	RPM パッケージの更新、アップグレード、およびフレッシュ化	159
10.8	Linuxカーネルのアップグレード	161
10.9	rpm2archive と rpm2cpio	162
10.10	研究室	163
11	dnfとyum	167
11.1	dnf	168

11.2	うん	169
11.3	クエリ	170
11.4	パッケージのインストール/移動/アップグレード	171
11.5	追加の dnf コマンド	172
11.6	研究室	173
12	ザッパー	177
12.1	ザッパー	178
12.2	クエリー	179
12.3	パッケージのインストール/移動/アップグレード	180
12.4	zypper の追加コマンド	181
12.5	研究室	182
13	GITの基礎	185
13.1	リビジョン管理	186
13.2	基本コマンド	188
13.3	gitコマンドを試す	190
13.4	Gitでソフトウェアを入手する	196
13.5	研究室	202
14	プロセス	205
14.1	プログラムとプロセス	206
14.2	プロセス・リミット	210
14.3	プロセスの作成	213
14.4	プロセス制御	215
14.5	未来のスタートプロセス	218
14.6	プロセス・ステート	222
14.7	実行モード	223
14.8	デーモン	226
14.9	やさしさ	227
14.10	研究室	229
15	プロセス・モニタリング	237
15.1	プロセス監視	238
15.2	トラブルシューティング	239
15.3	ps	241
15.4	プスツリー	243
15.5	トップ	245
15.6	研究室	247
16	メモリの監視、使用状況、スワップの設定	249

16.1	メモリのモニタリングとチューニング	250
16.2	/proc/sys/vm	252
16.3	ブイエムスタット	253
16.4	スワップ	254
16.5	メモリ不足キラー (OOM)	255
16.6	研究室	257
17	I/Oの監視とチューニング	261
17.1	I/Oモニタリング	262
17.2	イオスタット	263
17.3	イオトップ	264
17.4	研究室	265
18	コンテナの概要	269
18.1	容器	270
18.2	コンテナと仮想マシン	271
18.3	ドッカー	272
18.4	カウセイの例	275
18.5	再現可能なビルド	279
18.6	研究室	283
19	LinuxファイルシステムとVFS	287
19.1	ファイルシステムの基本	288
19.2	ファイルシステム の概念	289
19.3	仮想ファイルシステム (VFS)	291
19.4	利用可能なファイルシステム	292
19.5	ファイルシステムのジャーナリング	294
19.6	特殊ファイルシステム	295
19.7	研究室	296
20	ディスク・パーティショニング	299
20.1	一般的なディスクの種類	300
20.2	ディスク・ジオメトリ	301
20.3	パーティショニング	302
20.4	パーティション・テーブル	304
20.5	ディスクデバイスの命名	306
20.6	blkidとlsblk	307
20.7	パーティションのサイズアップ	309
20.8	パーティションテーブルのバックアップと復元	310
20.9	パーティションテーブルエディター	312

20.10	エフディスク	313
20.11	研究室	314
21	ファイルシステムの機能：属性、作成、チェック、使用、マウント	319
21.1	拡張属性	320
21.2	ファイルシステムの作成とフォーマット	321
21.3	ファイルシステムのトラブルシューティング	322
21.4	ファイルシステムのチェックと修復	324
21.5	ファイルシステムの使用	325
21.6	ディスク使用量	326
21.7	ファイルシステムのマウント	327
21.8	NFS	331
21.9	ブートと/etc/fstabでのマウント	332
21.10	オートマウント	334
21.11	ネットワーク・ブロック・デバイス	336
21.12	研究室	340
22	Ext4ファイルシステム	349
22.1	ext4 機能	350
22.2	ext4 のレイアウトとスーパーブロックとブロックグループ	351
22.3	dumpe2fs	354
22.4	tune2fs	355
22.5	研究室	356
23	論理ボリューム管理（LVM）	361
23.1	論理ボリューム管理（LVM）	362
23.2	ボリュームとボリュームグループ	363
23.3	論理ボリュームの操作	364
23.4	論理ボリュームのサイズ変更	367
23.5	LVM スナップショット	368
23.6	研究室	369
24	カーネルサービスと構成	371
24.1	カーネルの概要	372
24.2	カーネルブートパラメータ	373
24.3	カーネルコマンドライン	374
24.4	ブートプロセスの失敗	375
24.5	sysctl	376
24.6	研究室	377
25	カーネルモジュール	379

25.1	カーネルモジュール	380
25.2	モジュール・ユーティリティ	382
25.3	モデインフォ	384
25.4	モジュール構成	385
25.5	研究室	386
26	デバイスと udev	387
26.1	udev とデバイス管理	388
26.2	デバイス・ノード	389
26.3	規則	392
26.4	研究室	395
27	ネットワークアドレス	397
27.1	IPアドレス	398
27.2	IPv4アドレスの種類	399
27.3	IPv6アドレスの種類	401
27.4	IPアドレスクラス	402
27.5	ネットマスク	403
27.6	ホスト名	404
27.7	NTP	405
27.8	研究室	411
28	ネットワークデバイスと構成	413
28.1	ネットワーク機器	414
28.2	予測可能なネットワークインターフェースデバイス名	415
28.3	ネットワーク設定ファイル	417
28.4	ネットワーク・マネージャー	419
28.5	ルーティング	424
28.6	仮想ネットワークインターフェース	427
28.7	DNSと名前解決	431
28.8	ネットワークのトラブルシューティング	436
28.9	ネットワーク診断	437
28.10	研究室	441
29	LDAP	445
29.1	LDAP認証	446
29.2	ラボ	450
30	ファイアウォール	459
30.1	ファイアウォール	460
30.2	インターフェイス	463

30.3	ファイアウォール	465
30.4	ゾーン	468
30.5	ソース管理	472
30.6	サービスと港湾管理	473
30.8	研究所	481
31 System Init: systemdの歴史とカスタマイズ		485
31.1	イニシャルプロセス	486
31.2	スタートアップの選択肢	487
31.3	システムド	488
31.4	systemctl	490
31.5	研究室	491
32 バックアップとリカバリーの方法		495
32.1	バックアップの基本	496
32.2	バックアップとアーカイブ	498
32.3	バックアップの方法と戦略	500
32.4	タール	503
32.5	圧縮: gzip、bzip2、xzとバックアップ	506
32.6	dd	507
32.7	rsync	508
32.8	バックアップ・プログラム	509
32.9	研究室	510
33 Linuxセキュリティ・モジュール		513
33.1	Linuxセキュリティ・モジュール	514
33.2	SELinux	516
33.3	AppArmor	531
33.4	研究室	537
34 システム・レスキュー		543
34.1	レスキューメディアとトラブルシューティング	544
34.2	レスキュー／リカバリーメディアの使用	545
34.3	システムの救出と回復	546
34.4	エマージェンシー・ブート・メディア	547
34.5	レスキューメディアの使用	548
34.6	緊急モード	550
34.7	シングルユーザーモード	551
34.8	研究室	552
35 決算・評価調査		555
35.1	評価調査	555



ご注意

** これらのセクションは、一部または全部をオプションとすることができる。**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。講師は、授業の経験や時間的制約に応じて、これらの項目を取り上げるか取り上げないかを選択することができます。

図表一覧

2.1	/bin ディレクトリ: 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	/ホームディレクトリ	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	/ディレクトリ	46
2.11	/sbinディレクトリ: Ubuntu 18.04	48
2.12	/ディレクトリ	52
2.13	/ディレクトリ	53
4.1	usermodの使用	79
4.2	チャージの使用	85
6.1	chmodの使用	111
7.1	RPMとAPT	123
8.1	Ubuntuでソースパッケージを入手する	137
10.1	RPMパッケージのアンインストール	158
11.1	rpmリポジトリ	168
14.1	リミット	210
14.2	ユーザーとカーネルのモードとシステムコール	224
15.1	BSD オプションで ps を使う	241
15.2	ps 出力のカスタマイズ	242

15.3	ps と UNIX オプションの使用	242
15.4	トップ	245
15.5	/proc 内容	246
16.1	/proc/sys/vm	252
16.2	バイエムスタット	253
16.3	vmstatの使用	253
16.4	つのディスクでvmstatを使う	253
17.1	イオスタット	263
17.2	iostatの使用	263
17.3	iotopの使用	264
17.4	iotopオプション	264
18.1	LinuxコンテナとOCIコンテナの比較	270

19.1	ハードリンクとソフトリンク	290
20.1	fdiskの使用	301
20.2	MBRディスクパーティションテーブル	304
20.3	GPT ディスクパーティションテーブル	305
20.4	blkidの使用	307
20.5	lsblk の使用	308
21.1	mkfs	321
21.2	フスク	324
21.3	dfを使用	325
21.4	デュオの使用	326
21.5	マウント	328
21.6	現在マウントされているファイルシステム	329
21.7	/etc/fstab の例	333
21.8	自動マウントの例	335
21.9	ネットワーク・ブロック・デバイス	339
22.1	ext[3,4] ファイルシステムのレイアウト	351
22.2	ブロック・グループ	352
23.1	LVMコンポーネント	363
23.2	論理ボリュームユーティリティ	364
23.3	物理ボリューム・ユーティリティ	364
23.4	ボリューム・グループ・ユーティリティ	366
24.1	sysctl	376
25.1	lsmod の使用	381
25.2	モデインフォ	384
26.1	デバイス・ノード	390
26.2	udev ルール	391
27.1	IPアドレス	398
27.2	ネットワーク・タイム・プロトコル	405
28.1	nmtui メイン画面	421
28.2	nmtui 編集画面	421
28.3	nmtui ワイヤレス設定	422
28.4	routeとip routeの使用	424
28.5	DNS	434

28.6	ピング	438
28.7	トレースルート	439
28.8	メートル	440
29.1	LDAPクライアント認証プロセス	448
29.2	ターンキーOpenLDAPアプライアンスサービス	450
29.3	Wireshark LDAPパケットトレース	456
30.1	宛先NAT	478
30.2	ソースNATとMASQUERADE	480
30.3	SNATまたはMASQUERADEの例	483
33.1	SELinux強制モード	517
33.2	SELinux パーミッシブモード	517
33.3	セマネージ	529
35.1	コース調査	555

テーブル一覧

2.1	主なディレクトリー その	29
2.2	主なディレクトリ その	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	利用可能なソース管理システム	128
10.1	rpm クエリーコマンドの例	154
15.1	プロセスと負荷の監視ツール	238
16.1	メモリ監視ユーティリティ	250
19.1	特殊ファイルシステム	295
33.1	AppArmor ユーティリティ	535

第1章

はじめに



1.1 リナックス・ファンデーション 2

1.2 Linuxファウンデーショントレーニング 4

1.3 Linux Foundation認定資格 8

1.4 Linux Foundation デジタルバッジ 11

1.5 実験演習、解答とリソース 12

1.6 Linuxとオープンソースプロジェクトの変化 14

1.7 EラーニングコースLF207 15

1.8 ディストリビューション詳細 16

1.9 研究室 23

1.1 Linuxファウンデーション

リナックス・ファウンデーションとは？

- 非営利のコンソーシアムである：
 - リナックス
 - 他の多くのオープンソースソフトウェア（OSS）プロジェクトとコミュニティ
- 持続可能なOSSエコシステムの構築を支援する：
 - 財政的・知的資源とサービス
 - トレーニング
 - イベント
- 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る。
- 一流のテクノロジー企業や開発者が、神経質なまでの協力体制でサポート
- <https://linuxfoundation.org>。

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 を超えるメンバーに支えられ、オープンソースソフトウェア、オープンスタンダード、オープンデータ、オープンハードウェアのコラボレーションを推進する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対応することで、オープンコラボレーションの持続可能なモデルを構築することに重点を置いています。

リナックス・ファウンデーションは、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

であるリナックスを主催している。また、**Linux**の生みの親であるリーナス・トーバルズとリード・メンテナであるグレッグ・クロア＝ハートマンの本拠地でもある。**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きました。

その結果、**Linux Foundation** は今日、**Linux** 以上に多くの重要なオープンソース プロジェクトをホストしています。ビッグデータおよびアナリティクス、ネットワーキング、組み込みシステムおよび 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、オートモーティブ、セキュリティ、ブロックチェーンなど、Linux Foundation が注力しているテクノロジーは多岐にわたります。

Linux Foundation イベント

これは **Linux Foundation** のイベントのごく一部です。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。

- 埋め込みソースサミット会議
- ノース・ネットワーク
オープン・ネットワーキングと
エッジ・サム・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カーネルメインテナーサミ
ット
- リナックス・ファウンデーショ
ン・メンバー・サミット

• オープン・コンプライアンス・
サミット

• 他にもたくさんある。

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、協力しています。**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管理者、システム管理者、オープンソース プログラム オフィスを率いる技術者、その他の重要なリーダーシップ機能にとって、最適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内での知名度を素早く獲得し、次世代のテクノロジーを評価・創造する人々とのつながりを形成することで、オープンソース開発作業を推進するための最良の場所です。また、知識を共有し、得るためのフォーラムを提供し、組織が将来の技術投資に情報を提供するためにソフトウェアのトレンドを早期に特定するのを支援し、雇用者と人材を結びつけ、世界中の影響力のあるオープンソースの専門家、メディア、アナリストに技術やサービスを紹介します。

1.2 Linux Foundation トレーニング

トレーニング会場

Linux Foundationは、いくつかの種類のトレーニングを提供している：

- 物理的な教室（多くの場合オンサイト）
- オンライン・バーチャル教室
- インターネットを利用した個人のペースに合わせたEラーニング
- イベントベース

Linux Foundation のトレーニングは、コミュニティのコミュニティによるコミュニティのためのものであり、開発者コミュニティのリーダーたちによる講師陣とコンテンツが特徴です。

受講者は、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に柔軟に対応でき、技術的に高度で、開発コミュニティの実際のリーダー自身によって作成されたトレーニングを受けることができます。**Linux Foundation** のコースは、受講者が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を提供します。**Linux Foundation** のクラスでは、オンラインまたは対面式のトレーニングにより、オープンソースの管理および開発の本質について、常に一步先を行くことができます。

グ

トレーニング

現在開講しているコースは以下の通り：

- 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システム管理コース
- 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>。

Linux Foundation はまた、**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る。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をカバーしています。

edXのウェブサイトでは、"Linux Foundation"で検索してください。

著作権

- 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 です。無断複写・転載を禁じます。



コピーや配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、複写、写真、磁氣的、電子的、またはその他の記録を含む (ただし必ずしもこれらに限定されない) いかなる方法によっても、検索システムに保存、送信、または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、一切の責任を負いません。

The Linux Foundation の資料が使用、複製、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。

グ

機密情報について



機密情報の非開示

「(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する情報。

このコースには機密情報は含まれません。また、機密情報を授業中に漏らしてはなりません。

1.3 Linux Foundation認定資格

Linux Foundation認定試験

- **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は、<https://training.linuxfoundation.org/certification>でご覧いただけます。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**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ますので、[https](https://training.linuxfoundation.org/certification)を参照してく

ださい：

現在のリストは、[//training.linuxfoundation.org/certification](https://training.linuxfoundation.org/certification)。

- コースの説明、技術的要件、その他のロジスティックスなどの詳細については、<https://training.linuxfoundation.org>。

認定/トレーニング ファイアウォール

- **Linux Foundation**には、2つのトレーニング部門があります：
 - 認証
 - コース・デリバリー
- これらは**ファイアウォール**で隔てられている：
 - 第三者機関が**LF**認定準備クラスを開発・提供できるようにする。
 - **LF**コースでの**秘伝ソース**の使用を防ぐ
 - **LF**コースで**テストを教えること**を防ぐ
- インストラクターは（今日も含めて）、一般に公開されているインフォメーションに全面的に導かれている。

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、採点に直接関与することはありません。

この自らに課した**ファイアウォール**を実施することで、独立した組織や企業は、受験者が認定試験に合格することを目的としたサードパーティのトレーニング教材を開発できるようになる。

さらに、成功するために知っておかなければならない秘密の"ヒント"（あるいは一般的な秘密）がないことも保証する。

また、**Linux Foundation** は、**テストを教えるだけ**でなく、**Linux** システム管理で成功するために習得が必要とされる多くの分野の幅広い知識を受講者に提供する、非常に堅牢な一連のコースを開発することができます。

準備リソース

- 何かする前に、<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

- ドメインとコンピテンシー
- 無料トレーニング・リソース
- 有料トレーニング・リソース
- 試験を受ける

- また、**候補者ハンドブック**は以下から入手できる：

<https://training.linuxfoundation.org/go/candidate> ハンドブ

ック

- 試験に関する情報は、

<https://training.linuxfoundation.org/certification/lfcs>

もご参照ください。

私たちは、上で引用した文書にある問題の一部（すべてではない！）について説明する。これらの文書はすべて、**Linux Foundation**のウェブページ（<https://training.linuxfoundation.org/certification>）のリンクからも簡単に見つけることができる。

トピックは以下の通り：

- 試験で扱われる内容
- 本人確認、認証、資格、アクセシビリティ、守秘義務などの候補者要件。

- 受験登録、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チェックする。
- 試験の開始と終了方法
- 試験のインターフェースと形式
- 試験結果、採点および再採点のリクエスト、再受験ポリシー
- 証明書の発行、検証、有効期限、更新など
- 技術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・バッジで能力と資格を伝える
- 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。
- 資格とその取得プロセスを説明する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
コースと資格
- 詳細は<https://training.linuxfoundation.org/badges/>

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お約束します。私たちは、あなた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。そのため、**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、あなたの資格とその取得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、履歴書への埋め込み、Eメール署名、ウェブなど、あらゆるデジタルプラットフォームで共有できます。
- 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と、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に発行されます。
- また、試験やコースの開発チームに貢献した人にもバッジが発行されます。

仕組み

1. パートナーであるCredlyの**Acclaim**プラットフォームウェブサイトでバッジを請求してください。
2. そのメールのリンクをクリックする。
3. **Acclaim**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。
4. バッジを受け取ってください。
5. 共有を開始する。

1.5 実験演習、解答とリソース

ラボ

- 各セッションの最後には実践的なエクササイズが用意されている。
- 仮想マシンでもベアメタルでも可能
 - 特に指定がない限り
- オプションとしてマークされたエクササイズ
- 解答は各問題の最後にあります。

コースの解決策とリソースの入手

- 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 <https://training.linuxfoundation.org/cm/LFS307> からダウンロードできます。

- ブラウザをお持ちでない場合は、こちらをご利用ください:

```
$ wget --user=LFtraining --password=<ask instructor> \
  https://training.linuxfoundation.org/cm/LFS307/LFS307 V1.4 SOLUTIONS.tar.xz
RESOURCESファイルも同様である。
```

```
$ wget --user=LFtraining --password=<ask instructor> \
  https://training.linuxfoundation.org/cm/LFS307/LFS307 V1.4 RESOURCES.tar
```

- 正誤表や最新の解答なども、このサイトに掲載される。
- これらのファイルは

```
tar xvf LFS307 V1.4 SOLUTIONS.tar.xz
tar xvf LFS307 V1.4 RESOURCES.tar
```

SOLUTIONSとRESOURCESの両ディレクトリには、s 01、s 10、s 13のようなサブディレクトリがあります。今後のセクションで、これらのファイルを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用のRESOURCESファイルがない場合があります。様々なラボのソースtarボール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リソースが提供されます。

コースにデモビデオが含まれている場合、それらはほとんど講義時間外の補助的な学習を目的としています。しかし、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、あるいは代替Linuxディストリビューションでの実践例を示すために、インストラクションはそれらのいくつかを見せることを選択するかもしれません。

1.6 Linuxとオープンソースプロジェクトの変化

オープンソースソフトウェアは頻繁に変わる

- **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、非推奨の機能は消えていくだろう。
- 私たちは問題を最小限に抑えようと努めているが、自分たちでは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い。
- その代わり、重要な特徴を欠いた陳腐な素材を持つことになる。
- こうしたソースやターゲットの移り変わりに対応するのも、オープンソース・ソフトウェアを扱う楽しみのひとつだ！

Linux Foundation のコースは常に更新されており、中には、以下のように、依存するプロジェクトの新しいバージョンがリリースされるたび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は常に進化している。技術レベル（カーネル機能を含む）でも、ディストリビューションやインターフェースのレベルでも。

ですから、このクラスがリリースされた時点では、できる限り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ご留意ください。それは避けられません。

その結果、どんなに努力しても、次のような問題が生じる可能性がある：

- 私たちがサポートしている**Linux**ディストリビューションで動作しなくなる。
- **API**が変更されると、特定のカーネルやアップストリームライブラリや製品を使った実験実習ができなくなる。

- 私たちがサポートしてきた非推奨の機能は、ついに姿を消すことになる。

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る。私たちはそれを達成しようと努力していますが、わかっています：

- 非現実的な期待だ。
- 問題点を見つけ出し、それを解決する方法を考えることは、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で働く「冒険」の一部なのだ。動く標的は、常に当てるのが難しい。

このような問題をすべて避けようとするれば、教材はすぐに古くなり、変化についていけなくなります。私たちは、このような問題に気づき、次のリリースで適切な改善策を取り入れることができるよう、コースウェアの設計者に知らせてくれる講師や学生を頼りにしています。

これが真のオープンソースのやり方だ。

1.7 EラーニングコースLF207

コンパニオンEラーニングコースLF207

- Linuxファウンデーションは、自習型のeラーニング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

- このコースは、このコースと同じ内容の多くをカバーし、実験実習の多くを共有する。
- オンライン教材へのアクセスは1年間有効で、すべてのコンテンツと練習問題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。
- インストラクターが指導するこのバージョンでは、時間の制約上、すべてのテーマに広さでも深さでも十分な時間を割くことができない。

- このコースの受講の一部として、**LF207**の定期購読を受け取られたかもしれません。そうでない場合は、
<https://training.linuxfoundation.org/>。

1.8 ディストリビューション詳細

ソフトウェア環境

- 授業で使用する教材は、さまざまな環境を想定して作られている
- 現在の3つの主要なLinuxディストリビューションファミリーに焦点を当てています：

- レッドハット / フェドラ

- OpenSUSE / SUSE

- デビアン

ご注意

このセクションの教材は、**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必要とするか、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、どのような環境でも実行できるものもあります。しかし、そのような場合、この資料を読み飛ばすことはできますが、それでも、その情報を吸収することをお勧めします。

Linux Foundation が作成する資料は、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、技術的な説明やラボ、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（特定のシナリオのために言及することはあります）。

実際には、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。受講生が使用するディストリビューションは、これら3つのうちの1つ、またはそれらから派生した製品になる傾向があります。

詳細

どのディストリビューションを選ぶか

- 考慮すべき要素がいくつかある：
 - あなたの雇用主はすでに標準化を行っていますか？
 - もっと知りたい？
 - 認証しますか？
- 複数のディストリビューションで実験することをお勧めします。

新しいディストリビューションを選ぶ際には、いくつかの質問を自分に投げかけるべきです。1つの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を比較しなければなら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、ファイルの場所であることにすぐに気づくでしょう。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に乗り換えるのは比較的簡単です。

ツールやユーティリティの中には、ベンダーが提供するフロントエンドを持つものもある。本文に含まれる手順は、異なるプラットフォーム上で実行するために変更する必要があるかもしれない。

レッドハット / フェドラ・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
- RPMベースで、**dnf**（または**yum**）を使っ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い。
- CentOSと**Oracle Linux**のアップストリーム
- **CentOS ストリーム**用ダウンストリーム

CentOS Streamは無償で利用できるため、デモやラボに使用されている。

Fedoraは、**Red Hat Enterprise Linux**、**CentOS**、**Oracle Linux**の基盤となるコミュニティ・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1つは、**Fedora**の構築に多様なコミュニティが関わっていることです。

Fedoraコミュニティは、半年に1度ほど新しいバージョンをリリースしています。このため、**Red Hat / Fedora**の部分は、リリースサイクルがより長い最新バージョンの**CentOS / CentOS Stream**に統一することにしました。一度インストールすれば、**CentOS Stream**は、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)**に非常に近くなります。



詳細

CentOSとCentOS Stream

- **CentOS**は歴史的に、基本的に**RHEL**のコピーであり、アップデート後に若干の時間遅れがある。
- **CentOS Stream**は**RHEL**より先にアップデートを受けるが、それ以外は**RHEL**にかなり近い。そのため、新しい機能はより早く吸収されるだろう。
- **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、このコースは現在**CentOS Stream**ディストリビューションでテストされています。**CentOS 8**や**RHEL 8**との違いは軽微で、気にならないはずです。

OpenSUSEファミリー

- 現在の資料は**OpenSUSE**の最新リリースに基づいており、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は**zypper**を使う。
- 管理用に**YaST**が利用可能



• ~~x86およびx86_64~~

- **SUSE Linux Enterprise Server (SLES)**のアップストリーム

OpenSUSEは無償で利用できるため、デモやラボに使用されています

。

OpenSUSEと**SUSE Linux Enterprise Server**の関係は、先ほど説明したFedoraと**Red Hat Enterprise Linux**の關係に似ています。ただし、この場合、**SUSE Linux Enterprise Server**の無償バージョンを入手するのが困難なため、**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**OpenSUSE**を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**に関する資料は通常、**SUSE Linux Enterprise Server**にも問題なく適用できます。

Debian ファミリー

- サーバーでもデスクトップでもよく使われる
- **DPKG**ベースで、インストールとアップデートにaptとフロントエンドを使用する。
- **Ubuntu**、**Linux Mint**、その他のアップストリーム
- 現在の資料は**Ubuntu**の最新リリースに基づいており、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**

- 長期リリース (LTS)

Ubuntuは、**Debian**と同様に無償で入手できるため、デモやラボに使用されているが、**Linux**の新規ユーザーの裾野が広い。

Debianディストリビューションは、**Ubuntu**や**Linux Mint**など、他のいくつかのディストリビューションのアップストリームです。**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焦点を当てています。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指しています。**Ubuntu**はほとんどのパッケージを**Debian**のunstableブランチから取得しているため、**Ubuntu**は非常に大規模なソフトウェアリポジトリにもアクセスできます。このような理由から、私たちは**Ubuntu**を**Debian**ベースのリファレンス・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。

新たな流通の類似性

- 現在の傾向と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る。

- **systemd**、システムのスタートアップとサービス管理

- **journald**、システムログの管理

- **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

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、講義とラボの情報はほとんどこれらに基づいている。

systemd は、**SysVinit** や **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る。もしあなたが選んだディストリビューションやリソースがこれらのコマンドをサポートしていない場合は、適宜翻訳してください。

journald はロギングデータを収集・保存する **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、インデックス化されたジャーナルを作成し、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ースの信頼レベルを定義する 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をサポートしています。これは **iptables** 設定を置き換えるものです。

ip プログラムは **net-tools** パッケージの一部で、**ifconfig** コマンドの代わりとなるように設計されている。**ip** 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、トンネルの表示や操作を行います。

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

古いコマンドを **systemd** の対応するコマンドに翻訳する際に、以下のドキュメントが役に立つかもしれません。

https://en.opensuse.org/openSUSE:Cheat_Sheet_13.1#サービス

AWS 無料ティア

- **アマゾン ウェブ サービス (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る。
- 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・コースの演習を行い、**Linux** の経験を積むために必要なすべてかもしれない。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してくれるかもしれません。

<https://training.linuxfoundation.org/cm/prep/aws.pdf> から、**AWS** のフリー・ティアを試すのに役立つ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を提供せず、グラフィカル・インターフェースも提供しないので、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にはいくつかの特別な課題がある。

1.9 ラボ

練習1.1: sudoのためのシステム設定

絶対に必要な場合を除き、**rootシェル**を実行するのは非常に危険である。たった一つのタイプミスやその他の間違いが、深刻な（致命的な）損害を引き起こす可能性がある。

したがって、賢明な手順としては、**sudo**メカニズムを使って、単一のコマンドをスーパーユーザー権限で実行できるように設定することである。**sudo**を使えば、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のようなディストリビューションを使用している場合、コースのために**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。しかし、それでも手順を理解しておくべきです。

使用しているユーザーアカウントで**sudo**が実行できるようにシステムが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だ：

```
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。代わりにエラーメッセージが表示された場合は、以下の手順を実行する必要があります。

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、通常、**root**が**sudo**アクセス権を付与したいユーザーの名前でファイルを作成する必要がある。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ので、この規約は実際には必要ない。ファイルには単に

```
学生 ALL=(ALL) ALL
```

ユーザが学生の場合

古くから行われている方法（今でも確かに有効）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ある。**visudo**プログラムを使うのがベストで、正しい構文で編集できるように注意してくれる。

また、次のように入力して、ファイルに適切なパーミッションを設定する必要もあるだろう：

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

（**Linux**ディストリビューションによっては、パーミッションに440ではなく400を要求する場合があることに注意）。

これらの手順を実行した後、**exit**と入力してrootシェルを終了し、もう一度**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、特定のユーザーに対して特定のパーミッションのみを指定したり、検索されるパスを制

限したりなど、他にもたくさんある。[etc/sudoers](#)ファイルは非常によく自己文書化されている。

しかし、たとえシステムにすでに**sudo**が設定されていたとしても、もう1つぜひやっていただきたい設定がある。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では、実行可能ファイルを見つけるパスが異なります。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rootユーザーではなく、ユーザーのPATHを継承するからです。

したがって、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なければならないだろう。セキュリティの向上は、余分な入力や、これらのプログラムがどのディレクトリにあるかを調べることに見合うものではないだろう。従って、ホームディレクトリの[.bashrc](#)ファイルに以下の行を追加することをお勧めする：

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

一旦ログアウトしてから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	データの区別	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	/etc	36
2.9	/ホーム	37
2.10	/libおよび/lib64	39
2.11	/メディア	40
2.12	/mnt	41
2.13	/オプト	42
2.14	/プログラム	43
2.15	/シス	45
2.16	/ルート	46
2.17	/sbin	47
2.18	/srv	49
2.19	/tmp	50
2.20	/usr	51
2.21	/var	52
2.22	/ラン	53
2.23	研究室	54

学習目標

このセッションを終えるまでに、あなたは次のことができるようになるはずだ:

- Linuxがなぜ1つの大きなファイルシステム・ツリーを構成する必要があるのか、また、それをどのように行うかについて、どのような主要な考慮事項があるのかを説明してください。
- ファイルシステム・ヒエラルキー・スタンダードが果たす役割を説明しなさい。(FHS)
- ブート時にルート (/) ディレクトリで利用可能でなければならないものと、システム起動後にのみ利用可能なものについて説明する。
- 主なサブディレクトリのツリーをそれぞれ調べ、その目的を説明し、内容を調べる。

2.1 ひとつの大きなファイルシステム

ひとつの大きなファイルシステム

- ひとつの大きなファイルシステムとして見られる
- ルートディレクトリを頂点とする倒立ツリー
- 追加のパーティションとファイルシステムはサブディレクトリにマウントできる。
- Linuxディストリビューションは、標準的な処方箋に従っている。

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る。実際には、ルート・ディレクトリである/がツリーの一番上にある倒立ツリーとして図示されるのが普通である。

この1つの大きな論理ファイルシステムの中に、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、さらには多数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、ネットワーク上にあるものを含め、いくつものデバイス上にある可能性があります。

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上にあるかなんてまったく気にしない。

むかしは、さまざまなUNIXライクなオペレーティング・システムが、この1つの大きなツリーをさまざまな方法で構成していた。そのため、複数のシステム上でアプリケーションを書いたり、システム管理タスクをこなしたりするのは難しく、しばしばイライラさせられた。

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しようと努力してきた。

2.2 データの区別

データの区別

- 共有可能か非共有可能か
- 可変 vs 静的

ファイルやデータが1つの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種類の区別がある：

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できるもの。非共有データとは、特定のホストに固有でなければならないものです。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. バリアブル対スタティック

静的データには、バイナリ、ライブラリ、ドキュメントなど、システム管理者の支援なしには変更されないものが含まれます。可変データとは、システム管理者の助けがなくても変更される可能性があるものです。

多くの場合、こうした論理的な区別は、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。

2.3 FHS Linux標準ディレクトリツリー

ファイルシステム階層標準

- ファイルの種類と場所の標準レイアウト。
 - ファイルの場所を予測可能にする
- BSDなどの歴史的な標準から発展した。
- 管理はLinux Foundationが行っています。

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

。

Filesystem Hierarchy Standard (FHS) は、もともとはFree Standards Groupによって管理され、現在はLinux Foundationによって管理されている。https://refspecs.linuxfoundation.org/FHS_3.0/fhs-3.0.pdfからダウンロードできます。

標準的なレイアウトを指定することで、FHSはファイルの場所の予測を簡単にする。ほとんどのLinuxディストリビューションはFHSは、おそらくどのディストリビューションも正確には追従しておらず、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ります。

FHS Linux標準ディレクトリツリー

ファイルシステム階層全体のプライマリディレクトリ。

/bin

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可能プログラム。

/boot

カーネル、initrdまたはinitramfsイメージ、ブートファイルなど、システム起動に必要なファイル。

表2.1: 主なディレクトリ

ディレクトリの目

設定ファイルとブートローダプログラム。

/dev

ハードウェアおよびソフトウェア・デバイスとのやりとりに使用されるデバイス・ノード。

/etc

システム全体の設定ファイル。

/home

個人設定、ファイルなどを含むユーザーホームディレクトリ。

/ライブラリ

binと/sbinにある実行バイナリに必要なライブラリ。

/lib64

64-bit /binおよび/sbinにある実行バイナリに必要なライブラリ。
32ビットと64ビットの両方のプログラム。

/メディア

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。

ルートディレクトリの下には、さらにディストリビューション固有のディレクトリがあるかもしれない。雑多なデータに使われる/miscや、tftpを使ったブートに使われる/tftpbootなどである。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。他のディレクトリーを持つことはFHSに違反しないが、標準が指示するディレクトリー以外のディレクトリー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すること。

監督	目的
/オプション	オプションのアプリケーション・ソフトウェア・パッケージ。
/proc	仮想的な擬似ファイルシステムで、システムとその上で実行されているプロセスに関する情報を提供する。
/走る	ランタイム変数データ。システムが起動したときからの情報を含む。 古い/var/runを置き換える。
/シス	仮想的な擬似ファイルシステムで、システムとその上で実行されているプロセスに関する情報を提供する。システムパラメーターの変更に使用できる。デバイスツリーに似ており、統合ファイルシステムの一部である。
/ルート	ルートユーザーのホームディレクトリ。
/sbinの	必須システムバイナリ。
/srv	システムが提供するサイト固有のデータ。めったに使用されない。
/tmp	一時ファイル。多くのディストリビューションでは、再起動時に失われ、メモリ内のラムディスクになることがある。
/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。
/var	システム運用中に変化する可変データ。

FHS Linux標準ディレクトリツリー（続き）

表2.2: 主なディレクトリとその目的

これらは、メインのルート (/) ディレクトリから外れた追加ディレクトリである。

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。

- ブーツ
- リストア
- 回復する
- 修理

必要に応じてシステム



- アプリケーションやパッケージは、ルート・ディレクトリの新しいサブディレクトリを作成してはならない。

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について話しているのではない。

ファイルシステム全体を1つの大きなツリーと見なすこともできるが、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る。

ルートディレクトリそのものが含まれるパーティションとファイルシステムはかなり特殊で、多くの場合、特別な専用パーティションにあり、/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。

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重要なファイルが含まれていなければなりません。そのため、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。ルートパーティションは、次のような要件を満たす必要があります：

- システムを起動する。

- 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。
- 経験豊富なメンテナンス担当者は、損傷したシステムを診断し、再構築するためのツールを持っていないければならない。

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に新しいサブディレクトリを作ってはならない。



/ は /root ではない

/rootはスーパーユーザーのホーム・ディレクトリである。ここでの言葉の選択は混乱を招くかもしれない!

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、例えばシングルユーザーやリカバリーモードで起動する場合など、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に必要となる。
- また、スクリプトによって間接的に使用される実行可能ファイルを含むこともある。
- サブディレクトリを含んではならない。

最近のいくつかのディストリビューションは、`/bin`と`/usr/bin`（`/sbin`と`/usr/sbin`も同様）を分離する戦略を放棄し、シンボリックリンクで1つのディレクトリを持つだけにし、2つのディレクトリのビューを維持している。彼らは、`/usr`を別パーティションに置き、ブート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：

`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`、`uname`。

[また、オプションとして「テスト」を含めることもできる：

`csh`、`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/`に置く。非 root ユーザーのみが必要とするプログラムは、このカテゴリに置かれます。

/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        hclconfig    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          mv            pidof          systemd      zcat
bzdiff        dumpkeys     kmod          nano          ping           systemd-ask-password zcmp
bzegrep       echo         less          nc            ping4          systemd-escape zdiff
bzeze        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     nlsdomainname readlink        systemd-notify zgrep
bzip2recover false         ln            ntfs-3g       red            systemd-sysusers zless
bzless        fgconsole    loadkeys     ntfs-3g.probe rm             systemd-tmpfiles zmore
bzmore        fgrep        login         ntfscluster  rmdir          systemd-tty-ask-password-agent znew
cat           findmnt      loginctl     ntfscluster  rnano          tar
```

図2.1: /binディレクトリ: Ubuntu 18.04

2.5. /bin

33

RHEL、CentOS、Fedora、Ubuntuの最近のディストリビューション・バージョンは、/binと/usr/binをシンボリックリンクしているので、実際には同じである。

上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独立したディレクトリはなくなっている。

2.6 /ブート

ブート

- ブートプロセスに必要なものがすべて含まれている。絶対に必要なファイルは以下の2つ:
 - `vmlinuz`: 圧縮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。
- 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されるデータを格納する。
- また、情報とデバッグに使用される2つのファイルも含まれている:
 - `config` カーネルコンパイルの設定に使用する。
 - `System.map`: デバッグに使用されるカーネルシンボルテーブル
- ディストリビューションによっては他のファイルもあるかもしれない。

あるUbuntuシステムではこうなっている:

```
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 efi/
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・バージョンに依存する長い名前を持っている。また、`initramfs`の代わりに`initrd`（初期ラムディスク。）

2.7 /dev

- キャラクター と ブロッ
ク デバイス・ノード、
- 適切なシステム運用に不可欠
消去
- 最近のシステムは、**udev**を介し
てデバイスファイルを自動的に
- 管理するシステム（または組み込み
デバイス）では、インストール
時に **MAKEDEV** または **mknod**

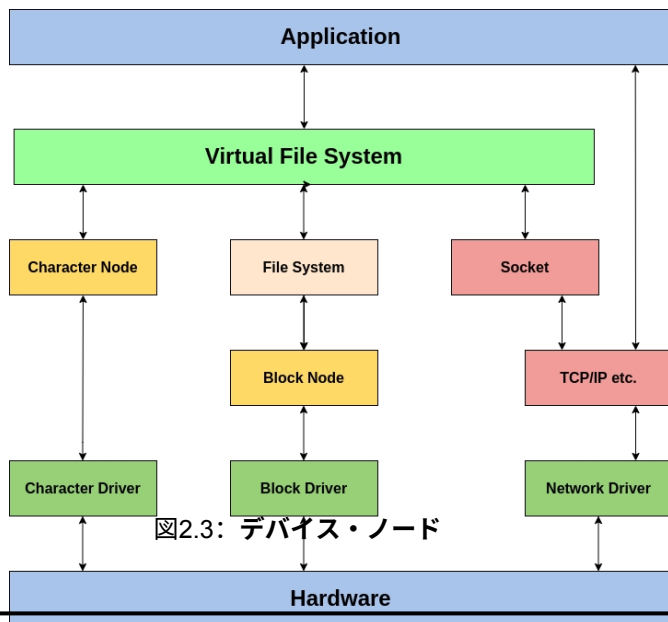


図2.3: デバイス・ノード

で作成できます。

または必要に応じていつでも

このディレクトリには、システムに組み込まれた、またはシステムに接続された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。これらの特別なファイルは、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、**キャラクタ**（バイトストリーム）と**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などの名前で参照されます。

```

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-rw-rw- 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ノードだけを/devに作成する。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だろう。

2.8 /etc

/etc

- マシンにローカルなコンフィギュレーション・ファイル（およびいくつかのスタートアップ・スクリプト）を含む。
- このディレクトリにバイナリを置くことはできない
- ディストリビューションは、しばしば特定のファイルやディレクトリを追加する。例えば、**Red Hat**は/etc/sysconfigを含む多くのディレクトリを追加します。

このディレクトリには、マシンローカルのコンフィギュレーション・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ており、実行可能なバイナリ・プログラムは含まれていないはずである。

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には、以下のようなものがある：

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、mtools.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われているmtools.confのように、今日ではほとんど関係のないものもある。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る。

ディストリビューションは、/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、/etc/sysconfig を含む他の多くのディレクトリを追加しており、そこには多くのシステム設定ファイルやディレクトリがあります。

その他の重要なサブディレクトリは以下の通り：

- `/etc/skel`: 新しく作成されたホームディレクトリにデータを入れるためのスケルトンファイル。
- `/etc/systemd`: **systemd** を使っているときにシステムサービスを起動、停止、制御するための設定スクリプトが含まれているか、そのスクリプトを指す。
- `/etc/init.d`: **System V**初期化時のスタートアップとシャットダウンのスクリプト。

2.9 /ホーム

/ホーム

- 通常、ユーザーのホームディレクトリが含まれる
- 別の名前がついているかもしれない
- ユーザーグループ用のサブディレクトリを含むことができる

Linuxシステムでは、ユーザー・ディレクトリは、`/home/coop`、`/home/student`などのように、慣例的に`/home`の下に置かれる。すべての個人設定、データ、実行可能プログラムは、このディレクトリ階層に置かれる。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系オペレーティング・システムでは、`/home`ディレクトリ・ツリー概念は存在するが、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、自動マウント機能によって最終的に`/home`にマウントされる。これは、通常、ホーム・ディレクトリは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、ホーム・ディレクトリは使用時に自動的にマウントされるからである。

Linuxにもこれと同じ自動マウント機能があるが、多くのユーザーはそれを意識していないし、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だろう。

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ことができる：

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

1つ例外がある。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`の下にある。いくつかの古いUNIXシステムでは、`/`を代わりに使うことがあるが、これは乱雑になる可能性がある。

/ホーム 例

```
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-r----- 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-r-----  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-r-----  1 student student  51 May 26 08:05 .xauthority
-rw-r-----  1 student student 3462 May 26 08:05 .xsession-errors
-rw-r-----  1 student student 3462 May 25 08:49 .xsession-errors.old
student@ubuntu:~$
```

図2.5: /homeディレクトリ

システムを何度アップグレードしたか、ホームディレクトリをコピーしたかによって、多種多様なファイルが存在する可能性がある:

- **XDГ** で指定されたディレクトリ (`~/Documents`、`~/Downloads`、`~/Pictures`、`~/Desktop` など) にあるメディアとデータ。
- ドットファイルや、`~/.config`や`~/.local`のような新しい場所にあるプログラム・ユーザー・データ。
- `.xsession-errors`のようなログファイルでさえも。

2.10 /lib と /lib64

/lib と /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含む。

/bin と /sbin

- ディストリビューションによっては、64ビットライブラリを /lib64 に置くものもある。
- 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 のようなディストリビューションに依存する場所に置く。

最近のディストリビューションの中には、/lib と /usr/lib （/lib64 と /usr/lib64 も同様）を分離する戦略を放棄し、シンボリックリンクで1つのディレクトリを持つことで、2つのディレクトリを表示するようにしているものもある。彼らは、/usr を別パーティションに置き、ブート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ます。

これらのディレクトリーには、/bin と /sbin にあるバイナリーを実行するために必要なライブラリーだけを入れるべきである。これらのライブラリは、システムを起動し、ルート・ファイルシステム内でコマンドを実行するために特に重要である。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 や /lib/x86_64-linux-gnu/security.

32ビットと64ビットの両方のバイナリをサポートするシステムでは、両方の種類のライブラリをシステム上に置いておく必要があります。ディストリビューションによっては、32ビット・ライブラリ用 (/lib) と64ビット・ライブラリ用 (

/lib64) に別々のディレクトリがあります。

`ls -l /lib*`

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


2.11 /メディア

/メディア

- 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ドライブ
- ダイナミックマウント可能



/メディア vs /ラン

最近のシステムでは、`/media`の代わりに`/run/media/[username]/...`を使うことがある。

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われた。これには**CD**、**DVD**、**USBドライブ**、さらには旧石器時代のフロッピーディスクも含まれる。

Linuxシステムは、このようなメディアを挿入すると動的にマウントし、**udev**はディレクトリを作成し、コンフィギュレーション・ファイルで指定された**udev**ルールで設定された名前で、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われたディレクトリは消えます。

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、メディアがマウントされ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に取って代わった。

注意：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
/メディア。**run**については後で説明する。

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、SAMBAなど）
 - 一時パーティションまたは論理ボリューム

`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る。よく使われるのは、以下のようなネットワークファイルシステムである：

- ネットワークファイルシステム
- サンバ
- CIFS
- 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われていた。

一般的に言って、このディレクトリはインストール・プログラムが使用すべきではない。現在使用されていない別の一時ディレクトリを使用した方がよい。

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]` ディレクトリになければなりません。

`ls -l /opt`

```
合計20K
drwxr-xr-x. 4 root root 4.0K Jul 26 2021 brother
drwxr-xr-x. 4 root root 4.0K 2019年9月1日 google
drwxr-xr-x   3 root root 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 May 30 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、全部または大半のファイルを1つの隔離された場所に保管したいソフトウェア・パッケージのために設計されている。

例えば、**dolphy app**が`/opt`以下に存在するパッケージの名前であった場合、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するはずであり、これにはバイナリ用の`/opt/dolphy_app/bin`とマニュアルページ用の`/opt/dolphy_app/man`が含まれる。

これによって、ソフトウェアのインストールとアンインストールが比較的容易になる。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のも簡単になります。

しかし、後述するRPMやAPTのようなパッケージングシステムを使えば、このような反社会的な振る舞いをすることなく、ファイルマニフェストやロケーションを明確に意識してインストールやアンインストールを行うことも容易であることに注意されたい。

Linuxでは、 `/opt`ディレクトリは、プロプライエタリ・ソフトウェアを持つアプリケーション・プロバイダや、ディストリビューションのばらつきによる複雑さを避けたいアプリケーション・プロバイダによってよく使われる。たとえば、あるシステムでは、パッケージが `/opt/brother`、`/opt/zoom`、および `/opt/google`、後者には `chrome` と `earth` のサブディレクトリがある。

`opt/bin`、`/opt/doc`、`/opt/include`、`/opt/info`、`/opt/lib`、`/opt/man` ディレクトリは、ローカルのシステム管理者が使用するために予約されています。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プログラムがなくても機能できなければなりません。ほとんどのシステムでは、これらのディレクトリーにはアクセスしません。

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ない
- カーネル・データ構造へのインターフェース
- 各アクティブ・プロセスは、このディレクトリにサブディレクトリを持つ。

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、`/proc`ディレクトリは、実行されていないシステムでは空である。

カーネルは、いくつかの重要なデータ構造を**/proc**エントリを通して公開している。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っている。

`proc`内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る。ほとんどの場合、サイズは0バイトと表示されているが、見てみると大量の情報が含まれていることがある。

加えて、仮想ファイルの日時設定のほとんどは、現在の日時を反映しており、常に変化していることを示している。実際、これらのファイルの情報は、閲覧されたときにのみ取得され、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の最新情報を垣間見せてくれる。

その他、`/proc/filesystems`や**`/proc/sys/`**ディレクトリは、システム構成情報とインターフェースを提供する。

整理する目的で、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 **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`内のエントリを調べます。カーネル・コンフィギュレーションとシステム・モニタリングに関する今後の章では、`/proc`内のエントリをさらに詳しく見ていきます。

/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 vmstat
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

以下は、あるプロセスに属する `/proc` ディレクトリの内容である。

```
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 smaps 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ひとつ、`/proc/interrupts`。

```
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: 17900 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: 84 149 20 IR-PCI-MSI 520192-edge enp0s31f6
127: 225390 29 383 46 IR-PCI-MSI 2097152-edge iwlwifi
NMI: 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 74809 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 /シス

シス

- もうひと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ていない
- 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ーの検査や変更に変更可能
- **proc**より制御が良い

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ある。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ある。

sysfsは、システムに関する情報を収集し、実行中の動作を変更するために使われる。その意味ではしかし、**/proc**より歴史が浅く、どのようなエントリーを含むことができるかについて、厳格な基準を守っている。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ない。

procと同様に、このコース全体を通して**/sys**のエントリーを調べ、カーネル・コンフィギュレーションとシステム・モニタリングに関する今後の章でも関連してくる。

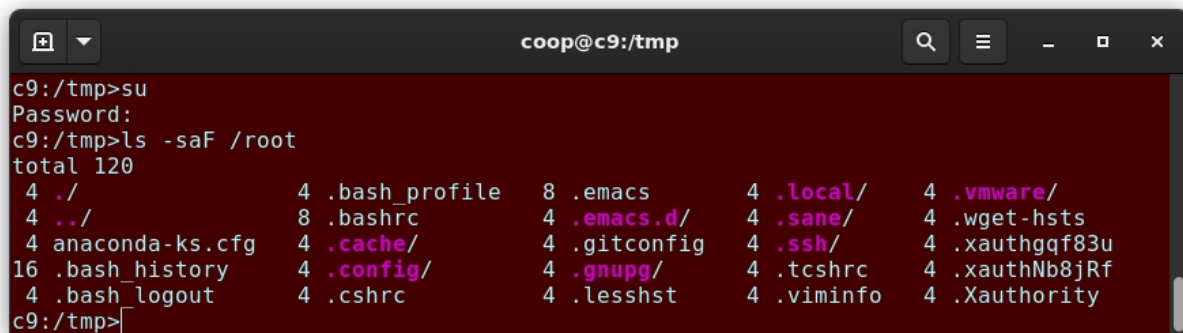
```
File Edit View Search Terminal Help
c7:/tmp>ls -lF /sys
total 0
drwxr-xr-x  2 root root 0 May 26 12:08 block/
drwxr-xr-x 23 root root 0 May 26 12:08 bus/
drwxr-xr-x 41 root root 0 May 26 12:08 class/
drwxr-xr-x  4 root root 0 May 26 12:08 dev/
drwxr-xr-x 23 root root 0 May 26 12:08 devices/
drwxr-xr-x  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 /ルート

/ルート

- ルート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（「スラッシュ・ルート」と発音する）は、ルート・ユーザーのホーム・ディレクトリである。

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べきである。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の

/sbinの

- 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
- 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ーが使用する。
- ブートと `/usr` のマウントに使用されるこのディレクトリの実行可能ファイル (`/bin` 内の実行可能ファイルに加えて)。
- このディレクトリにある実行可能ファイルは、システム回復操作の実行に使用されます。

前述したように、最近の分配金の中には、「分離」という戦略を放棄したものもある。

`/sbin`と`/usr/sbin` (`/bin`と`/usr/bin`も同様)、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つのディレクトリを表示することができる。

このディレクトリには、ブート、リストア、リカバリ、および/または修復に不可欠なバイナリが含まれています。

`/bin`ディレクトリにマウントする必要がある。また、ブート時にルートファイルシステムが正常であることがわかれば、必要に応じて`/usr`、`/home`、その他の場所にある他のファイルシステムをマウントできなければならない。

以下のプログラムは、このディレクトリに含まれている必要がある（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-rw-rw- 1 root 8 Apr 23 08:23 /sbin -> usr/sbin
```

/sbinの例

```

student@ubuntu:~$ ls /sbin
acpi_available  fsck.minix      lvmxchange      ntfsundelete    stop
agetty          fsck.msosdos    lvm              on_ac_power     sulogin
alsa            fsck.nfs        lvmchange       osd_login       swaplabel
apm_available  fsck.vfat       lvmconf         pam_extrusers_chkpwd  swapoff
apparmor_parser  fsck.xfs        lvmconfig       pam_extrusers_update  swapon
badblocks       fsfreeze        lvmdiskscan     pam_tally       switch_root
blkdiscard      fstab-decode    lvmdump         pam_tally2      sysctl
blkid           fstrim          lvmetad         partprobe       tc
blockdev        gdisk           lvmopolld       pccardctl       telinit
brctl           getcap          lvmsadc         pivot_root      tftp
bridge          getpcaps        lvmsar          plipconfig      tune2fs
brltty          getty           lvreduce        plymouthd       u-d-c-print-pci-ids
brltty-setup    halt            lvremove        poweroff         udevadm
capsh           hdparm          lvrename        pvchange         umount.nfs
cfdisk          hwclock         lvresize        pvcreate         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        mli-tool        pvresize         upstart-event-bridge
cryptdisks_start  insmod         mkdosfs         pvs             upstart-file-bridge
cryptdisks_stop  installkernel  mke2fs          pvscan          upstart-local-bridge
cryptsetup       ip              mkfs             quotaoff         upstart-socket-bridge
cryptsetup-reencrypt  ip             mkfs.bfs        quotaon         upstart-udev-bridge
ctrlaltdel      ipset           mkfs.cramfs     rar             veritysetup
debugfs          ipset           mkfs.ext2       raw             vgcfgbackup
depmod           ipset           mkfs.ext3       reboot          vgcfgrestore
dhclient         ipset           mkfs.ext4       regdbdump       vgchange
dhclient-script ipset           mkfs.ext4dev    reload          vgck
dmeventd         iptables        mkfs.fat        request-key     vgconvert
dmsetup          iptables        mkfs.minix      resize2fs       vgcreate
dosfsck          iptables        mkfs.msosdos    resolvconf      vgdisplay
dumpe2fs         iptables        mkfs.ntfs       restart         vgexport
e2fsck           iptables        mkfs.vfat       rmmod           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・バージョンは、/sbinと/usr/sbinをシンボリックリンクしているため、実際には同じである。

上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独立したディレクトリはなくなっている。

2.18 /srv

/srv

- サイト固有の変数データ
- データ、スクリプト、サーバー用（http、ftpなど）
- 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によれば

/srvには、このシステムによって提供されるサイト固有のデータが含まれている。

これを指定する主な目的は、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ことであり、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）のために単一のツリーを必要とするサービスを合理的に配置できるようにすることである。

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

システム管理者（およびディストリビューション）によっては、/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ベースのLinuxディストリビューションでは、/srvはデフォルトで空になっている。

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はすべて削除されるものとする。
 - **Ubuntu**はシステム再起動のたびに/tmpを消去する！
- (**Fedora**のような)/tmpをメモリ上に擬似ファイルシステムとしてマウントするシステムには注意してください。
 - でストレージ上の本物のファイルシステムに変換する：
`sudo systemctl mask tmp.mount`
再起動

一時ファイルの保存に使われ、どのユーザーやアプリケーションからもアクセスできる。ただし、/tmpにあるファイルは、長期間の保存は期待できない：

- 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cron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もある。これはUbuntuのポリシーである。
- 最近のディストリビューションの中には、/tmp ディレクトリを 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使用する仮想ファイルシステムを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、すべての情報が失われます！

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らない。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害を与えたり、クラッシュさせたりすることが容易になるからである。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。（おそらく環境変数を指定して）他の場所に置くことが可能であっても、多くのユーザーはその設定方法を知りませんし、すべてのユーザーが/tmpにアクセスできます。

このポリシーは、**Feodra**のようなsystemdを使っているシステムでは、コマンドを実行することでキャンセルできる：

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

その後、システムを再起動する。

2.20 /usr

/usr

表2.3: /usr以下のディレクトリ

/usr/bin	非必須コマンドバイナリ
/usr/etc	必要でない設定ファイル（通常は空）
/usr/games	試合データ
/usr/include	アプリケーションのコンパイルに使用されるヘッダーファイル。
/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	第三階層（マシンローカルファイル用）
/usr/sbin	必須でないシステムバイナリ
/usr/share	読み取り専用のアーキテクチャ非依存ファイル
/usr/src	Linuxカーネルのソースコードとヘッダー
/usr/tmp	セカンダリー一時ディレクトリ

usrディレクトリはセカンダリ階層と考えることができ、システム起動に必要でないファイルに使われる。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越しに共有することができます。

ソフトウェア・パッケージは、/usr直下にサブディレクトリを作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ある。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

/usr/localディレクトリにはローカルバイナリなどが格納される。

**非常に重要**

最近のいくつかのディストリビューションは、`/bin`、`/sbin`、`/lib`、`/lib64`を`/usr`の下にあるそれらのパートから分離し、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。これらのディストリビューションは、`/usr`を別パーティションに置き、ブート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を含む：読み取り専用でマウントできない
 - 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ファイル

ディレクトリ	目的
<code>/var/ftp</code>	ftpサーバーベースに使用。 表2.4: /var以下のディレクトリ
<code>/var/lib</code>	プログラムの実行に伴って変更される永続的なデータ。
<code>/var/lock</code>	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。
<code>/var/log</code>	ログファイル。
<code>/var/mail</code>	ユーザーのメールボックス。
<code>/var/run</code>	前回のブート以降の実行中のシステムに関する情報。
<code>/var/spool</code>	印刷キューなど、 スプールされた タスクや処理待ちのタスク。
<code>/var/tmp</code>	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る。
<code>/var/www</code>	ウェブサイトの階層のルート。

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・ファイルが含まれる。セキュリティ上の理由から、`/var`を別個のファイルシステムとしてマウントするのがよい考え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ある。

`/var/log`は、ほとんどのログファイルが置かれている場所です。`/var/spool`は、メール、印刷、cronジョブのようなプロセスのローカルファイルが、アクションを待つ間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 /走る

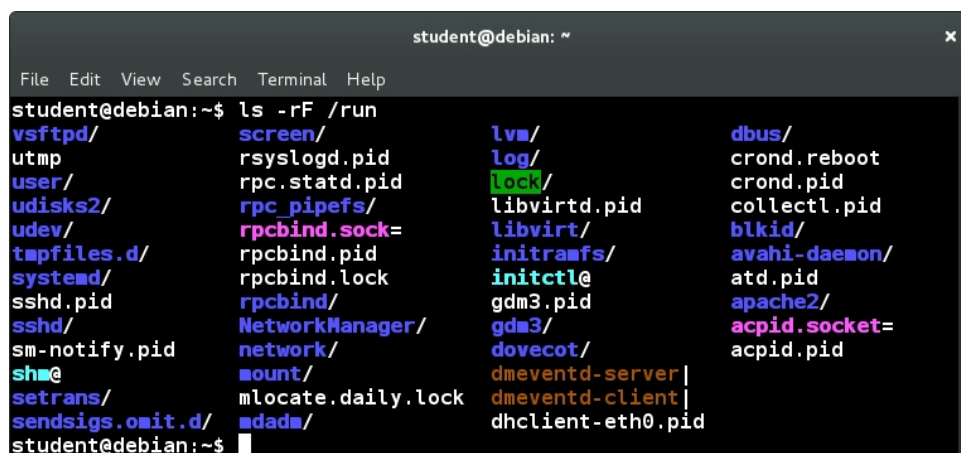
/走る

- メモリ上にのみ存在する疑似ファイルシステム
- 一時的なランタイムファイルに使用
- 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ある。ランタイム情報を含むファイルは、システム起動の初期に書き込む必要があり、再起動時には保存する必要がない。

一般的に`/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`ラムディスク（`/dev/shm`のようなもの）がマウントされる。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`のようないくつかの既存の場所は、`/run`配下のディレクトリへのシンボリックリンクになります。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`以下の場所を指すようになるかもしれません。



```

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
shm@         mount/       dmeventd-server|
setrans/     mlocate.daily.lock dmeventd-client|
sendsigs.omit.d/ mdadm/       dhclient-eth0.pid
student@debian:~$

```

図 2.13: `/run` ディレクトリ

2.23 ラボ



ビデオ・デモンストレーション・リソース

[using_linux_distros_demo.mp4](#)
[using_disk_usage_demo.mp4](#)

🔪練習2.1: デフォルトLinuxディレクトリのサイズ

duユーティリティを使用して、システムの各トップレベル・ディレクトリの全体のサイ

ズを計算する。コマンドを入力する:

`du --help`

を参照してください。

✅解決策 2.1

の下にあるディレクトリの全リストを、そのサイズとともに入手する:

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

```
4.3M   /ホーム
16K    /失われたもの
39M    /その他
4.0K   /srv
3.6M   /ルート
178M   /オプト
138M   /ブート
6.1G   /usr
1.1G   /var
16K    /mnt
4.0K   /メディア
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、ツリー内のその下にあるものを再帰的にすべて合計する。
- `-h`: 人間が読める数字 (KB、MB、GB) を指定する。
- `-p`: パーティションにないディレクトリは見ない。この場合は無視する:

`/dev /proc /run /sys`

なぜなら、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だからである。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ない:

`/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リンクしているだけだからだ。

演習2.2: procファイルシステムを見る



ご注意

この練習で表示される正確な内容はカーネルのバージョンに依存するので、表示される出力と正確に一致しないかもしれない。

2.23. ラボ

55

1. rootとして`/proc`に`cd`し、ディレクトリのリストを作成する。これにより、多くのファイルとディレクトリが表示されるはずである：

```
cd /proc
ls -lF
```

1/	128/	1510/	20/	2411/	30895/	53/	6925/	802/	951/	kmsg
10/	129/	1511/	2015/	2425/	31/	54/	7/	81/	952/	kpagecgroup
1002/	13/	1512/	2022/	2436/	31449/	55/	70/	813/	957/	ページ数
1007/	130/	1513/	2023/	2444/	32/	56/	702/	814/	97/	ページフラグ
10540/	131/	1514/	20300/	2451/	33/	58/	709/	816/	9742/	ロードアブジ
10590/	13172/	152/	20354/	2457/	34/	585/	71/	817/	98/	ロック
10798/	132/	15552/	20380/	2489/	35/	59/	718/	82/	99/	メモインフォ
10805/	133/	15663/	20388/	25/	36/	60/	719/	83/	9923/	ミス
10806/	134/	15737/	20392/	2503/	37/	61/	72/	834/	acpi/	モジュール
10809/	135/	159/	20396/	2504/	374/	6193/	721/	835/	アサウンド/	マウント
10810/	136/	15981/	2086/	2531/	379/	62/	723/	84/	バディイン	マートル
10813/	137/	16/	2090/	2546/	38/	63/	725/	841/	バス	ネット
10894/	138/	162/	211/	2549/	380/	634/	727/	842/	cグループ	ページタイ
10925/	1384/	1632/	22/	2562/	40/	64/	73/	85/	コマンドラ	パーティシ
10932/	1385/	1636/	2205/	25794/	41/	65/	7300/	857/	config.gz	ョン
10934/	1387/	166/	2209/	26/	42/	662/	74/	86/	コンソール	sched_debug
10935/	139/	1670/	2212/	2610/	43/	663/	757/	864/	CPU情報	scsi/
10941/	1390/	17/	2232/	26108/	44/	665/	758/	867/	暗号	セルフ
10983/	1393/	17271/	2238/	2619/	4435/	666/	76/	87/	デバイス	セルフ
10998/	14/	17361/	2296/	2624/	45/	67/	761/	88/	ディスク	スラビンフ
11/	140/	1793/	2298/	2627/	46/	670/	762/	881/	統計	オ
11047/	1410/	18/	23/	2644/	468/	671/	765/	886/	dma	ソフトアー
1105/	1415/	1831/	23042/	2645/	47/	673/	766/	887/	ドライバ	ク
1105/	1415/	1831/	23042/	2645/	47/	673/	766/	887/	エクゼクド	スタット
1105/	1415/	1831/	23042/	2645/	47/	673/	766/	887/	メイン	スワップ
1121/	1429/	18880/	2344/	2679/	470/	674/	768/	888/	fb	sys/
1123/	1437/	18903/	2348/	27/	484/	678/	769/	889/	ファイルシ	システム
1123/	1437/	18903/	2348/	27/	484/	678/	769/	889/	ステム	スレッドセ
1135/	1445/	19/	2353/	2706/	49/	679/	77/	89/	fs/	セルフ
1135/	1445/	19/	2353/	2706/	49/	679/	77/	89/	fs/	タイマーリ
11420/	146/	19392/	2354/	2762/	492/	68/	771/	9/	割り込み	スト
11420/	146/	19392/	2354/	2762/	492/	68/	771/	9/	割り込み	タイマー統
11499/	1463/	19488/	2365/	28/	493/	682/	78/	90/	アイオメモ	計
11515/	147/	1954/	23683/	2858/	5/	683/	79/	92/	アイオポ	tty/
11530/	1476/	1963/	2370/	28730/	50/	686/	793/	921/	ート	アップタイ
1163/	148/	19727/	2372/	28734/	51/	687/	794/	928/	イラク	ム
1163/	148/	19727/	2372/	28734/	51/	687/	794/	928/	カルスィム	バージョン
1164/	1485/	19734/	2374/	29/	510/	69/	8/	930/	ス	vmallocinfo
1164/	1485/	19734/	2374/	29/	510/	69/	8/	930/	コア	ブイエムスタ
12/	149/	19984/	24/	2973/	514/	690/	80/	931/	キー	ット
12/	149/	19984/	24/	2973/	514/	690/	80/	931/	キー	ゾーンイン
12/	149/	19984/	24/	2973/	514/	690/	80/	931/	キー	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ほしい。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**である。後で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ーを調べたり変更したりすることができる。

2. 以下のファイルをご覧ください：

- [/proc/cpuinfo](#)：
- [/proc/meminfo](#)：
- [/proc/mounts](#)：
- [/proc/swaps](#)：
- [/proc/version](#)：
- [/proc/パーティション](#)：
- [/proc/interrupts](#)：

その名前から、どのような情報が明らかになるかがよくわかる。
この情報は常に更新されているわけではなく、見たいときにだけ得られるものである。

3. 任意のプロセス・ディレクトリを覗いてみよう（自分のプロセスでない場合は、**sudo**を使わない限り、情報の一部が制限されるかもしれない）：

```
$ ls -F 4435
```

attr/	コレダンプフィルタ	gid_map	マウント情	sessionid	シス
報	oom_score_adj	オートグループ	cpuset	IO	マウ
ント	ページマップ			setgroups	テム
auxv	cwd@	制限	マウント統計	smaps	コー
リディ	cgroup	環境	ログインID	stack stat	ルタ
	projid_map	clear_refs	exe@	statm	スク
	root@		map_files/	status	timerslack_ns
コマンドライン	fd/	マップ	oom_adj		uid_map
	スケジュールコマンド	fdinfo/	メモ		ワンチャン
	スコア	スケジュースタット			

cmdline、cwd、environ、mem、statusなどのフィールドを見てみよう。

第3章

ユーザー環境



3.1	環境変数.....	58
3.2	コマンドの歴史.....	62
3.3	コマンド・エイリアス.....	66
3.4	研究室.....	67

学習目標

このセッションを終えるまでに、あなたは次のことができるようになるはずだ：

- 環境変数を理解し、使用し、設定し、変更し、エクスポートする。
- 過去のコマンド履歴を調べ、以前のコマンドを検索、編集、使用する。
- 頻繁に使用するコマンドやオプションのエイリアスを設定します。

57

3.1 環境変数

環境変数

- 名前と変数の値を関連付ける

```
ホー  
ム ホ  
スト  
パス
```

- さまざまなフォーマットで掲載可能:

```
env  
輸出  
セット
```

- 変数を参照するときは、すべて先頭に\$を付ける:

```
echo PATH = $PATH
```

- 定義されているとき以外はね:

```
MYCOLOR=blue
```

等号の周りにはスペースを入れてはいけない!

`env`、`set`、`printenv`で環境変数を列挙すれば、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使用し、環境変数を期待している。**Linux**は多くのことに環境変数を使用し、システム起動時に設定する。自分の目的のために自分で作成し、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`

```
BASH=/usr/bin/bash
```

`エクスポート | ヘッド -2`

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

エコー \$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

```
/ホーム/bjmoose
```

いくつかの重要な環境変数

- ホーム
 - 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
- パス
 - 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
- PS1
 - カスタマイズが容易なコマンドラインプロンプト
- 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

```
/ホーム/rjsquirrel
```

エコー \$PATH

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

エコー \$PS1

```
\PWD
```

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か、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

ソース .bashrc

エコー \$MYVAR

```
マイバリュー
```

環境変数のエクスポート

- 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、その動作が変わる：

```
VAR=value ; エクスポートVAR  
export VAR=value
```

- エクスポートした変数を一覧表示できる：

輸出

デフォルトでは、スクリプト内で作成された変数は、現在のシェルのみで使用できます。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** コマンドを使っ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ステップで行えます：

```
export VAR=value
```

または2段階で：

```
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搬しないことに注意してください。

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

輸出

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

3.2 コマンド履歴

歴史

- 以前の履歴を表示するコマンド:

歴史

- シェルは〜 `./bash_history` に履歴を保存する。
- 履歴ファイルの場所: HISTFILE
- 履歴ファイルの最大行数: HISTFILESIZE
- 現在のセッションにおける履歴リストの最大行数: HISTSIZE

履歴ファイルの場所を表示する。

`echo $HISTFILE`

```
/ホーム/jbond/.bash_history
```

履歴ファイルの最大サイズを行単位で表示する。

`echo $HISTFILESIZE`

```
2000
```

履歴ファイルの現在のサイズを行単位で表示する。

`エコー $HISTSIZE`

```
1000
```

履歴ファイルの最初の2行を表示する。

`ヒストリー | ヘッド -2`

```
40 CDリリース
41 ls -ltr
```

以前のコマンドの呼び出し、編集

- 前後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：
 - !!
- 以前に使用したコマンドを検索します：
 - CTRL-R
 - 複数回打つと、前のコマンドを逆方向にスキャンする。

矢印キーを使えば、基本的に履歴リストを上下に移動することができる。

!!(しばしば「バンバン」と発音する) は、直前のコマンドを実行する。以

下は、CTRL-Rで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キー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をタイプして実行する

ヒストリーを使った過去のコマンド

- 歴史の代筆を始める：

!

- 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：

!n

- 文字列で始まる最新のコマンドを参照する

履歴の置換はすべて「!」で始まる。したがって、コマンドでは

`ls -l /bin /etc /var`

!`$`は、上の行の最後の引数である`/var`を指す。

歴史

```
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
```

`$!!` # 上記のコマンド #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
```

3.3 コマンド・エイリアス

別名

- エイリアスを作成してカスタマイズされたコマンドを作成する

エイリアス `name=command`

等号の周りにスペースがないことに注意

- 永続化するには、`~/.bashrc`を編集する。
- のように、標準的なプログラムの動作を変更するのに優れている:

エイリアス `rm='rm -i'`

- エイリアスを取り除く:

`unalias 名`

- 現在アクティブなエイリアスを一覧表示します:

エイリアス

エイリアスはカスタム定義が可能です。`alias`を引数なしで入力すると、定義されているエイリアスのリストが表示される。

いくつかの例を挙げよう:

```
エイリアス l='ls -laF'
エイリアス dir='ls -latF'
エイリアス rm='rm -i'
alias mv='mv -i'
alias cp='cp -ipdv'
エイリアス df='df -T'
alias gitstat='git status -uno'
alias diffside='diff --side-by-side --ignore-all-space'
alias scapt='gnome-screenshot -i -w'
```

3.4 ラボ

🔗練習3.1: パスに~/workディレクトリを追加する

```
echo HELLO、これはニセのlsプログラムだ。
```

この行だけを含む小さなファイル、~/work/lsを作成する：

そして、このファイルを実行可能にする：

```
$ chmod +x ~/work/ls
```

1. パスに~/work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する。lsと入力し、どちらのプログラムが実行されるかを確認する：/bin/lsか~/work/lsか。
2. 通常のパスが考慮される前に、~/workが検索されるように、~/workをあらかじめパスに追加しておく。もう一度、lsと入力し、どちらのプログラムが実行されるかを確認する：/bin/lsか~/work/lsか。

このようにパスを変更する際のセキュリティ上の考慮点は？

🟢解答 3.1

まず、エディターを使って、あるいはただ単純に、ニセのlsプログラムを作成する：

```
$ echo "echo HELLO、これはニセのlsプログラムです。" > ~/work/ls
```

```
$ chmod +x ~/work/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、新しいシェルを起動するとよい。bashと入力するだけで、新しいシェルを起動できる。

1. バッシュ

```
PATH=$PATH:~/work
```

```
bin etc games include lib lib64 libexec local sbin share src tmp
```

終了

2. バッシュ

```
PATH=~/:work:$PATH
```

```
ls /usr
```

```
ハロー、これはニセのLSプログラムだ。
```

終了

**非常に重要**

もし誰かが悪意のあるプログラムを `~/work` に入れることができれば、あなたを騙して誤ってそれを実行させることができる。

`PATH`内のディレクトリに他のユーザーが**書き込み**権限を持っていないことを確認してください。

 練習3.2: コマンド履歴

あなたはLinuxワークステーションで忙しく働き、ある特定の**bash**コマンドシェルで100個ほどのコマンドを入力した。

以前のある時点で新しいコマンドを使ったが、正確な名前は忘れてしまった。

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。

指令が何であったかをどうやって確認するのですか？

一度履歴でコマンドを見つけたら、プロンプトでそのコマンドをすべて入力することなく、簡単にもう一度コマンドを発行するにはどうすればいいのだろう？

🟢解答 3.2

コマンドの履歴：

履歴コマンドは、あなたが入力したコマンドを表示する方法です：

歴史

```
1 cd
2 ls
3 cd
4 プッシュボタン
5 エコー $SHELL
6 ls /var/
7 ls /usr/bin
8 ls /usr/local/bin
9 man fstab
10 ls
... -
```

以前のコマンドを再実行するには、いくつかの選択肢がある。例えば、最初にログインしたときに実行した**man**コマンドをもう一度実行したいとしよう。次のようにタイプする。

\$!9

と入力して、#9としてリストされているコマンドを再実行する。入力した**man**コマンドがこれだけなら、次のように入力することもできる。

!マン

これで、入力したコマンド名を覚えている。最後に、いくつかの**man**コマンドを入力したなら、CTRL-Rで履歴をさかのぼって検索し、再実行したい特定の**man**コマンドを見つけ、Returnを押すだけで実行できる。

📌練習3.3: 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何度も入力するのはかなり面倒で、タイプミスなどの些細なミスにつながる。**エイリ**

アスを導入することで、ショートカットを定義し、タイピングの手間を省くことができる。

あなたがプロジェクトチームのメンバーで、プロジェクト用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にあります。

プロジェクトXで作業していると、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よくあります。入力するまでにそれほど時間はかからない：

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

退屈になる。

projx "というエイリアスを定義して使い、上記のcdコマンドを実行させる。

🟢解答 3.3

コマンドの別名:

エイリアスの行は次のようになる:

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

一重引用符の代わりに二重引用符を使ってもいいし、定義したエイリアスには空白がないので、引用符をまったく使わなくてもいい。

このディレクトリに移動するために必要なことは、次のとおりです。

```
$ projx
```

エイリアスを永続化するには、\$HOME/.bashrcファイルに置くだけでいい。

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	72
4.2	シェル起動ファイル	75
4.3	ユーザーアカウントの管理	78
4.4	ロックされたアカウント	80
4.5	パスワード	81
4.6	/etc/shadow	82
4.7	パスワード管理	84
4.8	パスワードのエージング	85
4.9	ルートアカウント	86
4.10	SSH	88
4.11	研究室	95

学習目標

このセッションを終えるまでに、あなたは次のことができるようになるはずだ：

- 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。
- 新しいユーザーアカウントの作成、既存のアカウントプロパティの変更、アカウントの削除やロックが可能です。
 -
- ユーザーのパスワードがどのように設定され、暗号化され、保存されるかを理解し、セキュリティのためにパスワードの長期的な変更を要求する方法を理解する。
- rootアカウントの役割と使用するタイミングを理解する。
- セキュア・シェル (**ssh**) を使用し、ログインとコマンドを削除する。

4.1 ユーザーアカウント

ユーザーアカウントの目的

- 利用者一人ひとりに、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。
- ユーザー間の権限を区別する。

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する。

1つの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。高価なミス为了避免のため、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ときだけ使うべきです。

通常のユーザーアカウントは、システム上で作業する人のためのものである。いくつかのユーザー・アカウント（**デモン・アカウント**のような）は、プロセスがroot以外のユーザーとして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関心に従ってファイルや権限などを共有できる**グループ**管理についても議論する。

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: ウ ォ ー ド ・ ク  
リ ー バ ー :/home/warden:/bin/bash dobie:x:1002:1002: ド ビ  
ー ・ ギ リ ス :/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

4.1. ユーザーアカウント

73

システム上の各ユーザーは、`/etc/passwd`ファイルに対応する行を持ち、基本的なアカウント属性を記述している（パスワードとこのファイルについては後で説明する）。ここには7つの要素がある:

1. ユーザー名

各ユーザーに割り当てられた固有の名前。

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。

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活動の追跡など、さまざまな目的でシステムによって使用される。

ユーザのプライマリ、プリンシパル、またはデフォルトグループを示します。

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)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。(GECOSが何を意味するかは心配しないでください。)

6. ホームディレクトリ

ほとんどのユーザーにとって、これはユーザーの作業領域を提供する固有のディレクトリーである。通常、このディレクトリはユーザーによって所有され、**rootを除けば**、システム上の/homeのどこかにある。

7. ログインシェル

通常、これは/bin/bashや/bin/cshのようなシェル・プログラムである。しかし、特殊な場合には、別のプログラムがここで参照されることもある。一般に、このフィールドはどんな実行ファイルでも受け付ける。

現在のユーザーを決定する

- 現在ログインしているユーザーを調べる：
 - 誰
- 現在のユーザーIDを調べる：
 - ホワミ
- 現在のユーザーID、グループID、およびそれらの数値バージョンを調べる：
 - アイドル

例えば、こうだ：

誰

```
ユーザ ポイン 2023-03-09 12:58 (:0)
ー ト/0
ユーザ 点／1 2023-03-09 23:23 (:0)
ー
ユーザ 点／2点 2023-03-09 23:55 (:0)
ー
ユーザ 点／3点 2023-03-09 23:37 (:0)
ー
ユーザ 点／4点 2023-03-09 23:38 (:0)
ー
ユーザ 点／5点 2023-03-31 09:47 (:0)
ー
ユーザ ポイン 2023-03-31 11:13 (:0)
ー ト/6
ワル ポイン 2023-04-01 16:54 (ニャー.ロー
ト/8 カル)
```

\$フーアミ

```
ユ
ー
ザ
ー
```

id

```
uid=1000(user) gid=1000(user)グループ=1000(user),999(qubes)
```

4.2 シェル起動ファイル

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作成します。
- インタラクティブな環境は、多少異なる影響を受ける
- **etc**のファイルはグローバルな設定を提供する。
- ユーザー・ホーム・ディレクトリのファイルはグローバル設定を上書きする

新しいシェル（コマンドウィンドウまたは実行されるスクリプトのいずれか）が実行を開始するたびに、適切な機能を確保するために使用される要素を含むファイルが含まれます。これには以下が含まれる：

- 多くのプログラムやスクリプトで使用する**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われる**エイリアス**を定義する。
- 後続のスクリプトで使用できる**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初期化ファイルがあり、個別のファイルが使われる前に、すべてのユーザーによって使われる。

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る。

スタートアップファイルの利点

- ユーザーのプロンプトをカスタマイズする
- ユーザーの端末タイプを設定する
- 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を設定する

スタートアップ（初期化）ファイルの処理がなければ、コマンドやプログラムが実行されるたびに、適切に機能するためのセットアップ作業が発生する可能性がある。

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する。

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見ることができます。

nano;ユーザーが使いたいほうのエディター。

上記の例は、環境変数のセクションで紹介する。

起動ファイルの順序

- 全ユーザーのデフォルト値
`/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` を上書きする
プロフィール
- インタラクティブな非ログインシェルの設定
`~/.bashrc`
- 通常、`~/.bashrc`をカスタマイズするだけでよい。

Linuxにログインすると、`/etc/profile`が常に読み込まれ、評価され

る。次に、以下の順番でファイルが検索される：

`~/.bash_profile`

`~/.bash_login`

プロフィール

Linuxのログイン・シェルは、最初にたどり着い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
を評価し、残りのファイルはすべて無視する。これは冗長に聞こえるかもしれないが、さまざまなLinuxディスト
リビューションが異なるスタートアップ・ファイルを使用する傾向がある。

ログインしていない状態でサブシェルを作成するたびに、`~/.bashrc`だけが読み込まれ評価される。ログインシェルで
は読み込まれず評価されないが、ほとんどのディストリビューションとユーザーは、ユーザーが所有する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。

したがって、カスタマイズの大部分は`~/.bashrc`に入れるべきである。

4.3 ユーザーアカウントの管理

useraddによるユーザーアカウントの作成

- useraddでデフォルトの操作が可能

```
sudo useradd bjmoose
```

- アカウントbjmooseを作成
- 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
```

- シェル、スケルディレクトリ、コメントフィールドをオプションで指定する。
 -

```
sudo useradd bjmoose
```

は以下のステップを実行させる：

- デフォルトでは、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る。
- 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が所有する。
- 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にコピーされる。デフォルトでは、/etc/skelにはbashとX Windowシステムの起動ファイルが含まれています。
- `!`というエントリーが、`/etc/shadow`ファイルのbjmooseのエントリーのパスワード・フィールドに置かれる。このた

め、アカウントを使用可能にするには、管理者がパスワードを割り当てる必要がある。

デフォルトは、**useradd**のオプションを使って簡単に上書きできる：

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

ここで、いくつかのユーザー属性には、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できます。

```
sudo userdel rjsquirrel
```

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

- ユーザーアカウントは**usermod**で変更できる

```
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への参照がすべて消去されます。これによりアカウントは削除されるが、ホームディレクトリ（通常は`/home/rjsquirrel`）は削除されない。**userdel** に `-r` オプションを指定すると、ホームディレクトリも消去される。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を変更します。

```
student@CentOS7:~$ sudo usermod --help
[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る**
 - bin、daemon、sysなどのアカウント
 - プログラムを実行することはできるが、ログイン目的で使用されることはない。
- ロックされたアカウントは、他の目的で作成されることもある。
 - データベースのような主要アプリケーションで使用するアカウント
- 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「!
- すべてのロックされたアカウントが無効なシェル（通常 `/sbin/nologin`）

Linuxには、ロックされたアカウントがいくつか同梱されている。つまり、プログラムを実行することはできるが、システムにログインすることはできず、有効なパスワードも関連付けられていない。例えば、`/etc/passwd`には以下のようなエントリーがある：

```
bin:x:1:1:bin:/bin:/sbin/nologinデーモン:x:2:2:
デーモン:/sbin:/sbin/nologin
```

このアカウントは現在ご利用いただけません。

ロックされたユーザーがシステムにログインしようとすると、`nologin`シェルは以下のように返す：

または`/etc/nologin.txt`に格納されているメッセージのどれでもよい。`nologin`シェルで`/etc/passwd`をスキャンしてユー

ザーを調べれば、システム上のユーザーを知ることができる。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ある：

```
sudo usermod -L bjmoose
```

これは、アカウントはシステム上に残るが、ログインは不可能であることを意味する。ロック解除は-Uオプションで行うことができる。慣例として、ユーザが組織を離れたり、長期休暇に入るたびに、そのユーザのアカウントをロックする。アカウントをロックするもう1つの方法は、**chage**を使ってアカウントの有効期限を変更することである：

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

実際の日付は過去であれば関係ない。

4.5 パスワード

ユーザーIDと/etc/passwd

```
beav:x:1000:1000:セオドア・クリーバー:/home/beav:/bin/bash
rsquirrel:x:1001:1001:ロケット・J・リス
:/home/rsquirrel:/bin/bash
```

- `/etc/passwd` ファイルには、各ユーザーにつき1レコード（1行）が含まれ、各レコードはコロン（:）で区切られたフィールドのリストである：
 - 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`を使用しない場合）、またはプレースホルダー（`/etc/shadow`を使用する場合は "x"）。
 - UID: ユーザー識別番号
 - GID: ユーザーの主なグループ識別番号
 - comment: コメント欄、通常はユーザーの実名
 - 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から始まります。

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から始まる。実際の値はUID_MINとして定義され `/etc/login.defs`。

歴史的に、**Red Hat** 由来のディストリビューションは UID_MIN=500 を使用しており、1000 を使用していませんでしたが、**RHEL 7** 以降、より一般的な値である 1000 が採用されました。

84 **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る。

第4章 ユーザーアカウント管理

さらに、各ユーザーは**プライマリグループID**を取得するが、これはデフォルトでは**UID**と同じ番号である。これらは**UPG** (**User Private Groups**) と呼ばれることもある。

`etc/passwd`、`/etc/group`、`/etc/shadow` を直接編集するのは良くない習慣です。**usermod** のような適切なユーティリティを使うか、上級ユーザーには **vipw/vigr** という少し安全なツールを使ってください。

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る。

`/etc/passwd`のデフォルトのパーミッションは644（`-rw-r--r--`）で、誰でもファイルを読むことができる。システム・プログラムやユーザー・アプリケーションは、このファイルに含まれる情報を読む必要があるため、これは残念ながら必要なことである。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`だけがファイルを変更することができる。

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`にある場合、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**のような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パスワードを推測することができる。これはセキュリティ・リスクである！

`/etc/shadow`のパーミッション設定は400（`-r-----`）、つまり`root`だけがこのファイルにアクセスできる。このため誰かがハッシュ化されたパスワードを収集することがより困難になる。

やむを得ない理由がない限り、`/etc/shadow`ファイルを使うべきだ。

/etc/shadowフォーマット

- `/etc/shadow` ファイルには、各ユーザーにつき1レコード（1行）が含まれている。
- 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ます。
 - username: 一意のユーザー名。
 - password: パスワードのハッシュ化（sha512）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: パスワードを変更するまでの最短日数
 - 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何日前にユーザーに警告を発するか。
 - 猶予: パスワードの有効期限が切れた後、アカウントが無効になるまでの日数
 - expire: アカウントが無効になる／無効になる日付
 - reserved: 予約フィールド

`/etc/shadow`には、以下のように各ユーザーごとに1レコード（1行）が含まれている:



`/etc/shadow`

```
daemon*:16141:0:99999:7:::
-----
beav:$6$SiCZyCnBJH9rmq7P.$RYNm10Jg3wrhAtUnahBZ/mTMg.RzQE6iBXyqaXHvxxbK
TYqj.d9wpoQFuRp7fPEE3hMK3W2gcIYhiXa9MIA9w1:16316:0:99999:7:::
```

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ていなければならず、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**）からの日数として保存される。

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が続く。

文字（sha512）のパスワード・ハッシュ。

4.7 パスワード管理

パスワード管理

- パスワードは`passwd`で変更できる
- ユーザーは自分でパスワードを変更できる
- `root`はすべて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。パスワードを変更する一般ユーザー:

パスワード

```
bjmooseのパスワード変更
(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の新しいパスワード>
新しいUNIXパスワードを再入力する。
passwd: すべての認証トークンが正常に更新された。
```

`root`がユーザーのパスワードを変更するとき、`root`は現在のパスワードの入力を求められないことに注意:

`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: <rjsquirrel's-password>を再入力
する。
passwd: すべての認証トークンが正常に更新された。
```

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、不正なパスワードを設定することはできない。ただし、`root`は許可されている。

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）

- チャージを使用して管理

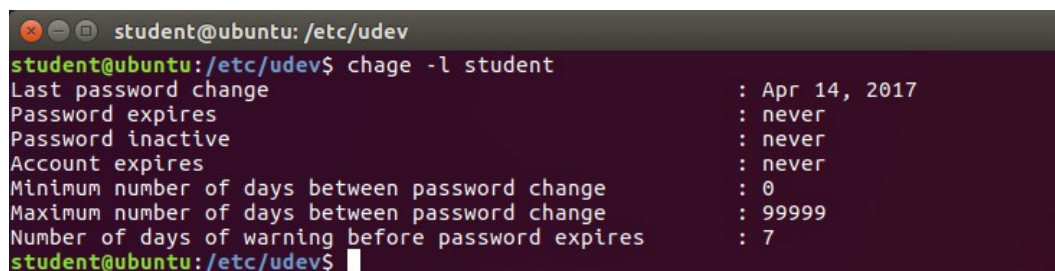
```
$ chage [-m mindays] [-M maxdays] [-d lastday] \
        [I inactive] [-E expiredate] [-W warndays] ユーザー
```

- 例を挙げよう：

```
sudo chage -l 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る。こうすることで、クラックされたパスワードが侵入者にとって有用である時間を制限することができ、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もできる。欠点は、ユーザーがこのポリシーを煩わしく思い、変更し続け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ことだ。これを管理するユーティリティが**chage**である：

```
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: チャージの使用

chageを使えるのはrootユーザーだけである。例外として、どのユーザーでも-lオプションをつ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を知ることができる。

次回ログイン時にパスワードを強制的に変更させる：

4.9 ルートアカウント

ルートアカウント

- rootアカウントはシステムにフルアクセスできる
- ユーザーにroot権限を与えない
- rootアカウントに変更する：
ス
- rootとしてコマンドを実行する：
sudo コマンド
- の**sudo**設定： `/etc/sudoers` と `/etc/sudoers.d` にあります。
- **su**よりも**sudo**の方が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。これは非常に強力なアカウントだ。

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る。**sudo**は、特定のアカウントだけにこの権限を持たせたり、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たりするように設定できる。詳細は **man sudo** を参照のこと。

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する。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特権で実行される。

ミスは必ず起こります！ 多くの重要なファイルを削除してしまうような、大きなミスを犯さないためには **su**サブシェルは、昇格特権を必要としないコマンドを実行する。

ルートアカウント

ルートアカウントの使用は稀な例外であるべきだ

- どうしても必要なときだけrootでログインする
 - ミスを避ける
 - 潜在的なセキュリティの脆弱性を回避する
- root権限を取得するには**su**の使用が必要
 - ルートからの直接ログインを禁止する
 - 誰がrootになったかの監査証跡を設定する
- セキュリティを強化するには、**sudo** を使用します。

4.9. ルートアカウント

87

rootアカウントは、絶対に必要な場合のみ管理目的に使用し、通常のアカウントとしては決して使用しないこと。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面で、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 ログインは一般的に禁止されている。[etc/ssh/sshd_config](#)で設定されるsshと、[pam securetty.so](#)モジュールと関連する[etc/securetty](#)ファイルを通して設定される**PAM**（Pluggable Authentication Modules）を使用した**Secure Shell**ログインを許可することができる。にリスト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、rootアクセスはすべて**su**または**sudo**を使うことが推奨されています（rootアクセスはすべて、**su**または**sudo**を使うことで監査証跡が残ります）。

sudo）。ディストリビューションによっては（**Ubuntu**など）、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ものもあるので注意してください。

PAMは、rootへの**su**を許可するユーザーを制限するのにも使える。rootとして実行されたすべてのコマンドをログに記録するように**auditd**を設定することも価値があるかもしれない。

4.10 SSH

SSH

- **ssh**: リモートシステムにログインしたり、コマンドを実行したりする。
- **scp**を含む: リモートシステムとのファイル転送
- セキュリティのために強力な暗号化を使用（これがSecure **SHell**という名前の由来です）。

```
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
```

いや、もっと簡単だ:

```
$ pssh -viH 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はよくある。あるいは、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ある。

SSH (Secure **SH**ell) はこの目的のために存在する。SSHは強力なアルゴリズムに基づく暗号化を使用する。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ない。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としています。別のユーザーとしてログインするには

`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:

`$ scp file.txt farflung.com:/tmp`
`$ scp file.tex student@farflung.com/home/student`
`$ scp -r some_dir farflung.com:/tmp/some_dir`

(スペースを節約するため、パスワードの要求は省略しました)。

pssh (Parallel SSH) ユーティリティを使えば、複数のシステム上でコマンドを一撃で実行することができる:

`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には、**psshのマニュアル**ページを読む必要があるかもしれない。

ssh 設定ファイル

- ホームディレクトリに
 - `id_rsa`: ユーザーの**秘密**暗号鍵。
 - `id_rsa.pub`: ユーザーの**公開**暗号鍵。
 - `authorized_keys`: ログインを許可する公開鍵。
 - **既知のホスト**: 過去にログインが許可されたホスト。
 - `config`: 様々なオプションを指定するためのファイル。
- `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・ディレクトリーの下にある隠し `.ssh` ディレクトリーに作成されます：

まず、ユーザーは**ssh-keygen**で秘密鍵と公開暗号鍵を生成しなければならない:

ssh-keygen

公開/秘密鍵ペアの生成

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、パスワードなしアクセスを許可したいマシンに 渡すべきである。認可された鍵には、このマシンであなたとしてログインできる公開鍵が含まれていることに注意。

cat authorized keys

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCAKc= 学生@debian
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCAKc= 学生@fedora
```

`known_hosts`は、**ssh**アクセスが発生するにつれて徐々に蓄積される。**ssh**でログインしようとしているユーザーの変更をシステムが検知した場合、警告が表示され、アクセスを拒否することができます。

cat known hosts

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

SSH設定ファイルの優先順位

設定ファイルの処理順序:

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

ファイルとコンテンツの優先順位は、**最初に使用されるマッチ**である。



`/home/student/.ssh/config`

ホストアップル

ホスト名 **192**.168.0.196

利用者 ポート

4242

IdentityFile /home/student/.ssh/custom

ホストのaws

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

User ubuntu

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

ForwardX11 no

パスワード認証なし

ホスト

上記の`~/.ssh/config`には、ホストappleとawsに固有のコンフィギュレーション情報が記載されている。どちらも一致しない場合は、汎用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。コマンドの使用法は以下のようになる:

`ssh apple`

または

クラウドシステム上のSSH

- クラウドシステム上のSSHも、多少の自動化を施せば同じように機能する。
- sshキーは、クラウド・システムの作成時にデフォルト・ユーザー用に生成されます。
- 公開鍵は、デフォルト・ユーザーの認証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。
- パスワードベースの認証は、一般的にクラウドシステム上のユーザーにはサポートされていません。

リモート・システムの公開鍵を置き忘れることは珍しくない。公開鍵の再作成は **ssh-keygen** で行うことができる：

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

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

公開鍵の名前を変更する：

```
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 student student 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 student student 96 Jul 11 07:01 id_ed25519.pub.lost
```


リモート・グラフィカル・ログイン

- フルグラフィカルなデスクトップでリモートマシンにログイン
- 多くの場合、**VNC**（仮想ネットワークコンピューティング）を使用します。
- 一般的な実装は**tigervnc**
- 簡単なことだ：
 - リモート・マシン（some_machineなど）：
`vncserver`
 - ローカルマシンで
`$ vncviewer -via server student@some_machine localhost:2`

これをテストするには、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する：

```
$ 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、以下のような方法でインストールする必要がある：

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

正しいパッケージ管理システムコマンドを使ってください。(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ないが、パッケージは大きくない)。通常のユーザーとしてサーバーを起動します：

```
vncserver
```

でテストできる。

```
$ vncviewer localhost:2
```

(現在使用しているソフトやマシンの設定によっては、2以外の数字(1、3、4、...)を使う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、少し違います：

```
$ vncviewer -via student@some_machine localhost:2
```

カラープロファイル」のために認証が必要だという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**デーモンを終了させる必要がある。

```
sudo systemctl stop colord
```

これは（機能ではなく）バグであり、はっきりしない理由で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る。

4.11 ラボ



ビデオ・デモンストレーション・リソース

ユーザーアカウント使用デモ.mp4

演習4.1: 勘定科目

どのアカウントでログインしていますか? どうやって確認するのですか?

✓解決策 4.1 idユーティリティを実行する:

id

```
uid=1000(user) gid=1000(user) グループ=1000(user),999(qubes)
```



ご注意

そのための方法はたくさんある。以下はその一部である:

\$フーアミ

```
grep ^Uid /proc/$$/status
```

演習4.2: 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てください。何が同じで何が違うか。
2. `useradd`を使用して`user1`アカウントを作成する。
3. `ssh`を使用して`user1`としてログインします。できます:
`ssh user1@localhost`
`user1`のパスワードが必要なので失敗するはずです。
4. `user1`のパスワードを`user1pw`に設定し、`user1`として再度ログインを試みる。
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`ファイル。
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する。`etc/login.defs`ファイルも見てください。
7. デフォルト・シェルとして**Korn**シェル (**ksh**) を使用する`user2`のユーザー・アカウントを作成する。(`/bin/ksh`がない場合

をインストールするか、`/bin/csh`のCシェルを使う)。パスワードをuser2pwに設定する。

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？
9. `chage`を使って、user1のアカウントの有効期限を2013年12月1日に設定する。`etc/shadow`を見て、新しい有効期限を確認する。

10. `usermod`を使ってuser1アカウントをロックする。

`etc/shadow`を見て、user1のパスワードについて何が変わったかを確認する。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。

✔解答 4.2

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$IGFFUU8ctrtrtOG0istJ4/30DrNLI1FS66qnn0VbS6Mvm
luKI08SgbzT5.IcOH05j/SOdCagZmF2RgzT vzLb11H0:16028:0:99999:7:::
```

(studentの代わりに通常のユーザー名でも構いません。)一致するのはユーザー名の欄だけです。

2. `sudo useradd user1`

3. `ssh user1@localhost`

user1@localhostのパスワード:

というように、まずsshdサービスを立ち上げる必要があるかもしれない:

`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

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

```
/etc/passwd:user1:x:1001:100::/home/user1:/bin/bash
/etc/shadow:user1:$6$OBE1mPMw$Cic7urbQ9ZSnyiniVOeJxKqLFu8fz4whfEexVem2
TFpucuwRN1CCHZ19XGhj4qVujslRIS.P4aCXd/y1U4utv.:16372:0:99999:7:::
```

6. 例えばRHELやopenSUSEのシステムで:

`cat /etc/default/useradd`

```
# ユーザー追加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`

....

つ目のファイルはかなり長いので再現しないが、あなたのシステムで調べてみてほしい。

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

8. `sudo grep user1 /etc/shadow`

```
user1:$6$OBE1mPMw$C1c7urbQ9ZSnyiniVOeJxKqLFu8fz4whfEexVem2TFpucuwRN1CCHZ  
19XGhj4qVujslRIS.P4aCXd/y1U4utv.:16372:0:99999:7:::
```

有効期限はないはずだ。

9. `sudo chage -E 2013-12-1 user1`
`sudo grep user1 /etc/shadow`

```
user1:$6$OBE1mPMw$C1c7urbQ9ZSnyiniVOeJxKqLFu8fz4whfEexVem2TFpucuwRN1CCHZ  
19XGhj4qVujslRIS.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	100
5.2	団体会員	101
5.3	グループ経営	102
5.4	ユーザー・プライベート・グループ	103
5.5	研究室	104

学習目標

このセッションを終えるまでに、あなたは次のことができるようになるはずだ：

- Linuxユーザを1つ以上のグループに所属させることが、なぜ便利なのか説明してください。
- グループメンバーシップの概念を説明する。
- **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・グループについて説明する。

5.1 グループ

グループ

- ユーザーが作業領域（ディレクトリ、ファイルなど）を共有できるようにする。
- グループメンバーにはアクセスを許可するが、全世界には許可しないファイルパーミッションの設定
- 特定のユーザーに対して、他の方法ではアクセスできないリソースへのアクセスを許可する。

Linuxシステムは**グループ**と呼ばれるユーザーの集まりを形成し、そのメンバーは何らかの共通の目的を共有している。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、共通の特権を保持します。これにより、システム上の他のユーザー、時には**ワールド**と総称される他のユーザーと区別されます。グループの使用は、共同プロジェクトを非常に助けます。

ユーザーは1つ以上の**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つ役割と同じ役割をグループに対して持つ。ファイルの各行は次のようになっている：



`/etc/group`

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

- `groupname`はグループ名。
- パスワードはパスワード・プレースホルダーである。グループパスワードは設定できるが、`/etc/gshadow`が存在する場合には

限る。

- GIDはグループ識別子。0から99の値はシステムグループ用である。100からGID_MIN（[/etc/login.defs](#)で定義され、通常はUID_MINと同じ）までの値は特別なものとみなされる。GID_MINを超える値はUPG（ユーザー・プライベート・グループ）用である。
- user1,user2,...は、グループのメンバーであるユーザーのコンマ区切りリストである。このグループがユーザの主グループである場合、ユーザをここにリストする必要はありません。

5.2 団体会員

団体会員

Linuxユーザー：

- 主要グループは1つ
 - プライマリグループは/etc/passwdにリストされているグループです。
 - また、/etc/groupにもリストされる。
- 0から15の二次グループに属する可能性
- 以下のいずれかを実行することで、所属グループを特定できる。
`グループ [ユーザー1 ユーザー2 ...]`
`id -Gn [user1 user2 ...].`
- 引数なしの場合、どちらのコマンドも現在のユーザーについて報告する。

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である。他のセカンダリ・グループに所属することで、ユーザーに追加のパーミッションが与えられます。

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特定することができます。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`

学生

Ubuntuで：

`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は、以下の方法で管理・維持することができる：

- **グループ追加** 新しいグループを追加する
- **groupmod**：グループの属性を変更する
- **groupdel**：グループを削除する
- **usermod**：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。

例を挙げよう：

```
sudo groupadd -r -g 215 staff
sudo groupmod -g 101 blah
sudo groupdel newgroup
sudo usermod -G student,group1,group2 student
```



非常に重要

注：**usermod -G**コマンドはグループの全使用リストであるため、1つのコマンドラインでグループの削除と追加のすべてを行う。非破壊的な使い方をする場合は、新しいグループを追加する際に既存のグループ・メンバーシップを保持する**-a**オプションを使うべきである。

ープ

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。
- **umask**は、UPGで作成されたすべてのユーザーに対して002に設定される (**umask**については後で詳しく説明する)。
- ユーザー・ファイルはパーミッション664で作成され、ディレクトリは775で作成される。

Linuxでは、**ユーザー・プライベート・グループ** (UPG) を使用します。

UPGの背後にある考え方は、各ユーザーが自分のグループを持つことです。しかし、UPGはプライベートであることが**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**でアカウントが作成されたユーザーは、プライマリGID = UIDであり、グループ名もユーザー名と同じである。

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の**umask**は002に設定される。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)、ディレクトリは775 (rwxrwxr-x)で作成される。(umask については後で説明する)。

5.5 ラボ

✍️練習5.1: グループでの作業

- 2つの新しいユーザーアカウント（以下ではrockyとbullwinkle）を作成し、ホームディレクトリがあることを確認する。
- friendsとbossesの2つの新しいグループを作成する（GIDは490）。etc/groupを見る。それぞれの新しいグループに与えられたGIDを確認する。
- 両方の新しいグループにロッキーを加える。
bullwinkleさんをグループフレンドに追加
etc/groupを見て、どのように変更されたかを確認する。
- rockyとしてログインする。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。（次のセッションで説明するchgrpを使う）。
(おそらく、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だろう)。
- bullwinkleとしてログインし、touchコマンドを使って/home/rocky/somedirにsomefileというファイルを作成してみてください。できるか？ いいえ、グループ所有権とディレクトリのchmod a+xのせいです。
- bullwinkleをbossesグループに追加し、もう一度試してください。新しいグループメンバーシップを有効にするには、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。以下を実行してください：

✅解答 5.1

- 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する：
passwd: パスワードの更新に成功

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する：
passwd: パスワードの更新に成功

ls -l /home

```
合計12
drwxr-xr-x 2 bullwinkle bullwinkle 4096 Oct 30 09:39 bullwinkle
drwxr-xr-x 2 ロッキー      ロッキー    4096 Oct 30 09:39 ロッキー
drwxr-xr-x 20 学生          学生        4096 Oct 30 09:18 学生
```

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

```
友人:x:1003: 上司:x:490
:
```

5.5. ラボ

105

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

```
ロッキー : ロッキー・フレ  
ンズ・ボス ブルウィンクル :  
ブルウィンクル・フレンズ
```

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

```
合計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 にタッチできません: パーMISSIONが拒否されました
```

終了

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)

第6章

ファイルのパーミッションと所有権



6.1	ファイルのパーミッションと所有権	108
6.2	ファイルアクセス権	109
6.3	chmod、chown、chgrp	110
6.4	ウマスク	113
6.5	ファイルシステムACL	115
6.6	研究室	116

学習目標

このセッションを終えるまでに、あなたは次のことができるようになるはずだ：

- 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。
- 適切なパーミッションを尊重し、ファイルアクセス要求を認証する。
- ファイルのパーミッションを変更するには**chmod**を、ユーザーの所有権を変更するには**chown**を、グループの所有権を変更するには**chgrp**を使う。
- 新しく作成されたファイルに必要なパーミッションを設定する際の**umask**の役割を理解する。
- ACLを使用して、より単純なユーザー、グループ、ワールド、読み取り、書き込み、実行モデルを拡張する。

107

6.1 ファイルのパーミッションと所有権

オーナー、グループ、世界

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

```
ls -l
```

```
-rw-rw-r-- 1 学生 学生      1601 Mar 9 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
```

そうする時だ:

```
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・オブジェクトのタイプを示す）の後に、ファイル・ユーザー候補に付与された**アクセス権**を示す9つの文字が続く。これらは、**オーナー**、**グループ**、**その他**（**world**と同じ）。

上のリストでは、ユーザーはcoop、グループはaprojectである。

6.2 ファイルアクセス権

ファイルアクセス権

- ファイルの長いリストを作成する場合：

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

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

トリプレット（文字2～10）は、それぞれ以下の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る。
- **w**：書き込みアクセスが許可されている。
- **x**：実行アクセスが許可されている。

- 認証は、最初に成功した者から順に行われる：

- ファイルの所有権
- グループメンバー
- 世界の許可

パーミッションが許可されていない場合は、これらの文字の代わりに「-」が表示される。

さらに、**setuid/setgid**パーミッションなど、各カテゴリーに特化したパーミッションも存在する。

これらの**ファイル・アクセス許可**は、**Linux**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と比較する必要がある。

この**認証**は、以下の順序で、これら3つの権限セットのいずれかに応じて付与される：

1. 要求者がファイル所有者の場合、ファイル所有者パーミッションが使用されます。
2. そうでなければ、要求者がファイルを所有するグループに属している場合、グループのパーミッションが調査される
；

6.3 chmod、chown、chgrp

chmod

- ファイルのパーミッションの変更は**chmod**で行う。
- 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与え直す:

```
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はグループを表す。

- スーパーユーザーでない限り、自分が所有するファイルのパーミッションしか変更できない。

許可-cPanebremreispsreisoenntsedaeritheor alsoanbgnvpe nusiuallny uwmritteenriicn 2bmr n,
asusycmhboalicsfo6rm44 オクタルのビットマップは通常0755のように見えますが、記号表現は
u+rwx,g+rwx,o+rxのように見えます。

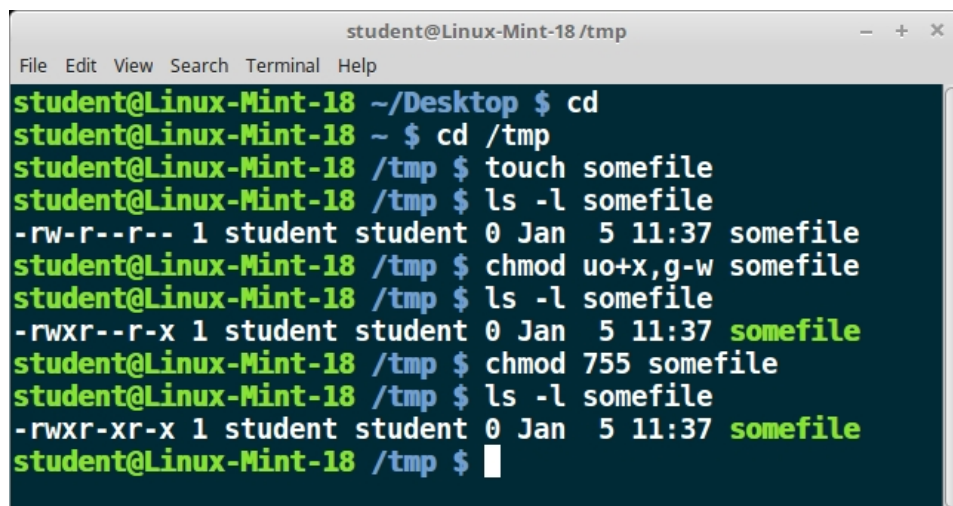
8進数

- 8進数表現は、各桁の和を表す：
 - 読み取り許可が必要な場合は4。
 - 書き込み許可が必要な場合は2。
 - 実行許可が必要な場合は1。

記号構文は入力が難しく、覚えるのも大変なので、8進数の省略記法を使うことが多い。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すべてを指定するには1桁の数字で十分である。

したがって、7はリード／ライト／実行、6はリード／ライト、5はリード／実行を意味する。

これを**chmod**で適用する場合、以下の例のように3桁の数字をそれぞれ指定する必要がある：

A terminal window titled 'student@Linux-Mint-18 /tmp' showing a series of commands and their outputs. The commands are: 'cd ~/Desktop', 'cd /tmp', 'touch somefile', 'ls -l somefile' (output: '-rw-r--r-- 1 student student 0 Jan 5 11:37 somefile'), 'chmod uo+x,g-w somefile', 'ls -l somefile' (output: '-rwxr--r-x 1 student student 0 Jan 5 11:37 somefile'), 'chmod 755 somefile', 'ls -l somefile' (output: '-rwxr-xr-x 1 student student 0 Jan 5 11:37 somefile'). The prompt is currently at 'student@Linux-Mint-18 /tmp \$' with a cursor.

```
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（スーパーユーザーのみ可能）

```
sudo chown wally somefile
```

- **chgrp**でグループの所有権を変更する（所属するグループのみ）

```
sudo chgrp cleavers somefile
```

- 両方を同時に変更する：

```
sudo chown wally:cleavers somefile
```

ここで、オーナーとグループをコロン（またはピリオド）で区切る。

- Rオプションは再帰的に使用できる：

```
sudo chown -R wally:cleavers ./
```

```
sudo chown -R wally:wally subdir
```

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

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に限られます。

Rオプションを使用すると、引数以下のすべてのファイルとディレクトリの所有者が確実に変更される。

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のパーミッションを設定するのに使用する。
- **umask**: umaskを設定するプログラム
 - 保留する許可をリストアップ
- [etc/profile](#)に設定されているデフォルトのumask
 - rootのumaskは022
- どのユーザーでも設定可能

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（0666）に対して読み取り/書き込み、ディレクトリに対しては全員（0777）に対して読み取り/書き込み/実行である。しかし、以下のようにすると

タッチ 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 9/16 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だろう。これらは現在の **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である。現在の値は

`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。この値は、ファイル作成パーミッションと組み合わせられ、実際の結果となる、

$0666 \& \sim 002 = 0664$; すなわち、`rw-rw-r--`。

umaskコマンドでいつでも変更できる。

`umask 0022`

6.5 ファイルシステムACL

ファイルシステムACL

- **POSIX ACL**（アクセス制御リスト）は、よりシンプルなユーザー、グループ、ワールドシステム
- 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のデフォルト設定
- ACLを取得/設定するにはgetfacl/setfaclを使用する。
- 例を挙げよう：

```
$ getfacl /home/stephane/file1
$ setfacl -m u:isabelle:rx /home/stephane/file1
$ setfacl -x u:isabelle /ホーム/ステファン/ファイル
```

特定のオブジェクトやオブジェクト・クラスにアクセスする際、特定のユーザーやユーザー・グループに特定の特権を与えることができる。

Linuxカーネルは**ACL**の使用を可能にしているが、特定のファイルシステムにも実装されていなければならない。最近のLinuxディストリビューションで使われている主要なファイルシステムにはすべて**ACL**拡張が組み込まれており、マウント時に-aclオプションを使うことができる。**ACL**のデフォルト・セットはシステム・インストール時に作成される。

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。

ACLを削除するには

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

ディレクトリにデフォルトを設定するには

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

6.6 ラボ



ビデオ・デモンストレーション・リソース

ウマスクデモ.mp4

練習6.1: chmodの使用

chmodを使うとき、パーミッションの指定には8進数かシンボリックのどちらかを使うことができる。シンボリック方式についてもう少し詳しく説明しよう。

直接パーミッションを与えることも、パーミッションを追加・削除すること

も可能です。以下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 afile
```

各ステップの後、こうする:

```
ls -l afile
```

でパーミッションがどうなったかを確認し、いくつかのバリエーションを試す。

練習6.2: umask

で空のファイルを作成する:

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

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

これを見ると、デフォルトでは所有者とグループには読み取りと書き込みの両方のパーミッションが与えられているが、worldには読み取りパーミッション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/書き込みである。

umaskと入力すれば、現在の値が表示される:

```
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。この値は、ファイル作成パーミッションと組み合わせられ、実際の結果となる、

$0666 \& \sim 002 = 0664$; すなわち、`rw-rw-r--o`。

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見てみよう：

```
umask 002
touch afile2
umask 0666
touch afile3
ls -l afile*
```

演習6.3: アクセス制御リストの使用

1. いつものユーザー名でファイルを作成し、getfaclを実行してプロパティを見る。
2. デフォルトのプロパティを持つ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っ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てみる。これは失敗するはずだ。
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、もう一度やり直してください。
5. setfaclを使って、新しいユーザーがファイルを読めないようにしてから、もう一度試してください。
6. 必要に応じて清掃する。

✓ 解答 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・ユーザーとして作業するのが最も簡単だろう。

1. ウィンドウ1:

```
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 '/' を取り除く # file:
tmp/afile
# owner: coop
# group: coop
user::rw-
group::rw-
other::r--
```

2. ウィンドウ1:

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

3. ウィンドウ2:

```
sudo su - fool
別の行をエコー > /tmp/afile
```

```
-bash: /tmp/afile: パーミッション拒否
```

4. ウィンドウ1:

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

\$ getfacl /tmp/afile

```
getfacl: 絶対パス名から先頭の '/' を取り除く # file:
tmp/afile
# owner: coop
# group: coop
user::rw-
user:fool:rw-
group::rw-
mask::rwx
other::r--
```


ウィンドウ2:

別の行をエコー > /tmp/afile

5. ウィンドウ1:

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

ウィンドウ2:

別の行をエコー > /tmp/afile

```
-bash: /tmp/afile: パーミッション拒否
```

6. 掃除だ:

```
rm /tmp/afile  
sudo userdel -r fool
```

第7章

パッケージ管理システム



7.1	なぜパッケージを使うのか?	120
7.2	ソフトウェアパッケージングコンセプト	121
7.3	パッケージの種類	122
7.4	利用可能なパッケージ管理システム	123
7.5	パッケージングツールのレベルと種類	124
7.6	パッケージ・ソース	125
7.7	ソフトウェア・パッケージの作成	126
7.8	リビジョン管理システム	127
7.9	利用可能なソース管理システム	128
7.10	Linuxカーネルとgit	129
7.11	研究室	131

学習目標

このセッションを終えるまでに、あなたは次のことができるようになるはずだ：

- 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管理システムを列挙する。
 - なぜ2つのレベルのユーティリティが必要なのかを理解する。1つは裸のパッケージだけを扱うもので、もう1つはパッケージ間の依存関係を扱うものである。
-
- 独自のパッケージを作成することで、ソフトウェアに何が含まれ、どのようにインストールされるか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、特にgitの役割を理解する。

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: 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: 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のシステムにインストールすることも可能です。
- 再現性と予測可能性
- 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 がエンタープライズ IT 環境にもたらした最大の進歩の 1 つとして広く見なされている。ファイルと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数千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・パッケージングのコンセプト

ソフトウェア・パッケージングのコンセプト

- 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)に集めて圧縮する。
- ソフトウェアのインストールや取り外しが容易
- 内部データベースを介してファイルの整合性を検証可能
- パッケージの出所を認証できる
- アップグレードの促進
- 論理的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の管理

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、設定ファイルなどが含まれます。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、すべての情報がローカルに内部データベースに保存され、バージョンステータスや更新情報を簡単に照会できる。

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、実行可能ファイルやライブラリなど、デプロイ準備の整った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。常にソース・パッケージからバイナリ・パッケージをリビルドできなければなりません。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・インタプリタの下で実行されるファイルやスクリプトのほか、ドキュメントや設定ファイルが含まれている。
- **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ある。

バイナリパッケージは、システム管理者がたいいていの場合扱わなければならないものである。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2つのバイナリ・パッケージがインストールされていることがある。

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、常にベンダから取得することができます。

ソースパッケージからバイナリパッケージをリビルドすることは常に可能であるべきだ。例えば、RPMベースのシステムでは、**p7zip** バイナリパッケージをリビルドすることができる：

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

これは結果を `/root/rpmbuild` に置く：

```
# 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
```

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する。

7.4 利用可能なパッケージ管理システム

利用可能なパッケージ管理システム



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)**

Red Hat Enterprise Linux、Fedora、CentOSなどのすべての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る。

Linux Mint.

他にも、**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パッケージ管理システムがあります。

もう一つの古いシステムは、実際の管理やクリーン・リムーバルの戦略なしに、パッケージを**tarボール**として提供することである。このアプローチは、最も古いLinuxディストリビューションの一つである**Slackware**の特徴である。

しかし、たいていの場合はRPMか**dpkg**のどちらかであり、このコースで検討するのはこれらだけである。

7.5 パッケージングツールのレベルと種類

パッケージングツールのレベルと種類

- **低レベルのユーティリティ:**

単一のパッケージ、または指定され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。
- 別のパッケージをインストールする必要がある場合、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とdpkg

- **ハイレベル・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する
- すでにインストールされているパッケージとの競合を避ける

- 例: dnf、yum、zypper、PackageKit、apt

包装システムには2つのレベルがある:

1. **低レベルユーティリティ:** このユーティリティは、単に単一のパッケージ、あるいは各パッケージが個々に 指定され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。依存関係は完全には扱われず、警告が出るかエラーが出るだけです。

- 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。

- その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、削除は失敗する。

rpmユーティリティと**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に対して、この役割を果たす。

2. ハイレベル・ユーティリティ：これは依存関係の問題を解決する：

- 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、そのようなニーズは満たされる。
- 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を選択肢が与えられます。

dnfユーティリティと**zypper**ユーティリティ（および古い**yum**）が**rpm**システムの依存解決を行い、**apt**と**apt-cache**やその他のユーティリティが、**dpkgシステム**のためにそれを処理してくれる。

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フロントエンドも便利ではあるが、私たちは1つのインターフェースに縛られることなく、より柔軟性を持ちたいと考えている。

7.6 パッケージソース

パッケージソース

- ディストリビューションには複数のリポジトリがある
- すべてのパッケージがうまく機能すること
- 外部リポジトリも存在する
 - いくつかのインターフェイスはきれいに
 - その他、コンフリクトの原因

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アップデートしたりする場所となっています。ディストリビューションの仕事は、リポジトリ内のすべてのパッケージが互いにうまく動作するようにすることです。

標準のディストリビューションがサポートする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によく適合します**。

しかし、外部リポジトリの中には、構築や保守があまりうまくいっていないものもあります。たとえば、あるパッケージがメインのリポジトリで更新されたとき、依存するパッケージが外部のリポジトリで更新されないことがあります。

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

- カスタムソフトウェア用のインストールしやすいパッケージを作成する
- パッケージでは、インストール時に追加タスクを実行できる。
 - 必要なシンボリックリンクの作成
 - 必要に応じてディレクトリを作成する
 - パーMISSIONの設定
 - スクリプトが可能なもの
- 多くのフォーマットが利用可能
 - **Red Hat**および**SUSEベースのシステム用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**のほとんどすべてのバージョンに、このためのメカニズムがあります。

独自のソフトウェアをビルドすることで、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。パッケージを作成し、それをインストールすることで、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を実行するようにできます。

7.8 リビジョン管理システム

なぜリビジョン管理システムを使うのか？

- プロジェクトの複雑さは、規模や貢献者、ユーザー、テスターの数によって増大する。
- リビジョン管理システム（バージョン管理システム）：
 - 変更履歴（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
- 多くの方法が使われている

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、管理が複雑になっていく。

更新を整理し、協力を促進するために、ソース管理にはさまざまな方式が利用できる。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する可能性のある更新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果たす。

7.9 利用可能なソース管理システム

利用可能なソース管理システム

製品	URL: 利用可能なソース管理システム
RCS	https://www.gnu.org/software/rcs/
CVS	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アーチ	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

プロプライエタリなものからオープンなものまで、利用可能な製品には事欠かない。

ライセンスを取得した。

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製品であるgitにのみ焦点を当てます。gitは、驚くほど短期間でオープンソースプロジェクトで使われるようになり、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る：

- 世界中に広く分布
- 数千人の開発者
- **GPL**ライセンスの下、非常に公開されている



- **git**はそのようなニーズを満たすために生まれ、**Linux**の効率的な成長を助けた。

gitとその使い方については、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のかについて、実習を交えて説明します。

Linuxカーネル開発システムには、世界中に広く分布し、文字通り何千人もの開発者が関わっているという特別なニーズがある。さらに、それはすべて**GPL**ライセンスのもとで非常に公に行われている。

長い間、本当のソース・リビジョン管理システムは存在しなかった。その後、主要なカーネル開発者は**BitKeeper** (<https://www.bitkeeper.com>) を使用するようになった。**BitKeeper**は**Linux**カーネル開発に制限付き使用ライセンスを与える商用プロジェクトである。

しかし、2005年春のライセンス制限をめぐる公然の論争で、**BitKeeper**の無償利用は**Linux**カーネル開発では利用できなくなった。

その対応策として開発されたのが**git**であり、その原著者はリーナス・トーバルズである。**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/>にある。

gitの仕組み

- 従来のバージョン管理システムではない
- **ファイル**を基本単位として動作しない
- **オブジェクト・データベース**には以下が含まれる：
 - **blob**
 - **樹木**
 - **コミット**
- **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キャプチャする。
- **グラフィカル・インターフェース**を持つ
- **github**のようなサイトには、何百万もの**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**です。

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る：

- **blob**：ファイルの内容を含むバイナリデータの塊。
- **ツリー**：ディレクトリ構造を示す、ファイル名と属性を含むblobの集合。
- **コミット**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・ツリーの状態をキャプチャする。

ファイル単位のシステムからコントロール・システムを解放することで、多くのファイルを含むチェンジセットを扱うことができるようになる。

gitは常に急速に開発されており、そのグラフィカル・インターフェースもまた、急速に構築されている。例えば、

<https://www.kernel.org/git/>。ソースツリーだけでなく、特定の変更も簡単に閲覧することができます。

<https://www.github.com> のようなサイトは現在、パブリックとプライベートの両方で、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**を有益に使う方法に関する記事や書籍、オンラインチュートリアルなどが簡単に見つかります。

7.11 ラボ



ビデオ・デモンストレーション・リソース

パッケージ管理デモ.mp4

演習7.1: gitによるバージョン管理



gitについては後のセクションで詳しく説明します。ここでは、git の基本的な使い方を説明します。

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だ。そうでない場合は、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るかもしれませんが、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 git* をインストールします。 # Fedora / RHEL 8 / CentOS 8
sudo yum install git*          # RHEL 7 / CentOS 7
```

特定のディストリビューションに従ってください。

gitの仕組みと使いやすさを感じてみましょう。とりあえず、自分のローカルプロジェクトを作ることしましょう。

1. まず作業ディレクトリを作成し、gitを初期化する:

```
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。このディレクトリの初期内容は次のようになります:

```
ls -l .git
```

```
合計40
drwxrwxr-x 2 coop coop 4096 Dec 30 13:59 branches/
-rw-rw-r-- 1 coop coop   92 Dec 30 13:59 config
-rw-rw-r-- 1 coop coop   58 12月30日13:59 説明
-rw-rw-r-- 1 coop クープ 23 12月 30 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/
drwxrwxr-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で説明する。

- 次にファイルを作成し、プロジェクトに追加する：

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

- プロジェクトの現在のステータスを見ることができる：

```
git status
```

ブランチ上の

master 最初のコミット

ット

コミットすべき変更

(ステージを解除するには "git rm --cached

<file>..." を使う) 新しいファイル:

somejunkfile

このファイルはステージングされているが、まだコミットされていない。

5. このリポジトリの責任者を **git** に伝えましょう:

```
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・コンフィギュレーション・ファイルであらかじめ定義していない限り、新しいプロジェクトごとにを行う必要がある。

6. では、ファイルを修正し、差分の履歴を見てみよう:

```
$ echo 別の行 >> somejunkfile
git diff
```

```
diff --git a/somejunkfile b/somejunkfile index
9638122..6023331 100644
--- a/somejunkfile
+++ b/somejunkfile
@@ -1 +1,2 @@
   ガラクタ
+もう一行
```

7. 変更を実際にリポジトリにコミットするには、こうする:

```
git commit -m "私の初期コミット"
```

```
最初のコミットを作成 eafad66: 私の最初のコミット
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添付する識別メッセージを指定しないと、エディターにジャンプして内容を入力すること

になる。そうしないとコミットが拒否されます。選択されるエディタはEDITOR環境変数に設定されているもので、GIT_EDITORを設定することで置き換えることができます。

8. 履歴を見ることができる:

`git log`

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

最初のコミット

そこに情報が入っているのがわかるでしょう。16進数の長い文字列が**コミット番号**であることに気づきでしょう。これは160ビット、40桁のユニークな識別子です。

9. これで、すでに終了しているファイルを自由に変更したり、`git add` で新しいファイルを追加したりできるようになりました。しかし、これらは別の `git commit` を行うまでステージされます。
10. 今はそれほど悪くない。でも、まだ表面しか見ていない。

第8章

dpkg



8.1	DPKG (Debian パッケージ).....	136
8.2	パッケージのファイル名とソース	137
8.3	DPKGクエリ	138
8.4	インストール/アップグレード/アンインストール	139
8.5	研究室.....	140

学習目標

このセッションを終えるまでに、あなたは次のことができるようになるはずだ：

- Debian パッケージングシステム (**DPKG**) の方法と使用方法について説明する。
- バイナリとソースの **.deb** ファイルに使用される命名規則について説明してください。
- ソースパッケージがどのようなものかを知る。
- パッケージに対するクエリおよび検証操作を使用する。
- **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。

135

8.1 DPKG (Debian パッケージ)

DPKGエッセンシャルズ

- **Ubuntu**および以下を含む、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る。

リナックスミント

- **rpm** のように、**dpkg** ユーティリティは、パッケージの取得や依存性の不具合の修正は行いません。
- パッケージファイルの接尾辞は **.deb**
- **DPKG**データベースは/var/lib/dpkgにある。

DPKG (Debianパッケージ)は、**Debian Linux**およびDebian Linux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ングシステムです。**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ためではなく、ローカルにインストールしたり削除したりするため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ディレクトリに存在します。

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のしか知らないが、システム上の他のディレクトリにあらうと、インターネット上にあらうと、他の利用可能なパッケージについては何も知らない。そのため、依存関係が満たされていない場合や、他のインストール済み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します。

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つのファイルで構成されています:

- アップストリームの tarball (末尾は .tar.gz)
- 説明ファイル (末尾は.dsc)
- 2番目の tarball (末尾は .diff.gz または .debian.tar.gz) には

Debian パッケージファイル名は、特定の情報を表すフィールドに基づいています。

、アップストリームソースへのパッチと、そのパッケージ用に作
歴史的な理由から、64ビットのx86プラットフォームはx86 64ではなくamd64と呼ばれ、Ubuntuのようなディストリビューターはパッケージ名にamd64が追加されます。

Ubuntuシステム上では、ソースパッケージをダウンロードし、どのファイルがダウンロードされ、どのファイルが作成されたかを見ることができる:

```
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・パッケージの入手

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リストする:

```
dpkg -l
```

- wgetパッケージ内のファイルをリストアップする:

```
dpkg -L wget
```

- インストールされたパッケージとパッケージファイルに関する情報を表示します:

```
dpkg -s wget
```

```
$ dpkg -I 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完全性を確認する

```
dpkg -V パッケージ
```

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は、しばしば重要である。

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場合などです:

```
dpkg -s dpkg | grep -i version
```

```
バージョン: 1.19.7ubuntu3.2
```

Ubuntu 20.04上で。

引数を追加しないと、dpkg -Vはシステム上の全パッケージを検証する:

```
sudo dpkg -V
```

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

出力の解釈については、**マニュアルページ**を参照のこと。

ール

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/アンインストール

- **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
```

- インストールされたパッケージを、設定ファイルを含めてすべて削除する。

```
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:

- そのパッケージが現在インストールされていない場合は、インストールされる。
- 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新しければ、アップグレードされます。

Pオプションは**パーズ**を意味する。

8.5 ラボ



ビデオ・デモンストレーション・リソース

[dpkgデモ.mp4](#)

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のための簡単な操作をいくつか行うだけです。

1. [etc/logrotate.conf](#)ファイルがどのパッケージに属するかを調べる。
2. パッケージに含まれるすべてのファイルを含む、パッケージに関する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. パッケージを取り出してみる。

✓解答 8.1

1. `dpkg -S logrotate.conf`

```
logrotate: /usr/share/man/man5/logrotate.conf.5.gz
logrotate: /etc/logrotate.conf
rsync: /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:

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。

第9章

自動プログラムツール



9.1	APT	142
9.2	APT ユーティリティーズ	143
9.3	クエリー	144
9.4	パッケージのインストール/移動/アップグレード	145
9.5	清掃	146
9.6	研究室	147

学習目標

このセッションを終えるまでに、あなたは次のことができるようになるはずだ:

- APTとは何かを説明せよ。
- 主な**APT**ユーティリティ・プログラムを説明する。
- パッケージクエリを実行する。
- 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 自動プログラムツール

APTとは何か？

- 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（下位レベルの**dpkg**プログラムを呼び出す）を含む。
- **Debian** パッケージ (.deb ファイル拡張子) で動作
- 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**を使う必要はない。しかし、習慣的に**apt-get**がいつも使われているのを見かけるで

しょうし、スクリプトでは**apt-get**の方がうまく機能します。このコ
Debianベースのシステムで使用する場合、**APT** (Advanced Packaging Tool) プログラムセットは、基盤となる**dpkg**プロ
グラムを使用するため、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
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、更新、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、それらと同期して、必要に応じ
てソフトウェアを取得・インストールします。

もう一度言うが、**synaptic**や**Ubuntu Software Center**のような（コンピューター上の）グラフィカル・インターフェース
や、**aptitude**のような**APT**の古いフロントエンドは無視する。

ティ

9.2 APTユーティリティ

apt、apt-get、apt-cacheなど。

- **Debian Linux** 用パッケージを扱うコマンドラインツール
 - 個別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
 - アップグレード配信
- **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できます。
- 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
- :

<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。個々のパッケージや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、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できます。

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で動作し

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`

- まず **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ない:

`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 パッケージに関する詳細情報を表示します。

`apt-cache depends apache2`

apache2 に依存しているパッケージをすべて表示します。

`apt-file search apache2.conf`

リポジトリから **apache2.conf** という名前のファイルを探してください。

`apt-file list apache2`

apache2 パッケージの全ファイルをリストする。
ディ

9.4 パッケージのインストール／削除／アップグレード

パッケージのインストール／削除／アップグレード

```
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、システムやパッケージを**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なし

dnfやzypperとは異なり、個々のパッケージをアップデート/アップグレードすることはありません。

`sudo apt install [パッケージ]`

新しいパッケージのインストールや、すでにインストールされているパッケージのアップデートに使用する。

`sudo apt remove [package]`

システムからパッケージを削除するときに使用する。設定ファイルは削除されません。

`sudo apt --purge remove [パッケージ]`.

パッケージとその設定ファイルをシステムから削除するのに使用する。

`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る。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
```

dist-upgradeは、一般的に誤解されているような、まったく新しいディストリビューションバージョンへのアップデートは行いません。

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**必要があることに注意してください。

9.5 後片付け

後片付け

- 古い **Linux** など、不要になったパッケージを削除する。
カーネルのバージョン：
`sudo apt autoremove`
 - インストール済みのキャッシュファイルやアーカイブされたパッケージファイルを削除する：
`sudo apt clean`
- これでスペースを大幅に節約できる。

`apt clean`と`apt autoremove`を使えば、ディスクの空き容量が少ないときに、`/var`のディスクスペースを素早く、かなり安全に確保できる。

9.6 ラボ



ビデオ・デモンストレーション・リソース

[ubuntu_demo.mp4](#)(パッケージ版)

練習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: 基本的なAPTコマンド

1. お使いのシステムに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する。
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。単純に
`apt-cache pkgnames`
ワイルドカード引数を与えてリストを絞り込むこともできる。

✓解答 9.1

1. まず、パッケージ・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る:

```
sudo apt update
```

実際にアップグレードすること:

```
sudo apt upgrade
sudo apt -u upgrade
```

(前述したようにdist-upgradeを使うこともできる。) 最初のフォームだけがインストールを試みる。

2. `apt-cache search "kernel"`
`apt-cache search -n "kernel"`
`apt-cache pkgnames "kernel"`

2番目と3番目の形式は、名前にkernelを含むパッケージのみを検索する。

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

でインストールされているパッケージのみを取得できます。**Debianベース**のシステムでは、カーネル関連のパッケージは通常名前にkernelがつかないため、おそらくkernelではなくlinuxを使うべきであることに注意してください。

3. `sudo apt install apache2-dev`

✍️ 練習9.2 APTを使ってパッケージの情報を探す

`apt-cache`と`apt`（`dpkg`ではなく）を使って、見つける：

1. 名前または説明に **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、利用可能な**bash**パッケージ。

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

上記で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。何か違いはありますか？

✓解答 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`

✍練習9.3: APTによるパッケージのグループ管理

APTは、メタパッケージを使うことで、dnfと同じようにパッケージのグループを管理する機能を提供する。メタパッケージは、インストールや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**のリストを取得する：

`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
ッケージ compiz - OpenGL ウィンドウと合成マネージャ
emacs - GNU Emacs エディタ (メタパッケージ)
....

```

そうすれば、通常の単一パッケージのように簡単にインストールできる：

`sudo apt install bacula-client`

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

状態情報の読み込み...完了

以下の追加パッケージがインストールされる:

bacula-common bacula-console bacula-fd bacula-traymonitor

推奨パッケージ:

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。742キロバイトのアーカイブが必要。

この操作の後、1,965kBの追加ディスク容量が使用されます。続行

しますか? [Y/n]

アンインストールされたメタパッケージを選択し、削除する。

第10章

回転数



10.1	RPM (Red Hat Package Manager).....	150
10.2	パッケージのファイル名	151
10.3	RPM データベースとヘルパープログラム.....	152
10.4	クエリー	154
10.5	パッケージの検証.....	155
10.6	パッケージのインストールと削除.....	157
10.7	RPMパッケージの更新、アップグレード、およびフレッシュ化.....	159
10.8	Linuxカーネルのアップグレード	161
10.9	rpm2archive と rpm2cpio	162
10.10	研究室.....	163

学習目標

このセッションを終えるまでに、あなたは次のことができるようになるはずだ:

- RPMシステムがどのように構成されているかを理解し、**RPM**プログラムがどのような主要業務を遂行できるかを理解する。
- バイナリとソース **rpm** ファイルの命名規則について説明する。
- パッケージのクエリ、検証、インストール、アンインストール、アップグレード、リフレッシュ。
- 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。
- **rpm2cpio** を使って、パッケージ化されたファイルを **cpio** アーカイブにコピーしたり、インストールせずにファイルを展開したりできます。

149

10.1 RPM (Red Hat Package Manager)

RPMを使用する利点

- ソフトウェアパッケージの管理
 - 単一のタスクまたはサブシステムに関連するファイルのコレクション
 - プログラム、データ、ドキュメンテーション、コンフィギュレーション情報が含まれています。
 - 依存情報も含まれる場合がある
 - パッケージを取得しない
- 管理者にとっての利点
 - パッケージのインストールとアンインストール（消去）が簡単
 - 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**を使ってパッケージをインストールできる
- デベロッパーにとってのメリット
 - オリジナルの "原始的な" ソースを使用可能
 - 異なるアーキテクチャ用のRPMを生成できる

RPMはRed Hatが開発したパッケージ管理ユーティリティである。(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、ファイルのインストールとアンインストールの方法と場所に関する情報も含まれています。開発者がプログラムの新バージョンを作成する場合、通常は新しい **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できます。また、ディスクスペースを空けるために、完全なパッケージを削除するのも簡単です。**RPM** はまた、文書ファイルをパッケージの他の部分と区別し、システムに文書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する。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コードを変更しなければなりません。**RPM**を使用すると、ビルダーは、**Linux**上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。この機能により、ビルドに関連する変更がすべて1か所に集約されるため、新しいバージョンのコードの組み込みが容易になります。また、異なるアーキテクチャ用の**Linux**バージョンのビルドも容易になります。



ネットワークからのインストール

特定の URL を指定しない限り、**rpm** はネットワーク経由でパッケージを取得しない。

しかし、HTTPのような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るし、非推奨の**FTP**。

10.2 パッケージファイル名

パッケージファイル名

- 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 (<http://www.rpm.org/>)

dnf, yum, zypper は RPM よりも上のレベルで動作します。

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

- RPM データベースは `/var/lib/rpm` にある。
- **Berkeley DB**ハッシュファイルとして保存された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
- データベースは再構築できる：

`sudo rpm --rebuilddb`

`/var/lib/rpm`は、**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格納するデフォルトのシステムディレクトリである。データベース・ファイルは手動で変更してはならない。更新は**rpm**プログラムを使ってのみ行うべきである。

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る。例えば、他のシステムからコピーした **RPM** データベースを検査する場合などである。

`rebuilddb`オプションを使えば、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できる。

RPMが使用するヘルパープログラムとスクリプトは、`/usr/lib/rpm`にある。例えば、**RHEL 8**システムには、かなりの数があります：

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

```
74
```

ここで`wc`は出力の行数を示す。

`rpmrc` ファイルを作成して、**rpm** のデフォルト設定を指定することができる。デフォルトでは

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

を上記の順序で実行する。これらのファイルはすべて読み込まれることに注意すること。rpmは、ファイルが存在するとわかったからといってすぐに停止するわけではない。代替の `rpmrc` ファイルは `--rcfile` オプションで指定できます。

10.4 クエリ

クエリ

- すべての **rpm** クエリには **-q** オプションがあります。
- **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る：
 - f: ファイルがどのパッケージから来たかを知ることができます。
 - l: 特定のパッケージの内容をリストアップする。
 - 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 クエリ・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ているか?	rpm -q bash
このファイルはどのパッケージから来たのですか?	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する。	rpm -qi bash
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このパッケージに関する情報を表示する。	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リストする。	rpm -qa

他に便利なパラメーターとして--requiresと--whatprovidesがある。

requiresオプションは、パッケージの前提条件のリストを返し、--whatprovidesオプションは、インストールされたパッケージが特定の必要パッケージを提供するものを示す。

```
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、パッケージの検証に使用します。
- 何も問題がない場合は出力されない
 - 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パスの不一致
 - U: ユーザーの所有権が異なる
 - G: グループの所有権が異なる
 - T: mTimeが異なる
- すべてのパッケージをチェックする：

`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。

出力では（問題がある場合のみ出力が表示される）、各文字は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す。単一の"."（ピリオド）は、テストがパスしたことを意味し、単一の"?"（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する（例えば、ファイルのパーミッションが読み取りを妨げている）。それ以外の場合は、対応する--verifyテストの失敗を表す。

何も問題がなければ出力はない。

```
rpm -V bash
```

ファイルのサイズ、チェックサム、更新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、パッケージをインストールする際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` で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`
`i`はインストール、`-v`は冗長、`-h`は進行状況を示すハッシュマークを表示する。

依存性チェックが必要なのは、他のパッケージもインストールされていないと正しく動作しないパッケージがあるからだ。

コンフリクトには、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の上にインストールしようとしたりすることが含まれます。パッケージをビルドする人は、インストールの前後に特定の作業を行うよう指定できます。

設定ファイルをインストールするとき、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てから変更されている場合、**RPM**は古いバージョンを接尾辞`.rpmsave`をつけて保存します。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、所有者、変更（ビルド）時間などの属性も設定する。

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すると、通常は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。

- 依存関係によるエラー

```
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_3.x86_64 によって必要とされています。 xz

は (インストールされた) sos-3.9.1-6.el8.noarch に

よって必要とされています。

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ています。 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ています。

xz は (インストールされた) dracut-049-

95.git20200804.el8_3.4.x86_64 によって必要とされます。

↳ libvirt-daemon-driver-qemu-6.0.0-28.module+el8.3.0+7827+5e65edd7.x86_64 xz

は (インストールされた) rpm-build-4.14.3-4.el8.x86_64 によって必要とされています。

xzは(インストールされている)必要がある

↳ libguestfs-1:1.40.2-25.module+el8.3.0+7421+642fe24f.x86_64 xz

は (インストールされた) sysstat-11.7.3-5.el8.x86_64 によっ

て必要とされています。

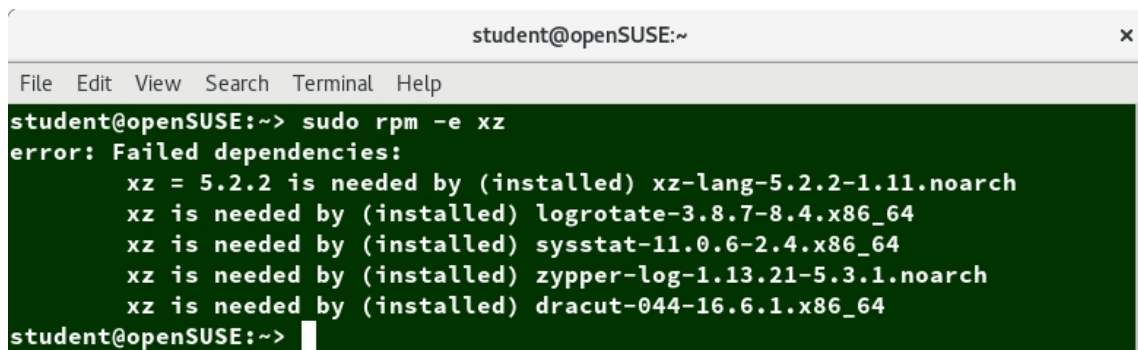
/usr/bin/xz は (インストールされた) file-roller-3.28.1-3.el8.x86_64 によって必要とされています。

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) させます。通常、アンインストールしようとしてい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出して失敗します。

e とともに --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しなくても、アンインストールが成功するか失敗するかを判断できます。操作が成功した場合、rpm は何も出力しません。より詳細な情報を得るには、-vv オプションを追加します。

消去するパッケージの引数は、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こと。

重要な注意: rpm パッケージ自体は絶対に削除 (消去/アンインストール) 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ため、依存関係が多少異なることに注意してください。

10.7 RPMパッケージの更新、アップグレード、フレッシュ化

パッケージの更新

- インストールされている場合、アップデートは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、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できない。
- `oldpackage`オプションでダウングレード可能

アップグレードの際、既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 `.rpmsave` という拡張子で保存されます。

Uオプションを使用し、パッケージがまだインストールされていない場合、単にインストールされ、エラーは発生しない。

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と、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出て失敗します。

しかし、各バージョンのパッケージに同じファイルが含まれていなければ、同じパッケージの異なるバージョンがインストールされるかもしれません。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、次のコマンドを追加する必要があります。

`--oldpackage`オプションをコマンドラインに追加する。

リフレッシュ・パッケージ

- 現在のディレクトリにあるすべてのパッケージをリフレッシュする:

```
sudo rpm -Fvh *.rpm
```

- 旧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る。
- 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、何も起こらない。
- インストールされているパッケージのバージョンがない場合、ディレクトリにあるパッケージは無視される。

- フレッシュニングは、多くのパッチ（つまりアップグレードされたパッケージ）を一度に適用するのに便利です。

リフレッシュはアップグレードと似ていますが、1つ例外があります。アップグレードするとき、そのパッケージがまだ存在しなければ、ロードされます。リフレッシュする場合、パッケージがまだロードされていなければ、単に無視されます。アップグレードもフレッシュ化も、元のパッケージがすでにロードされていれば、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、新しいパッチをいくつかダウンロードし、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たいが、新しいパッチをインストールしたくない場合に便利である。

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

- **Red Hat ベースのシステム**では、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）：

```
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は可能だが、動作が確認されている古いカーネルを少なくとも1つ残しておくことは常に良いアイデアである。

もし本当に古いバージョンのカーネルやその他のパックを削除したいのであれば、**Red Hatベースのシステム**ではそれが可能だ：

```
sudo dnf remove --oldinstallonly
```

このコマンドの使用には注意が必要だ。

システムに新しいカーネルをインストールすると、それを有効に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、現在動作している古いカーネルを削除してしまいます。

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、古いカーネルで再起動することはできなくなります。しかし、(-i)をインストールすると、両方のカーネルが共存するようになり、どちらか一方を選んで起動できるようになります。

これを行うと、**GRUB**設定ファイルは自動的に新しいバージョンを含むように更新され、システムを再設定しない限り、ブート時のデフォルトの選択肢となる。

新バージョンのカーネルがテストされたら、必要でなければ旧バージョンを削除してもかまいません。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な状態にしておくことをお勧めします。

10.9 rpm2archiveとrpm2cpio

rpm2archiveとrpm2cpioを使う

- 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
```

`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します;

```
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される。引数として `-` を与えると、入出力は標準入力と標準出力。

これは、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したりするために使用できる、古い **rpm2cpio** ユーティリティよりも直接的で効率的です。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展開されます。したがって、ユーザーが `/home/student` ディレクトリにいて、上の例のように `bin/bash` ファイルを展開すると、`/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ある:

```
rpm -qilp package.rpm
```


10.10 ラボ



ビデオ・デモンストレーション・リソース

[rpmを使ったデモ.mp4](#)

練習10.0: 必要な回転数



Red Hat、Centos、Fedora、openSUSEの場合

これらのラボを行うには、**RHEL**、**CentOS**、**Fedora**、**SUSE**、または以下のような**RPMベース**のシステムにアクセスする必要があります。

openSUSE。

練習問題 10.1: RPMの使用

ここでは、**rpm**パッケージのクエリーと検証のための簡単な操作をいくつか行うだけです。

。このラボは、**Red Hat** と **SUSE** ベースのシステムで同じように動作します。

1. `etc/logrotate.conf`ファイルがどのパッケージに属するかを調べる。
2. パッケージに含まれるすべてのファイルを含む、パッケージに関する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. パッケージを取り出してみる。

解答 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正式な形はこうなる：

```
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:

```
sudo rpm -e logrotate
```



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にが必要です。

logrotate >= 3.5.2 は、rsyslog-8.37.0-13.el8.x86_64 にが必要です。



openSUSEの場合

openSUSE-Leap 15.1で:

```
sudo rpm -e logrotate
```

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 で必要とされています

。 logrotate は (installed) syslog-service-2.0-lp151.3.3.noarch で必要と

されています。 logrotate は (installed) vsftpd-3.0.3-lp151.6.3.x86_64

で必要とされています。

(installed) libvirt-daemon-5.1.0-lp151.7.6.1.x86_64 logrotate が必要です

(installed) iscsiui-0.7.8.2-lp151.13.6.1.x86_64 logrotate が必要です

(installed) mcelog-1.60-lp151.2.3.1.x86_64

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する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。

演習10.2: RPMデータベースの再構築

`/var/lib/rpm`に格納されている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、新しいデータベースを作成し、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じように動作する。

1. リビルド処理によって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。これを怠って何か問題が起きたら、大変なことになる。
2. データベースを再構築する。
3. ディレクトリの新しいコンテンツとバックアップされたコンテンツを比較する。バイナリデータなので実際のファイルコンテンツは調べないが、ファイル数とファイル名はメモしておく。
4. システムの全回転数のリストを取得します。このリストと、実際に再構築手順を実行する前に生成されたリストを比較するとよいでしょう。クエリーコマンドが動作していれば、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内容をもう一度比べてみてください。同じファイルがありますか？
6. バックアップを削除することもできますが（おそらくサイズは100MB程度）、システムが正常に動作していることを確認するために、バックアップを破棄する前にしばらく残しておいた方がよいでしょう。

✔ 解答 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、この作業はすべきではないだろう！
`sudo rm -rf rpm_BACKUP`

第11章

dnfとyum



11.1	dnf	168
11.2	うん	169
11.3	クエリ	170
11.4	パッケージのインストール/移動/アップグレード	171
11.5	追加の dnf コマンド	172
11.6	研究室	173

学習目標

このセッションを終えるまでに、あなたは次のことができるようになるはずだ：

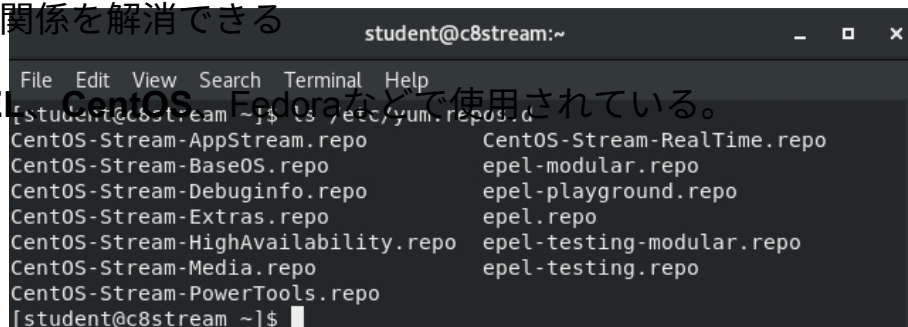
- パッケージインストーラとその特徴について説明する。
 - 高レベルのパッケージ管理システムとしてdnfがどのように機能するかを説明する。
 - リポジトリを使用するように**dnf**を設定する。
 - **dnf**が使用できるクエリーについて説明する。
-
- **dnf**を使用して、パッケージの検証、インストール、削除、アップグレードを行います。
 - 追加コマンドと新しいリポジトリのインストール方法について説明します。

167

11.1 ディーンフ

dnfとは?

- 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フェッチできる
 - 複数の独立したリポジトリに依存できる
 - `etc/yum.repos.d/`の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る。



```

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には、パッケージ管理に便利な多くの機能がある。**RPM**のフロントエンドであると同時に、1つ以上のリモートリポジトリからパッケージを取得する機能もある。最も優れた機能の一つは、依存関係を解決する機能である。

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、拡張子は`.repo`です。非常に単純なリポ



サンプルレポ

[レポ名]

```

name= リポジトリの説明
baseurl=http://somesystem.com/path/to/repo
enabled=1
gpgcheck=1
  
```

ジトリファイルは次のようになります:

dnfは、パフォーマンスを高速化するために情報とデータベースをキャッシュする。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、次のコマンドを実行する：

```
dnf clean [ packages | metadata | expire-cache | rpmdb | plugins | all ]。
```

enabledの値を変更するか、-disablerepo somerepoを使うことで、特定のレポの使用のオン・オフを切り替えることができる。と--enablerepo somerepoオプションがある。

ここでは、**dnf**のコマンドラインでの使用に集中し、ディストリビューションが提供するグラフィカル・インターフェースについては考慮しない。

11.2 うん

うん

- **dnf** は RHEL/CentOS 7 から 8 への移行時に **yum** に取って代わった。
- **Fedora**はもっと前から使っている
- dnfは後方互換性がある：
 - ほとんどすべての一般的な**yum**コマンドはまだ動作する
- 詳細は <https://docs.fedoraproject.org/en-US/quick-docs/dnf/> <https://dnf.readthedocs.io/en/latest/> を参照

。

yumコマンドを実行しようとする、**dnf**のバージョンによっては、これらのコマンドが非推奨であることを警告し、正しいコマンドを指示してくれる。

経験豊富な**yum**ユーザーであれば、**dnf**は日常的なタスクの大半を処理する**yum**コマンドのサブセットを受け付けるので、徐々に使い方を学ぶことができる。

11.3 クエリ

クエリ

- パッケージの検索に使用可能

`dnf search`

`dnf info`

`dnf リスト`

`dnf grouplist`

`dnf groupinfo`

- ファイル検索

`dnf provides`

`dnf` 検索キーワード

キーワードを含むパッケージを検索する。

`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 [installed | updates | available].`

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、アップデートを一覧表示します。

`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、アップデートに関する情報を表示します。

`dnf groupinfo packagegroup`

パッケージグループの情報を表示する。

`dnf provides /path/to/file`

ファイルの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 / を使う必要があることに注意。紛らわしいかもしれません)。

11.4 パッケージのインストール／削除／アップグレード

パッケージのインストール／削除／アップグレード

- コマンド:

```
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。依存関係の解決とインストールも行います。

`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する。

`sudo dnf groupinstall 'グループ名'`

また、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する。

`sudo dnf update [パッケージ].`

リポジトリからパッケージを更新します。表示されているパッケージがない場合は、すべてのパッケージを更新します。

インストール中 (またはアップデート中)、パッケージに更新された設定ファイルがある場合、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、新しい設定ファイル名は .rpmnew という拡張子になります。これらのファイル名の拡張子を検索して、ファイルを調整する必要があるかどうかを確認することができます。

11.5 その他のdnfコマンド

その他のdnf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].
```

```
sudo dnf list "dnf-plugin*"
```

その他の**dnf**プラグインのリスト

```
sudo dnf repolist
```

は有効なリポジトリのリストを表示します。

```
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する。2番目の形式は、**file.txt**のコマンドを実行する。

```
sudo dnf install --downloadonly パッケージ
```

パッケージだけをダウンロードしてくれる。パッケージは**var/cache/dnf**ディレクトリに保存される。

```
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できます。

```
sudo dnf clean [packages|metadata|expire-cache|rpmdb|plugins|all].
```

var/cache/dnfにローカルに保存されたファイルとメタデータをクリーンアップする。これはスペースを節約し、古くなったデータのハウスクリーニングを行う。

11.6 ラボ

✍️ 実習11.0: dnfまたはyumが必要



Red Hat、Centos、Fedoraの場合

これらのラボを行うには、RHEL/CentOSやFedoraなどの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dnfの代わりにyumを使用することができます。

✍️ 練習問題 11.1: 基本的なdnfコマンド

1. お使いのシステムに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する。
4. httpd-devel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。単純に

```
sudo dnf list
```

ワイルドカード引数を与えてリストを絞り込むこともできる。

✅ 解答 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試みる。

2.

```
sudo dnf update bash
```
3.

```
sudo dnf list installed "kernel*"
sudo dnf list "kernel*"
```
4.

```
sudo dnf install httpd-devel
```

✍️ 練習問題 11.2: パッケージについての情報を見つける

rpmではなく) dnfを使って、検索してください:

1. 名前または説明に **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、利用可能な**bash**パッケージ。
3. **bash**のパッケージ情報。
4. **bash**パッケージの依存関係。

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。何か違いはありますか？

✔解答 11.2

**ご注意**

ディストリビューションのバージョンによっては、以下のコマンドで**sudo**を使用しないと、情報を得ただけなのにパーミッションエラーが発生することがある。

```
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**

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でもパーミッションエラーが発生することがある。

dnfはパッケージのグループを管理する機能を提供する。

1. 以下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・グループをリストアップする:

```
dnf grouplist
```

2. 仮想化ホストグループを特定し、コマンドを使用してこのグループの情報を生成します。

```
dnf groupinfo "仮想化ホスト"
```

3. を使ってインストールする:

```
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て不要なパッケージグループを特定する。dnf groupremoveを使って、以下のように削除する:

```
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る。

groupremoveがインストールされたものすべてを削除しないことに気づくかもしれない。これがバグなのか機能なのかは、議論してほしい。

練習問題 11.4: 新しいリポジトリの追加

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ウェブベースのインターフェースです。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できます。Webmin は、以下のような Unix 設定ファイルを手動で編集する必要性を取り除きます。

/etc/passwdで、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る。"

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します。単純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を行うことはできません。

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。このファイルには以下を記述します:



webmin.repo

[ウェブミン]

name=Webmin Distribution Neutral

baseurl=http://download.webmin.com/download/yum



```
mirror=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	178
12.2	クエリー	179
12.3	パッケージのインストール/移動/アップグレード	180
12.4	zypper の追加コマンド	181
12.5	研究室	182

学習目標

このセッションを終えるまでに、あなたは次のことができるようになるはずだ：

- 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てください。
- **zypper**を使用して、パッケージをインストール、削除、アップグレードします。
- さらに高度な**zypper**コマンドを学ぶ。

177

12.1 ザッパー

ザッパー

- **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
- 依存関係を解決する
- **RPM**パッケージと連動
- コマンドと機能は**dnf**に似ている。

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、より高度な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/yum**と同じ役割を果たします。パッケージのインストール、更新、削除時に依存関係を自動的に解決できます。外部の**ソフトウェア・リポジトリ**にアクセスし、それらと同期し、必要に応じて**ソフトウェア**を取得・インストールします。

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。以下とよく似ています。

dnfは、その機能と基本的なコマンド構文において、**rpm**パッケージでも動作する。

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リストを表示します。

`zypper repos`

は利用可能なリポジトリを一覧表示します。

`zypper search <文字列>`

文字列をリポジトリから検索する。

`zypper info firefox`

パッケージに関する情報をリストアップします。

`zypper search --provides /usr/bin/firefox`

はリポジトリを検索して、どのパッケージがファイルを提供しているかを表示する。

12.3 パッケージのインストール／削除／アップグレード

パッケージのインストール／削除／アップグレード

- 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する。

```
sudo zypper --non-interactive install firefox
```

はインストールやアップグレードの確認を求めません。これはスクリプトに便利です。

```
sudo zypper 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する。

```
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、確認を求めません。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する。

dnfと同様、削除コマンドには注意が必要だ。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からだ。

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ーフェイスに入る:
`sudo zypper shell`
- 新しいリポジトリを追加します:
`sudo zypper addrepo`
- リポジトリを削除します:
`sudo zypper removerepo`
- `var/cache/zypp` を整理してスペースを節約する:
`sudo zypper clean [--all]`

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の再読込を避けるために、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
> バッシュをインストールする
...
> 出口
```

zypperは**readline**ライブラリをサポートしているので、**bash**シェルで利用できるのと同じコマンドライン編集機能を、**zypper**シェルですべて利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
```

削除したいリポジトリのエイリアスを使用してください。

12.5 ラボ



ビデオ・デモンストレーション・リソース

ヤストデモ.mp4

🔗練習12.0: zypperが必要



openSUSEの場合

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

🔗演習12.1: 基本的なzypperコマンド

1. お使いのシステムに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. 有効かどうかにかかわらず、システムが認識しているすべてのリポジトリをリストします。
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する。
5. **apache2-devel**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ます。(SUSEではhttpdは**apache2**です。
システム)簡単なことだ:

```
sudo zypper search
```


ワイルドカード引数を与えてリストを絞り込むこともできる。

✅解答 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
```

演習12.2: zypperを使ってパッケージの情報を探す

直接rpmではなく) zypperを使って、find:

1. 名前または説明に **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、利用可能な**bash**パッケージ。
3. **bash**のパッケージ情報。
4. **bash**パッケージの依存関係。

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。何か違いはありますか？

🟢解答 12.2

1. `zypper search -d bash`

dオプションを指定しないと、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る。おそらく、**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、次のようにすることです。

```
sudo zypper remove --dry-run bash
```

この練習では、bashはシステムに不可欠なので、悪い選択だ。

第13章

GITの基礎



すべてのリビジョン管理システムの概念を説明し、特にオープンソースで最も人気のあるシステムであるgitを使用するための最初のステップを学びます。

ここでは、ソフトウェアの開発にgitを使う方法に焦点を当てるのではなく、システムソフトウェアやユーザーソフトウェアをインストールしたりアップデートしたりする目的で、（ソフトウェアパッケージのような）ファイルセットを参照する複数の方法に焦点を当てます。

13.1	リビジョン管理	186
13.2	基本コマンド	188
13.3	gitコマンドを試す	190
13.4	Gitでソフトウェアを入手する	196
13.5	研究室	202

学習目標

このセッションを終えるまでに、あなたは次のことができるようになるはずだ：

- なぜリビジョン管理システムが重要なのかを説明する。
- `clone`、`add`、`commit`、`tag`、`checkout`など、`git`で使われる主なコマンドを使いこなす。
- `git help`を使えば、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。
- システム設定ファイルのシンプルで直線的な追跡のために`git`を使用するように適応させることができるワークスルーと例。
- タグや**ブランチ**の典型的な使用方法を含め、ソフトウェアのアップストリームバージョンをダウンロードする方法について学びます。

13.1 リビジョン管理

リビジョン管理システムとは

 はリビジョン管理システム

リビジョン管理システムは、ある時点のファイルのバージョンを保存して、変更を追跡するために使われる：

- When - 時点を示すタイムスタンプ。
- What - その時点でのファイルのバージョン
- 誰 - 変更の記録に関連付けられたユーザー
- なぜ - 変更について知っておくと便利な追加情報は何か **RCS**のような

非常に初期のシステムでさえ、この情報を追跡していた。

この初期のシステムでは、変更はファイルごとに同じ名前のテキストファイルで追跡されたが、サブディレクトリ **RCS/** に **.v** が付加されていた。このシステムは、複数のファイルにまたがる変更をグループ化する機能を欠いており、ファイルのロックとストレージ効率に欠点があったが、上記の基本的な考え方により、これは依然として強力なツールであった。

リビジョン管理システムとは

 は**分散**リビジョン管理システムです。...

1ファイル1変更システムからの進化における次のステップは、複数の変更を同時に追跡できるようにすることと、**真実の**ソースを集中化されたネットワークサーバーにすることだった。**CVS**は、ファイルのチェンジセットを保存するための同じ基本的なメカニズムを使用したか、同じネットワークにアクセスできる開発者が一緒に作業できるようにした。残念なことに、排他的なアクセスや同じファイルに適用される変更の問題により、このシステムは時に困難を極めた。

分散リビジョン管理システムは、多くの貢献者（**Linux**カーネルの場合は数千人）を可能にする。

- 多くのプロジェクトでは、開発者は別々に作業しているが、共通のバージョン管理システムによってまとまっている。これは、**CVSのように**中央の権威あるリポジトリがある場合でも可能です。
- gitが異なるのは、技術的なレベルでは、中央の権威あるリポジトリというものが存在しないことである。ある特定の場所が果たす重要性や中心的な役割は、技術的なものではなく、政治的・社会的なものです。
- **分散開発とは**、プロジェクトのさまざまな部分が別々のホストリポジトリの間で分割されて別々に作業されることではありません。**gitでは**、分散開発とは、すべてのリポジトリが権威を持ち、プロジェクトの一部分だけでなくコードベース全体を含むことを意味します。
- gitには政治的な要素はなく、望ましい組織モデルもありません。開発コミュニティの階層は、非常に中央集権的でトップダウンの場合もあれば、非常にフラットな場合もあります。情報や変更のやり取りは、いくつもの開発ツリーが

トップレベルのものに合体するピラミッド型になることもあれば、非常に平等主義的になることもあります。

- gitはツールであり、厳格なメソッドではないので、プロジェクトのニーズや哲学に合った方法で使うことができる。

ル

13.2 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]`

`git clone`

`git branch`

`git status`

`git add`

`git commit`

`git merge`

`git pull`

`git push`

インストールされている`git`のバージョンが表示されます:

`git --バージョン`

```
git バージョン 2.30.2
```



内蔵ヘルプ

`git help`サブコマンドは、各`git`呼び出しの詳細なマニュアルページを提供します。

`git help`

`git help` コマンド

中核となるgitコマンド

作業領域を開始する	(git ヘルプチュートリアルも参照)
クロム	リポジトリを新しいディレクトリにクローンする
アルモ参照)	空のGitリポジトリを作成するか、既存のリポジトリを再クローンする
ーン	初期化する
現在の更新に取り組む	git help everyday も参照くだ
開始	
さい) add	ファイルの内容をインデックス
に追加する	
mv	ファイル、ディレクトリ、シンボリックリンクを移動またはリネームする。
リストア	作業ツリーファイルのリストア
bisect	作業ツリー検索を使ってバグを導入したものを追跡する
diff	コミット間の変更、コミットと作業ツリーなどの表示 パ
sparse-checkout	必要なサブモジュールの初期化と修正
ログ	タミットのログを表示する行の表示
履歴と状態を調べる	様々な種類のオブジェクトも参照)
ショー	を表示する 作業ツリーの状
共通の歴史を育て、マージ	名し、微調整する
ブス	ブランチの一覧表示、作成
テチ	リポジトリへの変更を記録する
削除	ブランチ上の開発履歴を結合する 別のベースチッ
タヌ	ブの上にコミットを再適用する 現在のHEAD
ット	を指定された状態にリセットする ブランチを
マー	切り替える
共同作業 (git ヘルプチュートリアルも参照ください)	別リポジトリからリモートブランチをプルしてローカルのブランチと統合する
ブランチ	削除、検証
ブロードする	別のリポジトリまたはローカルブランチから取得し、統合する
ぶり	関連オブジェクトとともにリモート参照を更新する
ゼッ	
サス	
イツ	
オタ	
ダ	

gitコマンドの基本的なリストは、**git**と入力すれば上の写真のように表示される。

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--が付いたものだけである。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]に含まれている。



もっと知りたい？

公式リファレンスでは、サブコマンドや環境変数などの完全なリストを、特定のgitバージョンごとに見ることができます：

[https://git-scm.com/docs/git# gitコマンド](https://git-scm.com/docs/git#gitコマンド)

13.3 いくつかのgitコマンドを試す

最小限のグローバル設定

他の最近のリビジョン管理システムとは異なる、 をセットアップし現在のユーザーIDをデフォルトにするのではなく、**著者名**と**Eメールアドレス**を指定してください。

ここでは、よりモダンなデフォルトのブランチ名 *main* を追加して設定しましょう。

```
git config --global user.name "Gladys West"
git config --global user.email "gwest@example.com"
git config --global init.defaultBranch main
```

これらの設定は `~/ .gitconfig` に書き込まれます。

`~/ .gitconfig` ファイルには、さらに多くの内容を含めることができます。多くの人は、頻繁に入力するコマンドのショートカットや、履歴を印刷するカスタム方法、さまざまな**差分**コマンドを持っています。

みましょう。

リポジトリの作成と初期化

新しいディレクトリを作成し、gitリポジトリとして初期化する。

```
mkdir /tmp/LFgit
cd /tmp/LFgit
git init
```

空の Git リポジトリを /tmp/LFgit/.git/ に初期化した。

```
git status
```

main ブランチで

のコミット

何もコミットしない（ファイルを作成/コピーし、"git add "を使って追跡する）

これまで何が起こったかを検証してみよう：

```
ls -la /tmp/LFgit
```

合計12

```
drwxr-xr-x 3 user user 4096 May 9 18:12 .
drwxrwxrwt 16 root 4096 May 9 18:16 ....
drwxr-xr-x 7 user user 4096 May 9 18:15 .git
```

```
ls -la /tmp/LFgit/.git
```

```
ls -l /tmp/LFgit/.git/
total 32
drwxr-xr-x 2 user user 4096 May 9 18:12 branches
-rw-r--r-- 1 user user    92 5月 9 18:12 コンフィグ
-rw-r--r-- 1 user user    73 5月 9 18:12 説明
-rw-r--r-- 1 user user    21 5月 9 18:12 HEAD
drwxr-xr-x 2 user user 4096 May 9 18:12 hooks
drwxr-xr-x 2 user user 4096 May 9 18:12 info
drwxr-xr-x 4 user user 4096 May 9 18:12 objects
drwxr-xr-x 4 user user 4096 May 9 18:12 refs
```

リポジトリにファイルを追加する

`etc/motd`をリポジトリにコピーする。　小さくて、（うまくいけば）テキストしか入っていない。

(まだ/tmp/LFgitにある)

```
cp /etc/motd .  
git status -sb -uall
```

```
## main へのコミットはまだありません  
モットー
```

```
git add motd  
git status -sb -uall
```

```
## main A motd にはまだコミ  
ットがない
```

これで、ファイルがステージング・エリアに追加されたことが確認できた。

`git status` の呼び出し方が少し違うことに注目してください。

-sはショートフォーマットを表示

-b 短いフォーマットでもブランチ情報を表示する

-追跡されていないディレクトリを含む、追跡されていないファイルを表示する。

みましょう。

チェンジ・イット・アップ

ステージング・エリアに追加されたが、まだコミットされていないファイルを修正するつもりはない。それは重要ですか？

```
$ echo meow >> motd
git status -sb -uall
```

```
## メイン AM motd にはまだコ
ミットがない
```

ステータスが更新され、ファイルは「追加」されたままですが、「変更」されました。修正とは何ですか？

```
git diff
```

```
diff --git a/motd b/motd index
0c87dd3..c3ce8b3 100644
--- a/motd
+++ b/motd

usr/share/doc/*/copyright にある個々のファイル。
```

Debian GNU/Linux には、適用される法律で認められている限りにおいて、

まったく保証はありません。

```
+ニヤー
```

まだ一度もコミットしていないにもかかわらず、ファイルを追加し、それを修正し、違いを確認することができた。

すべての最新のリビジョン管理システムがこのように振る舞うわけではありません。例えば、*mercurial (hg)* はステージング・エリアをこのようには扱いません。

コミットする

ファイルをコミットして、ステータスをチェックしてください。どのバージョンのファイルがコミットされると思いますか？

```
git commit -m '今日のメッセージ'
git status -sb -uall
```

```
## main
M motd
```

これは、追加されたファイルのバージョンをステージングエリアにコミットしたものです。もし本当に "meow" を追加した現在のバージョンのファイルが欲しいのなら、もう一回 **git add** を行い、もう一回 **git commit** を行う必要があります。

あるいは、**git commit -all -amend** を使って、別のコミットを作成せずに一度に行うこともできます。

--(新規ファイルではなく)変更されたファイルや削除されたファイルを自動的にステージする。

--amend 最後のコミットを新しいコミットに置き換える

```
git commit --all --amend
```

git

コミットメッセージ

motdにクリティカルな「ニャー」を追加

- * 第4563回を閉じる：ニャンはどこへ行った？
- * 真剣にテストした結果、生産上の問題を引き起こしている単語の欠落を発見しました。

```
# 変更のコミットメッセージを入力してください。開始行
# の付いた#は無視され、空のメッセージはコミットを中止する。
```

正しくフォーマットされたコミットメッセージは、あなたの仕事にプロフェッショナルな体裁を与えるのに大いに役立ちます。コミットメッセージを作成する際には、以下のルールに従いましょう：

- 一番上の要約行は短く - 50文字程度にする。
- 現在の時制を使う - コミットが適用されたらどうなるか
- 行を飛ばしてインデントする。

みましょう。

州をチェック

我々の現状はどうか？

git status

```
ブランチ・メイン  
コミットするものはない。
```

git log

```
commit 8933f422d17d107d35b80f8a1c55b09dbec289fe (HEAD -> main)  
author: グラディス・ウェスト <gwest@example.com>  
日付   Fri May 9 20:01:53 2023 -0400
```

* 真剣にテストした結果、生産上の問題を引き起こしている
単語の欠落を発見しました。

```
commit 0872249ec1c9daeb60cb491fe6be7647cf606149 author  
: グラディス・ウェスト <gwest@example.com>  
日付   Fri May 9 19:31:03 2023 -0400
```

今日のメッセージ

これで作業ディレクトリはきれいになり、コミットもログに表示されるようになった。

git log コマンドには、相対時間、変更されたファイルの差分、通常は表示されない追加フィールドなど、多くの設定が用意されています。

コミットハッシュを短くし、1行でフィッティングするバージョンだ：

git log --pretty=oneline -n 20 --graph --abbrev-commit

```
* 8933f42 (HEAD -> main) critical 'meow' を motd に追加。  
* 0872249 本日のメッセージ
```

13.4 Gitでソフトウェアを手に入れる

ソフトウェアのダウンロード

ファイルの変更や追跡にはまったく興味がなく、ただソフトウェア・パッケージやコンフィグをダウンロードしたいだけで、`git`：

- これが、このソフトウェアを入手するための著者の推奨する方法である。
- 配布版よりも新しい、あるいは完全なものである。
- これは、ソースと変更点を見るための迅速な方法である。

`git`

全体 生態系 または ソフトウェア フォージ が
エコシステム を開発し、ウェブ・ブラウザだけで利用できるよ
うになった。

人気があるのは以下のようなものだ：

- ギットハブ
- ギットラボ
- コードバーク
- 作者または組織のセルフホスト・ソフトウェア・フォーゼ

通常、特定のリリース・バージョンのみ、ウェブ・ブラウザですぐにダウンロードできるが、ディスカッションに参加したり、問題を提出したり、ソース・コードを閲覧したりすることは、モバイル・ブラウザからでも可能なことが多い。

この方法でランダムなプロジェクトを見ることもできる。上記の特定のサイトはそれぞれ、ホスト名の最後に"/explore/"を追加することをサポートしている：

- <https://github.com/explore/>
- <https://gitlab.com/explore/>
- <https://codeberg.org/explore/>

Linuxカーネルの入手

自分自身のローカル・カーネル・コピーに**clone**サブコマンドを使ってみよう。

ソース

```
$ git clone --depth 1 -b master \
  https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git linux/
```

```
クローンを '/home/user/linux' に作成
中... リモートです: オブジェクトを列
挙しています: 85093, 完了。
リモートオブジェクトをカウント中: 100% (85093/85093),
完了: オブジェクトを圧縮: 100% (82795/82795)、完了。
リモート: 合計 85093 (デルタ 6421)、再使用 17237 (デルタ 1404)、パック再使
用 0 オブジェクトの受信: 100% (85093/85093), 237.53 MiB | 1.34 MiB/s, 完了。
デルタの解決: 100% (6421/6421)、完了。ファ
イルを更新: 100% (80335/80335)、完了。
```

大規模なプロジェクトの場合、ダウンロードにかかる時間や使用するディスク容量を減らす代わりに、履歴をすべて削除した最新の（または別の）

コミットだけを見たい場合は、いくつかの追加オプションを指定することができます。

これらのオプションを指定しないと、**git clone** はすべての履歴を含むデフォルトのブランチを取得します。

以下のオプションを**git clone**に渡しました。

- クローン作成時に指定したブランチを指す
- depth 指定したコミット数の浅いクローンを作成する。

cloneコマンドの結果を確認してみよう。

`git log`

```
commit 9561de3a55bed6bdd44a12820ba81ec416e705a7 (grafted, HEAD -> master, tag: v6.4-rc5, origin/master, origin/HEAD) author:
リーナス・トーバルズ <torvalds@linux-foundation.org>
日付 6月4日(日) 14:04:27 2023 -0400

Linux 6.4-rc5
```

なお、コミットログには1つの項目しかない。



Linuxカーネルについてもっと知りたい?

Linux Foundationには、組み込みシステムやカーネル・セキュリティなど、カーネル開発に関するさまざまなトピックを扱うクラスがあります:

[https://training.linuxfoundation.org/full-catalog/? sft topic area=linux-kernel-development](https://training.linuxfoundation.org/full-catalog/?sft%20topic%20area=linux-kernel-development)

ツリー型

最新のLinuxカーネルをダウンロードする際、**ブランチ**を指定した。

今すぐどういう意味ですか？

- ほと  コマンドは**コミットリファレンス**（特定の特定の時点でのファイルのコレクション）。

- **treeish** は、**コミット参照**を指定する構文である：

例 完全**コミットハッシュ** -

9561de3a55bed6bdd44a12820ba81ec416e705a7

- このハッシュの短縮版 - 9561de3a5

- **branch** - main または devel

- **タグ** - v6.4-rc5

- 特定の**リモートバージョン** - refs/remotes/origin/main

プロジェクトのディレクトリにある README や INSTALL ファイルには、そのプロジェクトの特定のバージョンを取得するための最適な方法が書かれていることがよくあります。**git** の機能や**コードフォージ**の機能の命名や使用方法はプロジェクトによって大きく異なるため、この情報を確認することは重要です。

たとえば、main（古いプロジェクトではmaster）が不安定な作業ブランチを指すこともあれば、テスト済みでリリースに適したバージョンを指すこともある。



もっと知りたい？

`git-revisions` のマニュアルページには、特定のコミットを指すために `treeish` を使用する例があります：

<https://mirrors.edge.kernel.org/pub/software/scm/git/docs/gitrevisions.html# リビジョンの指定>

ブランチのアナロジー

git の概念を理解するために、物理的な物体になぞらえて考えることは有用である。各コミットをバインダーの1ページのように考えるなら、**ブランチ**を、最後のページや最新のページを簡単にめくれるように、一番後ろに整理用のタブがついているバインダーと考えることができる。

バインダーの末尾にページを追加して**コミット**する場合、その末尾のタブは、やはり最後の、そして最新のページ／コミットに素早く導いてくれる。

このブランチから新しい**ブランチ**を作るのは、すべてのページをコピーして新しいバインダーに入れるようなものだ。新しいバインダーに新しいページを追加しても、古いバインダーは元の最終ページと最終タブで変わらない。

最後に、作成したそれぞれのバインダー/ブランチを、すべて同じシェルフに存在するものとして考えてみましょう。そのシェルフは**git** リポジトリで、**.git** サブディレクトリのあるディレクトリにローカルに存在します。

現在のブランチをコピー元として、新しいブランチを作成する：

```
$ git checkout -b <名前>.
```

アップストリームブランチをコピー元として、新しいブランチを作成します：

```
git checkout -b <名前> remotes/origin/main
```

フォーク? クローン?

例えを続けると、私たちの棚を`git`、**ブランチ**としていくつものバインダーを含みポジトリと想定している場合、それらのバインダー／ブランチの1つ以上をコピーして**クローン**を作り、まったく新しい**物理的な**棚／リポジトリに置く。

forkとcloneは、`git`。**Github**はこの**fork**という言葉の使い方を広めたが、**clone**は`git` プログラム内の機能の名前である。



非常に重要

同じファイルシステム上にある `git clone` をバックアップとして信用してはいけません。ローカルで **git clone** を実行すると、**git** は新しいサブディレクトリに `.git` ファイルのコピーを作成しますが、スペ

ースを節約するために可能な限り**ハードリンク**を使用します。

フォークの意味はさらに過大である。歴史的には、開発経路が分岐するソフトウェアのコピーを意味していた。現在では、ブランチやブランチのスタンドアロン（物理的）コピーを指すこともある。しかし、分岐した開発経路**以外**の独立したコピーを意味することもある。混乱するかもしれない。

ハードリンクを使わずに同じファイルシステム上にローカルを作る：

```
cd /tmp/LFgit
```

```
$ git clone --hardリンクなし /tmp/meow/
```

```
tmp/meow' にクローン...完了。
```

特定のスポットにタグを付ける

この例えの最後の部分として、**タグ**はバインダーのもうひとつのクイックリファレンス・タブであり、新しいコミットとともに *自動的に移動するものではない* と考えることができる。

タグは、特定のコミットへのエイリアスを追加する方法として使用されます。



あるコミットから別のコミットへタグを移動させるには、タグを削除し、再度作成する必要がある。

ほとんどのプロジェクトでは**注釈付きタグ**や**署名付きタグ**を使用しており、独自のコミットメッセージが必要です。デフォルトの **git タグ**

ビヘイビアはコミットメッセージを要求しないので、いわゆる **軽量タグ**はツールによっては無視される。

git tagで**sign**オプションを使用するには、GnuPGの設定が必要です。しかし、最後のステップを除けば、**注釈付きタグ**を作成する手順は同じです：

ギットタグ

-a, -annotated <タグ名> /は、符号なし注釈付きタグ・オブジェクトを作

る：

```
cd /tmp/LFgit
git tag -a v1.0 -m '初期バージョンをリリース'
git log --pretty=oneline v1.0
```

8933f422d17d107d35b80f8a1c55b09dbec289fe (HEAD -> main, tag: v1.0) critical 'meow' を motd に追加
0872249ec1c9daeb60cb491fe6be7647cf606149 本日のメッセージ



バージョン

ブランチと**タグ**で異なる命名スキームを持つことは、良い習慣です。そのため、バージョン3.17の次のリリースに向けて作業しているブランチは単に3.17とし、完成したリリースにはv3.17というタグをつけるかもしれません。

この慣例に従うことで、ブランチや**タグ**を最も単純な**ツリー式**で参照できるようになる。

13.5 ラボ

🔪練習問題 13.1: アップストリームソフトウェアリポジトリをクローンし、ローカルに変更を加える

システム管理では比較的一般的な作業である、ディストリビューションにパッケージされていない、あるいはバージョンが古いソフトウェア・パッケージへの対処について説明する。

使用するパッケージは<http://joeyp.name/code/moreutils/> の *moreutils* で、おそらくあなたのディストリビューションでパッケージ化され、最新版になっているはずだが、この例ではそうっていない:

1. には、かなりの数の枝と**タグ**がある。
2. は、多くの優れたRustやGoユーティリティのように多くの依存関係を必要としない。
3. をインストールしておくとお利便だ。

Debianバージョンをターゲットにすることで、ソースサーバーに少し優しくなり、もう少しリポジトリ情報を利用できるようになります。演習にはこのURLを使用してください:

<https://salsa.debian.org/nsc/moreutils>

1. まずリポジトリを**クローン**する。
2. `debian/latest` ブランチをベースに `local/latest` というローカルブランチを作成します。
3. リポジトリ内に `README.local` というファイルを作成し、好きなことを書いてください。
4. 新しいファイルを新しいブランチにコミットします。を設定する必要がある場合は、前のセクションの**最小グローバル設定**を参照してください。
user.nameおよびuser.email
5. (オプション) 現在のバージョンに基づいて注釈付きタグを作成する。
6. (オプション) パッケージをビルドし、ツールを使って実験する。

これらすべてを行うさまざまな段階で、`.git`ディレクトリの中身を調べ、どのファイルが変更されているか、その内容は何かなどを確認する。できる限り多くのことを学んでください。

✅解答 13.1

1. `$ git clone https://salsa.debian.org/nsc/moreutils`

```
$ git clone https://salsa.debian.org/nsc/moreutils
'moreutils' にクローンする...
警告: https://salsa.debian.org/nsc/moreutils.git/ リモートにリダイレ
クトしています: オブジェクトの列挙: 2107、完了。
リモートオブジェクトをカウント中: 100% (229/229),
完了: オブジェクトを圧縮: 100% (105/105), 完了。
リモート: 合計2107 (デルタ129)、再利用208 (デルタ121)、パック再利
用1878 オブジェクトを受信: 100% (2107/2107), 510.79 KiB | 1.45 MiB/s, 完了
。
デルタの解決: 100%(1349/1349)、完了。
```

2. `git checkout debian/latest`

```
すでに「debian/latest」になっている。
あなたのブランチは 'origin/debian/latest' で最新です。
```

(または)

`git status`

debian/latest ブランチ
あなたのブランチは 'origin/debian/latest' で最新に
なっています。

なぜならデフォルトのブランチはすでにdebian/latestだからだ。ですから、こ
れらのコマンドはどちらでも動作します：

`git checkout -b local/latest`

新しいブランチ「local/latest」に切り替えた。

(または)

`git checkout -b local/latest debian/latest`

新しいブランチ「local/latest」に切り替えた。

しかし、最後のものは、現在チェックアウトしているtextbfbranchに関係なく機能する。

3. `touch README.local`
4. `git add README.local`
`git commit README.local`

[local/latest cc7d808] ローカル・トラッキング・ファイルの追加
1 ファイル変更, 0 挿入(+), 0 削除(-) 作成モー
ド 100644 README.local

5. `git tag -a vlocal/1.0 -m '最初のローカルバージョン'`

第14章

プロセス



14.1	プログラムとプロセス	206
14.2	プロセス・リミット	210
14.3	プロセスの作成	213
14.4	プロセス制御	215
14.5	未来のスタートプロセス	218
14.6	プロセス・ステート	222
14.7	実行モード	223
14.8	デーモン	226
14.9	やさしさ	227
14.10	研究室	229

学習目標

このセッションを終えるまでに、あなたは次のことが
できるようになるはずだ：

- プロセスと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、スレッドを区別する。
- 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、フォアグラウンドに戻されるかについて話す。
- **ジョブ**（バックグラウンド・プロセス）を管理し、**at**を使用して将来プロセスを開始し、**スリープ**を使用して実行を一時停止する。

cronで定期的な実行をスケジュールする。

- 可能な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の実行の違いを説明する。
- デーモンプロセスについて説明する。
- 新しいプロセスがどのようにフォーク（生成）されるかを理解する。
- 優先順位の設定と変更には、**ナイス**と**レニーチェ**を使う。

14.1 プログラムとプロセス

プログラムとは何か？

- プログラムとは、実行される命令（およびそのために必要なデータ）の集合である。
- CPUが直接実行するマシンレベルの命令で構成されることもある。
- 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ることもある。
- プログラマーは、さまざまな言語（C、C++、Perlなど）を使用して、プログラムの命令をコーディングします。

バイナリ実行形式にコンパイルされるプログラムと、**bash**、**Python**、**Perl**などのインタプリタによって実行される必要のあるスクリプトを区別すべきである。

プログラムは命令の集合であり、命令を実行する際に使用される内部データも含まれる。プログラムは外部データを使用することもある。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る。外部データには、データベースからのデータが含まれる。

ls、**cat**、**rm**といった多くのユーザー・コマンドは、オペレーティング・システム・カーネルやシェルの外部にあるプログラムである（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る）。

プロセスとは何か？

- プロセスとは、実行されるプログラムのことである
- **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する。
- 複数のプロセスが同時に同じプログラムを実行する可能性がある。
- 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。

マルチスレッド

プログラムは複数の同時**スレッド**で構成されることがあり、各スレッドは独自のプロセスとみなされる。

プロセスとは、実行中のプログラムのインスタンスである。実行中やスリープ中など、さまざまな状態になることがある。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）を持つ。さらに各プロセスは、プログラムコード、データ、変数、ファイル記述子、環境を持つ。

プロセスは、完全にプリエンパティブな**スケジューリング**によって制御される。プロセスをプリエンプトする権利を持つのはカーネルだけであり、お互いにプリエンプトすることはできない。

歴史的な理由から、最大**pid**は16ビット、つまり32768に制限されてきた。この値を変更するには、[/proc/sys/kernel/pid_max](#)を変更する必要がある。プロセスが生成されると、最終的にはpid_maxに達し、その時点でPID = 300で再スタートする。

プロセスの属性

- すべてのプロセスには一定の属性がある：
 - 実行されるプログラム
 - コンテキスト（状態）
 - アクセス許可
 - 関連リソース

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中か、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る。これがプロセスの**コンテキスト**である。

プロセスは、他の人とCPU時間を共有するときにスケジュールされたりされなかったりする（あるいは、ユーザーがリクエストを出すとかデータが到着するとか、何らかの条件が満たされるのを待っている間にスリープ状態にする必要がある）ので、プロセスをスワップアウトするときにコンテキスト全体を保存し、実行再開時に復元できるようにすることは、カーネルが**コンテキスト・スイッチング**を行う能力にとって重要である。

各プロセスは、どのユーザーが実行を呼び出したかに基づいたパーミッションを持っている。また、誰がそのプログラム・ファイルを所有しているかに基づいたパーミッションを持っている場合もある。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"有効な"ユーザーIDを持っている。これらのプログラムは**setuidプログラム**と呼ばれる。非 setuid プログラムは、そのプログラムを起動したユーザーのパーミッションで実行されるのに対し、setuid プログラムは、そのプログラムを所有するユーザーのユーザーIDで実行される。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可能性がある。

passwdプログラムはsetuidプログラムの一例である。どのユーザーでも実行可能である。ユーザーがこのプログラムを実行すると、プロセスはroot権限で実行され、ユーザー・パスワードが管理されている書き込み制限ファイル [/etc/passwd](#) と [/etc/shadow](#) を更新できるようになる。

すべてのプロセスは、割り当てられたメモリやファイルハンドルなどのリソースを持っていることに注意してください。

プロセスリソースの分離

- カーネルはプロセスを互いに分離する（デフォルト）
 - 安全性を高める
 - 安定性の向上
- カーネルは、プロセスが任意のハードウェアリソースにアクセスすることを許可しない。
 - 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、独自のユーザー空間に隔離される。これによりセキュリティが促進され、安定性が増す。

プロセスはハードウェアに直接アクセスできない。ハードウェアはカーネルによって管理されているので、プロセスは**システムコール**を使わなければならない。

ハードウェアに間接的にアクセスするためのものです。システムコールは、アプリケーションとカーネル間の基本的なインターフェースである。

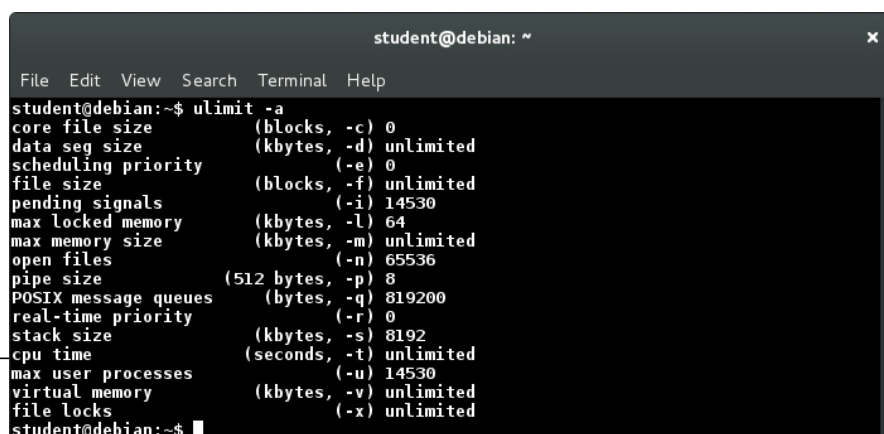
14.2 プロセスの限界

ulimitでプロセスを制御する

- **bash**コマンド内蔵
- プロセスリソース制限の表示またはリセット
- 個々のユーザーやプロセスが、以下のような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**ために使用する：
 - メモリー
 - 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できる

システム管理者は、これらの値のいくつかをどちらかに変更する必要があるかもしれない：

- 個々のユーザーやプロセスが、メモリや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処理するサーバーが、デフォルトの1024個のオープンファイルでは処理が不可能になる場合がある。



```
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 14.1: ulimit

制限の設定

- 特定のリミットを設定する:

`ulimit [オプション] [リミット]`

のように

`ulimit -n 1600`

これはファイル記述子の最大数を1600に増やすことになる。

- `/etc/security/limits.conf`を修正し、再ブートすることで永続化する。
 -

この変更は現在のシェルにしか影響しないことに注意してほしい。ログインしている全ユーザーに有効な変更を加えるには、`/etc/security/limits.conf`を修正する必要がある。このファイルは、非常にうまく自己文書化されている。

ハードリミットとソフトリミット

制限には2種類ある：

- **Hard**： ルートユーザーのみが設定できる、ユーザーがリソースの上限を引き上げることができる最大値。
- **ソフト**： ユーザーが変更できるが、ハードリミットを超えることはできない電流制限値。

`ulimit -H -n`

4096

`ulimit -S -n`

1024

14.3 プロセスの作成

プロセスの作成

- 最初のユーザー・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**からフォークすることで、子プロセスが生成される。
- 親が死亡した場合、その子供は「養子」となる。
- カーネルが作成したプロセスも存在する。これらのプロセスは、

**ps**

ご注意

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る。

平均的なLinuxシステムは、常に新しいプロセスを生成している。これはしばしば**フォーク**と呼ばれる。元の親プロセスは実行され続け、その間に新しい子プロセスが開始される。

単なるフォークではなく、親プロセスが終了し、子プロセスが親プロセスのプロセスIDを継承する**exec**を伴うこともよくある。**フォークとエグゼキューション**という言葉はよく使われるので、一つの単語として考えられることもある。

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**と呼ばれる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**の部分でもない。

新しいプロセスがどのように開始されるかを見るために、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よう。クライアントとの新しい接続が行われるたびに、新しいプロセスが起動するかもしれない。**Linuxでは**、完全なプロセスを作成しても、新しいスレッドだけを作成しても、技術レベルでは大差はない。

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され、**sshd init** スクリプトは **sshd デーモン**の起動を担当する。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、それ自身の新しいコピーを作成する。各リモートユー

ザは、リモートログインを行うために **sshd** デーモンのコピーを実行します。**sshd** プロセスはログインプログラムを起動し、リモートユーザを認証します。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、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のを確認したり、異なるCPUの負荷が均等にバランスされるようにしたり、デバイス・ドライバがキューに入れられた作業を処理するなどのメンテナンス作業を行う。これらのプロセスは、システムが稼動している間は、何かすることがあるとき以外はスリープしていることが多い。

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、そのコマンドは次のようになる。

バッシュ:

- 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・コールは、親シェル・プロセスをスリープ状態にする。
- コマンドは、**exec**コマンドで子プロセスの領域にロードされる。
システムコール。
- コマンドは実行を完了し、子プロセスはexitシステムコールで終了する。
- 親シェルは子プロセスの死によって再び起動し、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ます。

コマンドラインの末尾にアンパサンド（&）を追加して）バックグラウンド処理用のコマンドが発行された場合、親シェルは待機要求をスキップし、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列実行できるようにする。そうでなければ、**フォアグラウンドのリクエストに対して**、シェルは子プロセスが完了するかシグナルで停止するまで待つ。

いくつかのシェルコマンド（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ロードを伴わない。これらのコマンドでは、実行時に **fork** も **exec** も発行されません。

14.4 プロセス制御

背景プロセスと前景プロセス

- 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、そのコマンドが**バックグラウンド**で実行される：

```
sudo updatedb &
```

- CTRL-Zでフォアグラウンドプロセスを一時停止
 - **bg**はバックグラウンドで実行する
 - **fg**はそれを前景に戻す
- CTRL-Cでフォアグラウンド・プロセスを終了

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。これは問題ではないかもしれませんが、ジョブの完了に時間がかかる場合、バックグラウンドに置くことでシェルを解放し、さらに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ので、対話的な作業をスムーズに進めることができる。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・ウィンドウからログオフすることもできる。

```
$ updatedb &
```

```
[1] 22209
```

```
スリープ
```

```
100 ^Z
```

```
[2]+停止
```

```
スリープ
100
```

```
bg
```

```
[2]+スリープ100
```

```
fg
```

```
sleep 100
```

```
^Z
```

```
[2]+停止
```

```
スリープ
100
```

```
bg
```

```
[2]+スリープ100
```

```
fg
```

```
スリープ
```

```
100 ^C
```

求人管理

- jobsは現在のターミナルのバックグラウンドプロセスを表示する
- ジョブID、状態、コマンド名を表示します:

求人情報

1 - ランニング アップデート
2 + 停止 スリープ10

- ジョブIDはbgとfgで使用可能

\$ jobs -l

はジョブのPIDを提供する。

ログオフすると、そのウィンドウから開始されたジョブは表示されなくなる。

14.5 将来のスタートプロセス

未来から始めるアット・ユアーズ

- 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  
仕事 1 at 2013-01-16 13:24
```

- EOT文字を挿入するにはCTRL-Dを使用する。
- キューに入れられたジョブを調査または削除するために**atq**を実行する。

atに未来の時刻を入力すると、**at**の対話型パートが始まる。プロンプトで、実行するコマンドを入力し、Enterを押して、CTRL-Dで終了する。その後、**atq**を使ってジョブ情報を見る。

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することができる：
 - crontabでジョブを指定できる
 - `/etc/cron.d/`はフォーマットされたジョブファイルで拡張できる。
 - `/etc/cron.{hourly,daily,weekly,monthly}`. には には 任意のシステムスクリプト
- cronは何十年もの間、UNIX系OSで使われてきたが、最近のLinuxディストリビューションは**anacron**に移行している。

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します。

タスクは、1時間ごと、毎日、週に1回、月に1回、あるいは10秒ごとに実行するようにキューに入れることができる。ジョブが完了または失敗するとメールが送信されます。

`ls -l /etc/cron.d`

合計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
-l root root 235 Mar 29 2017 sysstat
```

`ls -l /etc/cron.daily`

合計 68

```
-rwxr-xr-x 1 root root      302 Mar 7 08:55 docker-latest-logrotate
-rwxr-xr-x 1 root 18812 Mar 20 00:01 google-chrome
-rwxr-xr-x 1 root root          1 2017年1月17日 google-earth
lrwxrwx 1 root ルート      45 2016/06/16 google-talkplugin -> (日本語)
  ↳ /opt/google/talkplugin/cron/google-talkplugin
-rwx----- 1 root root      219 Jan 24 2017 logrotate
-rwxr-xr-x 1 root root      618 2014/03/17 man-db.cron
-rwx----- 1 root root      208 Feb 4 2016 mlocate
-rwxr-x--- 1 root root 2126 2016年8月17日 prelink
-rwx----- 1 root root      256 2017年9月1日 rhsmcl
-rwxrwxr-x 1 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いくつかのジョブを実行するかという情報を含んでいた。
- しかし、マシンは常に動いていることが暗黙の前提だった。
- 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。
- 最近のシステムでは、`anacron`が`cron`に取って代わりました。
- `anacron`は、システムが稼動しているときに、必要なジョブを制御された順序で実行する。

anacron設定ファイルの例:

`/etc/anacrontab`

詳細は anacron(8) と anacrontab(5) を参照。

`SHELL=/bin/sh`

`PATH=/sbin:/bin:/usr/sbin:/usr/bin`

`MAILTO=""`

ジョブのランダム遅延に追加される最大ランダム遅延

ジョブは以下の時間帯にのみ開始される

`START_HOURS_RANGE=3-22`

期間 (日) 遅延 (分) コマンド

1	5	クローンデイリ	<code>nice run-parts /etc/cron.daily</code>
7	25	<code>cron.weekly</code>	<code>nice run-parts /etc/cron.weekly</code>

スリーピング

- スリープを使えば、簡単に命令を遅らせることができる:

`$ (sleep 600; launchbackup.sh) &`

`sleep`はシェルスクリプトでも使うことができる。例えば、会議の時間が近づいていることを10分ごとに知らせてくれる。

引数のデフォルトは秒だが、明示的に秒をs、分をm、時間をh、日をdとすることができる。

14.6 プロセス国家

プロセス国家

- プロセスは、いくつかの可能な状態のいずれかにある。
 - **実行中**：実行中：CPU上で実行中、またはO/Sが許可すれば実行可能な状態。
 - **待機**：ディスクや外部からの情報を待つこと。
 - **停止**：デバッガまたは Ctrl-Z で停止する。
 - **ゾンビ**：消えたが、親プロセスによって適切に削除されていない

実行状態のプロセスは、現在CPUから注目されているか（言い換えれば、実行中）、実行可能な状態で待ち行列に待機しているかのいずれかである。キューに入れられたプロセスは、その優先順位がキューの先頭に上がり、オペレーティング・システムにアイドル状態の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、そのプロセスが行った要求（通常はI/O）を待っており、要求が完了するまで先に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、処理が続行される。

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**ことがある。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たいときによく経験する。これが完了すると、プロセスは再開される。

ゾンビ状態は、プロセスが終了したときに入る。ゾンビ・プロセスは、O/Sのプロセス・テーブルでスペースを取り続けるため、重要である。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ーがある。ゾンビ・プロセスは、プロセス・テーブル・エントリ以外のすべてのリソースを解放する。

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。

14.7 実行モード

つの実行モード

- ユーザーモード
 - 制限された権限での実行
- 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

プロセス（またはマルチスレッド化されたプロセスの特定のスレッド）は、いつでも**ユーザーモード**または通常カーネル開発者は**カーネルモード**と呼ぶ。

どの命令を実行できるかはモードによって異なり、ソフトウェアレベルではなくハードウェアレベルで強制される。

モードはシステムの状態ではなく、プロセッサの状態である。マルチコアやマルチCPUシステムでは、各ユニットがそれぞれ個別の状態になりうるからだ。

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

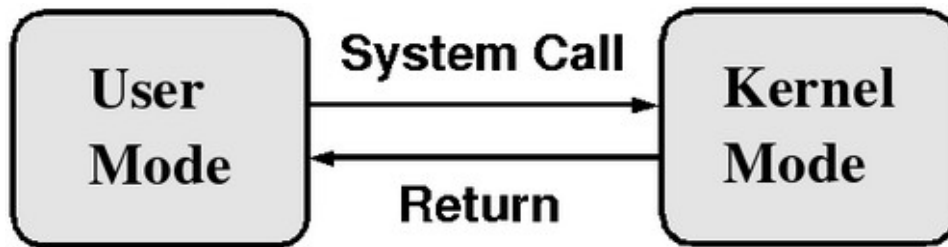


図14.2: ユーザーモードとカーネルモードとシステムコール

- **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る**

システム・コール（次のセクションで説明）を実行するときを除き、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しか持たない。

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、独自のユーザー空間に隔離される。これによりセキュリティが向上し、安定性が増す。これは**プロセス・リソースの分離**と呼ばれることもある。

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ー空間を持ち、その一部は他のプロセスと共有されることがある。共有メモリー・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ー空間への読み書きはできない。

ルート・ユーザーまたはsetuidプログラムとして実行されるプロセスでさえ、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る。

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続く
- ユーザー・モードより多くの特権
- 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る
 - ディスクアクセスの取得（読み取り、書き込み）
 - メモリの追加要求(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できる。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**を発行する必要があります、これにより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**が行われる。この手順は、ファイルの読み書きや新しいプロセスの作成などの際に行わなければならない。

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いが、システム・コール自体はカーネル・コードである。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。

システムがカーネルモードになるのは、ハードウェア割り込みを処理したり、スケジューリングルーチンやその他のシステム管理タスクを実行したりするときなど、プロセスとは関係のないときである。

14.8 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だけを目的としたバックグラウンド・プロセスである。
- 必要なときだけ作動するので、かなり効率的である。
- 多くは起動時に開始される。
- 名前はしばしば（常にではないが）**d**で終わる：たとえば、**httpd** や **systemd-udev**。
- 外部イベント(**systemd-udev**)や経過時間(**crond**)に反応することがある。
- 一般に、制御端子や標準入出力装置はない。
- より良いセキュリティーコントロールを提供することもある。

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ある。**xinetd**、**httpd**、**lpd**、**vsftpd**などがその例である。

14.9 やさしさ

ナイスを使って優先順位を決める

- 各プロセスは、プロセスの実行優先度に影響する**素敵な値**を持っている。
- **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る。
- ナイス値が高いほど優先順位が下がる
- ナイス値が低いほど優先順位が上がる
- 非特権ユーザーはナイス値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ができる

ナイス値の範囲は、-20 (最低のナイス、最高の優先度) から 19 (最高のナイス、最低の優先度) までである。ナイス値を下げるができるのはスーパーユーザーだけで、どのユーザーでもプロセスのナイス値を上げることができることに注意。デフォルトでは、プロセスの nice 値は 0 で、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われる：

```
nice -n 10 myprog
```

素敵な値10で**myprog**を実行

```
$ nice -n 19 myprog
```

マイプログを最も低い優先度で実行する。

```
sudo nice -n -20 myprog
```

最優先で**myprog**を実行し、いい値は-20

niceの値を与えない場合、デフォルトはniceを10増やす。引数を全く与えない場合は、現在の優しさを報告する。

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さえある。

ナイスバリューの修正

- **renice**は、実行中のプロセスの素敵な値を変更することができます。

`renice --help`

使い方:

```
renice [-n] <priority> [-p|--pid] <pid>...
renice [-n] <priority> -g|--pgrp <pgid>...
renice [-n] <priority> -u|--user <user>...
```

- pid 20003の値を5に上げる:

`$ renice +5 -p 20003`

- スーパーユーザーのみがプロセスのナイス値を下げるができる

renice は、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る。基本的には、その場で nice 値を変更することができる。

デフォルトでは、スーパーユーザーのみがnicenessを下げる、つまり優先度を上げることができる。しかし、[/etc/security/limits.conf](#)を編集することで、一般ユーザーにもあらかじめ決められた範囲内でnicenessを下げる権限を与えることができます。

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る。

すでに実行されているプロセスの "きれいさ" を変更するには、**renice** コマンドを使うのが簡単だ:

`レニス+3 13848ドル`

これは **pid = 13848** のプロセスの niceness を 3 にリセットする。複数のプロセスを同時に実行することもできるし、他にもいくつかオプションがあるので、**man renice** を参照のこと。

14.10 ラボ



ビデオ・デモンストレーション・リソース

レニス使用デモ.mp4

練習問題 14.1: ulimitを使ったプロセスの制御

どうぞよろしく:

[ヘルプ ulimit](#)

以下のステップを実行する前に、[/etc/security/limits.conf](#)を読んでください。

1. **bash**と入力して（または新しいターミナルを開いて）新しいシェルを起動し、変更が新しいシェルでのみ有効になるようにする。現在のオープンファイル数の制限を表示し、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を確認する。
3. ハードリミットを2048に設定し、動作することを確認する。
4. リミットを前の値に戻してみてください。うまくいきましたか？

✓ 解答 14.1

1. バッシュ
`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: 操作は許可されていません。
```

```
ulimit -n
```

```
2048
```

もうこんなことはできない!

もし、スタックサイズ (-s) のような別のリミットを選んでいたら、ハードリミットは次のようになる。
無制限だ。

🔗 演習 14.2: システムVのIPCアクティビティの検証

System V IPCは、UNIXの初期にさかのぼる、かなり古いプロセス間通信方法である。これには3つのメカニズムがある:

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

より近代的なプログラムでは、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るが、**System V IPC**アプリケーションはまだ数多く存在している。

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      msqid      所有者      パー      使用バイト数  メッ
キー      共有メモリ・セグメント
キー      シュミット      所有者      パーマ      バイト      ナッチ      ステータス

0x01114703 0          ルート      600        1000        6          デス
0x00000000 98305       コープ      600        4194304     2          ト
0x00000000 196610      コープ      600        4194304     2          デス
0x00000000 23068675    コープ      700        1138176     2          ト
0x00000000 23101444    コープ      600        393216      2          デス
0x00000000 23134213    コープ      600        524288      2          ト
0x00000000 24051718    コープ      600        393216      2          デス
0x00000000 23756807    コープ      600        524288      2          ト
0x00000000 24018952    コープ      600        67108864    2          デス
0x00000000 23363593    コープ      700        95408       2          ト
0x00000000 1441811     コープ      600        2097152     2          デス
ト
```

----- セマフォ・アレイ -----

キー	半	所有者	パーマ	ンスムス
0x00000000	98304	アパツ シュ	600	1
0x00000000	131073	アパツシ	600	1
ユ 0x00000000	163842	アパツシ	600	1
ユ 0x00000000	196611	アパツシ	600	1
ユ 0x00000000	229380	アパツシ	600	1
ユ				

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としても知られている）であり、親子関係にあるプロセス間でのみ共有されていることに注意。さらに、アタッチメントがなくなると、1つを除いてすべて破棄される。

セグメントを作成したプロセスや、セグメントに最後に付加されたプロセスについて、さらに詳しい情報を得ることができる：

ipcs -p

```

----- メッセージキューのPID -----
msqid      オーナー  lspid      lrpid
シュミット 所有者  cpid      エル
ド--- 共有メモリー・クリエーター ビーストオブPID -----
0          ルー      1023      1023
           ト
98305      コープ      2265      18780
196610     コープ      2138      18775
23068675   コープ      989       1663
23101444   コープ      989       1663
23134213   コープ      989       1663
24051718   コープ      20573     1663
23756807   コープ      10735     1663
24018952   コープ      17875     1663
23363593   コープ      989       1663
1441811    コープ      2048      20573

```

このように、やることによって:

```
ps aux |grep -e 20573 -e 2048
```

```

コープ      2048 5.3 3.7 1922996 305660 ?      Rl  10/27 77:07 /usr/bin/gnome-shell
コープ      20573 1.9 1.7 807944 141688 ?      Sl   09:56   0:01 /usr/lib64/thunderbird/thunderbird
プ
コープ      20710 0.0 0.0 112652 2312pt/0    S+   09:57   0:00 grep --color=auto -e 20573 -e 2048

```

thunderbirdが**gnome-shell**によって作成された共有メモリセグメントを使用していることがわかる。

システム上で以下の手順を実行し、誰が、どのようなリソースを使用しているかを確認する。潜在的な**漏れ**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ますか? 例えば

ipcs

```

-----
----- 共有メモリセグメント ----- キ
ー      シュミッド 所有者  パーマ  バイト  ナッチ  ステ
                                     タ
-----
0x00000000 622601                                     ス
0x0000001a 13303818                                デス
-----                                ト
                                     コープ  600      2097152  2
                                     コープ  666      8196    0

```

は、アタッチメントがなく、破棄するようにマークされていない共有メモリセグメントを示す。このため、後続のプロセスがアタッチしなければ、メモリがリークしたまま永遠に残る可能性がある。

🔪練習14.3: 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確認し、負荷の平均値を表示します。

✔解答 14.3

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ことだ：

アップタイム

```
10:26:40 up 3 :19, 5 users, load average: 1.46, 1.40, 1.19
```

もう一つの方法は、**トップからの出力の最初の行**を見ることである：

トップ

```
top - 10:28:11 up 3 :20, 5 users, 負荷平均: 1.93, 1.52, 1.25
タスク合計313個、                  1実行中、312スリープ、0 停止中          0 ゾンビ
CPU:  1.0us、0.3sy、0.0ni、98.2id、0.5wa、0.0hi、0.0si、0.0
KiB メモリ: 合計16284472、空き6556792、使用1029760、バッファ/キャッシュ8697920
kBメモリ: 合計8290300、空き8290300、使用10964220、空きメモリ
2615コープ      20   0   504536 186312 65488 S    6.7 1 .1   6分28秒30 スカ
                                     イピア・ビー・
                                     プラス
18248コープ      20   0   655804 50816 30884 S    6.7 0 .3   0:20.11 エマック
                                     ス
1ルート          20   0    204912   6508   3956 S    0.0 0 .0    0:00.92 systemd
```

第3の方法は、**w**を使うことだ：

```
10:30:51 up 3 :23, 5 users, load average: 0.55, 1.11, 1.14
ユーザー  TTY      フロム      ログイン   アイドル      JCPU   PCPU WHAT
コープ    0        :0           07:08      xdm?16:51     0.19s  gdm-session-
コープ    ポイン   :0           07:09      2:22m 0 .12s 0 .12s バッシュ
ト/0
コープ    点/1     :0           07:09      1:37m 0 .42s 0 .42s バッシュ
コープ    点/2     :0           07:09      0.00s 51.09s 0 .00s w
コープ    ポイン   :0           07:09      27:08   0.25s 0 .25s バッシュ
ト/3
```

練習問題 14.4: 背景と前景の仕事

ターミナル・ウィンドウからグラフィカル・プログラムを起動させ、ウィンドウ内で文字を入力できないようにする。

1. 次のように新しいファイルで**gedit**を開く。

```
$ gedit somefile
```

ターミナル・ウィンドウに文字を入力することはできなくなった。

2. ポインターをターミナルウィンドウの上に置いたまま、CTRL-Zを押す。

```
^Z
```

```
[3]+停止          ファイルを
                  編集する
```

geditウィンドウで文字を入力することはできなくなった。

3. **jobs -l**で、このターミナル・ウィンドウから起動されたプロセスを見ることができる：

```
$ jobs -l
```

[1] 17705 ランニン

グ

[2]- 18248 ランニ

ング

[3]+ 19827 停止

```
evince *.pdf &  
emacs /tmp/hello.tex &  
gedit somefile
```

4. 最新のジョブ (**gedit somefile**) をバックグラウンドに置く：

`bg`

```
[3]+ gedit somefile & (ファイル編集)
```

これで**gedit**ウィンドウに文字を入力できるようになるはずだ。

5. プロセスをフォアグラウンドに戻す:

```
fg
```

ファイルを編集する

ターミナル・ウィンドウに文字を入力できなくなったことに注意。

6. クリーンアップするには、プロセスを再度サスペンドし、killで終了させる:

```
^Z
```

```
[3]+停止          ファイルを
                  編集する
```

```
$ jobs -l
```

```
[1] 17705 ランニン          evince *pdf &
グ                          emacs /tmp/hello.tex &
[2]- 18248 ランニ          gedit somefile
ング
[3]+ 19827 停止
```

```
$ kill -9 19827
```

```
$ jobs -l
```

```
[1] 17705 ランニン          evince *pdf &
グ                          emacs /tmp/hello.tex &
[2]- 18248 ランニ          gedit somefile
ング
[3]+ 19827 死亡
```

```
$ jobs -l
```

```
[1]- 17705 ランニ          evince *pdf &
ング                      emacs/tmp/hello.tex&
[2]- 18248 ランニ
ング
```

✍️ 練習14.5: 将来バッチ処理にatを使う

今から未来の時刻に実行する、非常に単純なタスクをスケジュールする。これは、lsやdateを実行し、その出力を保存するだけの簡単なものです。(未来の1分という短い時間を使うこともできる)。

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。こうす

る:

1. 短い**bash**スクリプトから。
2. インタラクティブ

🟢解答 14.5

1. `testat.sh`ファイルを作成する:

```
#!/bin/bash  
日付 > /tmp/datestamp
```

それを実行可能にして、**at**でキューに入れる:

```
$ chmod +x testat.sh
現在+1分 -f testat.sh
```

ジョブが実行待ちになっているかどうかは、**atq**で確認できる:

```
atq
```

```
17      Wed Apr 22 08:55:00 2015 学生
```

ジョブが実際に実行されたことを確認する:

```
cat /tmp/datestamp
```

```
Wed Apr 22 08:55:00 CDT 2015
```

コマンドから>/tmp/datestampを取り除くとうなるでしょうか? (ヒント: プロンプトが表示されない場合は**mail**と入力してください!)。

2. 対話的には基本的に同じ手順だ。でジョブをキューに入れるだけだ:

```
現在$ + 1分
at> date > /tmp/datestamp
CTRL-D
atq
```

🔪練習14.6: cronによる定期タスクのスケジューリング

毎日午前10時に簡単なタスクを実行する**cron**ジョブを設定する。

✅解答 14.6

以下の内容で**mycrontab**という名前のファイルを作成する:

```
0 10 * * * /tmp/myjob.sh
```

そして、**/tmp/myjob.sh**を作成する:

```
SH /tmp/myjob.sh

#!/bin/bash
echo こんにちは、私は $(date) で $0 を実行しています。
```

を作成し、実行可能にする:

```
$ chmod +x /tmp/myjob.sh
```

で**crontab**システムに入れておく:

`crontab mycrontab`

でロードされていることを確認する:

`$ crontab -l`

```
0 10 * * * /tmp/myjob.sh
```

`sudo ls -l /var/spool/cron/student`


```
-rw----- 1 student student 25 Apr 22 09:59 /var/spool/cron/student
```

```
sudo cat /var/spool/cron/student  
0 10 * * * /tmp/myjob.sh
```

こんにちは、私は/tmp/myjob.shを Wed Apr 22 10:03:48 CDT 2015で実行しています。

もし、本当に毎日これを実行したくないのであれば、以下のようなメッセージをプリントアウトしてください：

で削除することができる：

```
$ crontab -r
```

ある日の午前10時にマシンが起動していない場合、アナクロンは適切な時間にジョブを実行する。

第15章

プロセス監視



15.1	プロセス監視	238
15.2	トラブルシューティング	239
15.3	ps	241
15.4	pstree	243
15.5	トップ	245
15.6	研究室	247

学習目標

このセッションを終えるまでに、あなたは次のことができるようになるはずだ：

- 様々なコマンドラインプロセスモニタリングツールを特定し、使用する。
- システム問題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を表示する方法を学びます。
- 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** を使用する。
- システム負荷をインタラクティブに表示するには **top** を使用する。

237

15.1 プロセス監視

プロセス監視ツール

- ・ トップ
- ・ アップタイム
- ・ ps
- ・ プスツリー
- ・ エムピースタット
- ・ イオスタット
- ・ サル
- ・ ヌマスタット
- ・ ストレース

プロセスを監視するために、Linux管理者は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る。

以下は、プロセス・モニタリングのための主なツールのリストである：

表15.1: プロセスと負荷の監視ツール

工具	目的
トップ	動的に更新されるプロセス活動
アップタイム	システムの稼働時間と平均負荷
ps	プロセスの詳細情報
プスツリー	プロセスとその接続のツリー
エムピースタット	複数プロセッサの使用
イオスタット	CPU使用率とI/O統計
サル	システムアクティビティに関する情報の表示と収集
ヌマスタット	NUMA（非均一メモリ・アーキテクチャ）に関する情報

ストレース

プロセスが行うすべてのシステムコールに関する情報

`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役立つ。

15.2 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:** すぐに教えることができる。
- **経験豊富:** 練習を数年積んだ後。
- **ウィザード** 生まれつきそうでなければならないと思っている人がいるが、それはナンセンスだ。

どんなによく管理されたLinuxシステムでも、問題は発生する。

トラブルシューティングは、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、問題がシステムローカルに起因するのか、ローカルネットワーク内またはインターネットに起因するのかを切り分けることができる。

適切なトラブルシューティングには判断力と経験が必要であり、それは常にいくぶん芸術的なものではあるが、優れた方法論に則った手順を踏むことで、再現可能な方法で問題の原因を切り分けることができる。

グ

基本的なトラブルシューティングのテクニック

- 問題の特徴づける。
- 問題を再現する。
- いつも簡単なことから試してみる。
- 可能性のある原因を一つずつ取り除く。
- 一度に1つだけ変更し、それで問題が解決しない場合は元に戻す。
- 詳細については、システムログ（`/var/log/messages`、`/var/log/secure`など）を確認してください。

直感に基づく飛躍は推奨されない。チェックリストと統一された手順を使用する動機は、ウィザードへの依存を避け、どんなシステム管理者でも、よく知られた手順に従えば、最終的には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、ウィザード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。

一方、自分の直感を尊重し、直感をチェックすることを選択するのであれば、生産性がありそうかどうかに基づいて、直感的な道を継続するか放棄するかを決定するのに十分なデータを素早く得られるようにすべきである。

直感を無視することで問題解決に時間がかかることもあるが、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基準となる。言い換えれば、有用な直感は魔法ではなく、抽出された経験なのだ。

15.3 ps

ps によるプロセス状態の表示

- その情報を `/proc` から取得する。
- 選択とプレゼンテーションのオプションに非常に柔軟性がある
- 一般的な選択肢をいくつか挙げる：

```
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主力ツールである。

このコマンド・ユーティリティは、すべてのUNIX系オペレーティング・システムに存在しており、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る：

1. UNIXのオプションの前には-をつけなければならない。
2. BSD オプションは - を前につけてはならない。
3. GNU ロング・オプション。それぞれのオプションの前には--を付けなければならない。

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。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う傾向がある。

```
c8:/tmp>ps auxf
USER      PID  %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         2   0.0  0.0      0     0 ?        Ss   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 15.1: BSDオプションでpsを使う

ps出力のカスタマイズ

- `o` オプションを使い、その後にフィールド識別子をカンマ区切りで並べると、`ps` フィールドの選択されたリストを印刷することができる。
 - `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割合。
- 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のこと。

```
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15.2: ps出力のカスタマイズ

もう一つの一般的な選択肢、`-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15.3: UNIXオプションでpsを使う

15.4 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 を表示するには -p を使用する。
- pid] とその祖先をハイライトするには -H [pid] を使う。

```
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する：

`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
|  |-{県},18040
|   |-{県},18046
|    -{県},18047
```

多くのオプションの説明については、**pstree**のmanページを参照されたい。

子プロセスの1つ (**evince**、pid=18036) には3つの子プロセスがあることに注意してください。別の見方は以下の通りです：

`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
```


15.5 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
- プロセスは最初、CPU使用率でソートされる
- セキュアモード(top s)で実行されていない場合、ユーザはプロセスにシグナルを送ることができる。
 - kキーを押す
 - プロンプトが表示されたらPIDを指定する
 - プロンプトが表示されたら、シグナル番号を入力してください。
- 豊富なオプション、古代のユーティリティ、インタラクティブ性
- hキーを押すとキーバインディング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。表示はさまざまな基準で並べ替えることができます。表示は5秒ごとに更新され（これは設定可能）、ユーザーが中断（Ctrl-C）するまで実行され続ける。

```
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15.4: 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ースである。
- `/proc` には、各アクティブ・プロセスのサブ・ディレクトリがあり、プロセスID (PID) で命名される。
- `/proc/self` は現在実行中のプロセスである。
- いくつかの調整可能なパラメーターは `/proc` ディレクトリにある。
- 詳細は `proc` のマニュアルページを参照。

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使われていることに注目してほしい。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を表している。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ファイルが含まれている。

```
student@opensuse: ~/Desktop
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15.5: /procの内容

15.6 ラボ



ビデオ・デモンストレーション・リソース

top_demo.mp4
ps_demo.mp4

練習問題 15.1: プロセス

1. efオプションを付けて**ps**を実行する。出力の違いに注意。
2. プロセスID、優先度、ナイス値、プロセス・コマンド行のみが表示されるように、**ps**を実行する。
3. コマンドラインで**bash**と入力して、新しい**bash**セッションを開始する。**nice**コマンドを使って別の**bash**セッションを開始するが、今度はniceの値を10にする。
4. ステップ2と同様に**ps**を実行し、プライオリティとniceの値の違いを確認する。2つの**bash**セッションのプロセスIDに注意してください。
5. **renice**を使って、**bash**セッションの1つのnice値を15に変更する。もう一度、優先順位とnice値の変化を観察する。
6. **top**を実行し、出力が変化するのを見る。qを押してプログラムを停止する。

✓ 解答 15.1

1. `ps -ef`
`ps aux`
2. `ps -o pid,pri,ni,cmd`

```
PID PRI NI CMD
2389 19 0 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. `バッシュ`
`nice -n 10 bash`
`ps -o pid,pri,ni,cmd`

```
2389 19 0 バッシュ
22115 19 0 バッシュ
22171 9 10 バッシュ
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 19    0 バッシュ
22115 19    0 バッシュ
22171  4 15 バッシュ
22246  4 15 ps -o pid,pri,ni,cmd
```

5. トップ

✍️ 練習15.2: プロセス状態の監視

1. ddを使って、`/dev/urandom`から読み込み、`/dev/null`に書き込むバックグラウンド・プロセスを起動する。
2. プロセスの状態をチェックしてください。どうあるべきか？
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。これでどうなる？ もう一度プロセスの状態を見てください。
4. 求人プログラムを実行してください。何を教えてくれる？
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`を使って終了させる。

✅ 解答 15.2

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かRであるべきだ。

3. `fg`
`^Z`
`ps -C dd -o pid,cmd,stat`

```
PID CMD          スタ
25899 dd if=/dev/urandom of=/dev/ T
                                ット
```

州はTであるべきだ。

4. `jobs`コマンドを入力してください。何が表示されますか？

求人情報

```
[1]+停止          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`コマンドを使ってジョブを終了させる。

```
fg
$ kill 25899
```

第16章

メモリの監視、使用状況、スワップの設定



16.1	メモリのモニタリングとチューニング	250
16.2	/proc/sys/vm	252
16.3	バイムスタット	253
16.4	スワップ	254
16.5	メモリ不足キラー (OOM)	255
16.6	研究室	257

学習目標

このセッションを終えるまでに、あなたは次のことができるようになるはずだ:

- メモリの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`を解釈する。
- `vmstat`を使用して、メモリ、ページング、I/O、プロセッサ・アクティビティ、およびプロセスのメモリ消費に関する情報を表示する。
- **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・エリアを設定し、監視し、規制するかを理解する。
- **OOMキラー**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かを選択する方法を理解する。

16.1 メモリの監視とチューニング

メモリ監視

Linuxにおけるメモリのモニタリング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
プロップス	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	詳細な仮想メモリ統計とブロックI/O、動的に更新される	プロップス
地図	プロセス・メモリー・マップ	プロップス

	合計	中	無	共有バッファ/キ	利用可
メモ	15901	古	料	992	4502
スワ	8095	2080	9317	キャッシュ	12457
ップ		0	8095		

時が経つにつれて、RAMの価格が下がり、性能が向上すると同時に、システムはメモリ・リソースに対する要求が高くなってきた。

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、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る。

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ある。

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可能性がある。まず第一に、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しなければならない。

したがって、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及ぼし、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の影響を及ぼす可能性がある。

メモトータル	16254132	キロ バイト	ライトバック :	0	キロ バイト	VmallocChunk:	0	キロ バイト
メンフリー:	162652	キロ バイト	アノン・ペー ジズ	2780004	キロ バイト	Percpu:	8352	キロ バイト
MemAvailable:	12234416	キロ バイト	マッピングされ た:	679632	キロ バイト	HardwareCorrupted:	0	キロ バイト
緩衝材:	847972	キロ バイト	シュメモ	523880	キロ バイト	AnonHugePages:	958464	キロ バイト
キャッシュされ た:	10959104	キロ バイト	KReclaimable :	1124064	キロ バイト	ShmemHugePages:	0	キロ バイト
スワップキャッ シュ:	13276	キロ バイト	スラブ:	1233300	キロ バイト	ShmemPmdMapped:	0	キロ バイト
アクティブだ :	3252848	キロ バイト	要求できる:	1124064	キロ バイト	FileHugePages:	0	キロ バイト
活動休止中:	11343484	キロ バイト	スレクレーム	109236	キロ バイト	FilePmdMapped:	0	キロ バイト
アクティブ(ア ノ人):	393416	キロ バイト	KernelStack:	16512	キロ バイト	HugePages_Total:	0	
活動休止中 (anon):	2919760	キロ バイト	ページテーブ ルズ	79092	キロ バイト	HugePages_Free:	0	
アクティブ(ファイル):	2859432	キロ バイト	NFS_Unstable:	0	キロ バイト	HugePages_Rsvd:	0	
非アクティ	8423724	キロ	バウンス:	0	キ	ページ数	0	

[proc/meminfo](#) 擬似ファイルには、メモリがどのように使用されているかについての豊富な情報が含まれている。

16.2 /proc/sys/vm

/proc/sys/vm

- バーチャル・メモリー・システムをコントロールするための調整可能なノブを多数搭載
- ほとんどすべてのエントリーが（rootによって）書き込み可能である。
- 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る。
- 値を変更するには、エントリーに直接書き込むか、あるいは **sysctl**

`proc/sys/vm`のパラメーターを調整する場合、一度にひとつずつ調整し、効果を確認するのが通常のベストプラクティスだ。主な（相互に関連する）タスクは以下の通り：

- フラッシング・パラメーターを制御する。つまり、どれだけのページがダーティで、どれだけの頻度でディスクにフラッシュされるかを制御する。
- スワップ動作を制御する。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しておくか、それとは対照的に、他の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メモリに残しておくかを制御する。
- 多くのプログラムは、特にコピー・オン・ライト（COW）技術のために、要求されたメモリー量をフルに必要とすることはないためである。

メモリーのチューニングは微妙なもので、あるシステム状況や負荷ではうまくいっても、他の状況では最適とはほど遠いこともある。

```

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_pdlush_threads       swappiness
dirty_bytes               lowmem_reserve_ratio     oom_dump_tasks          user_reserve_kbytes
dirty_expire_centiseecs   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_centiseecs min_free_kbytes          overcommit_ratio

```

図16.1: /proc/sys/vm

16.3 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。
- 多くのオプション
`vmstat [オプション] [delay] [count]`
- delayは秒単位で指定され、レポートはカウント回数繰り返される。
- 最初の行は、前回のレポートからの平均を示す。

```
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16.2: vmstat

オプション `-S m` を指定すると、メモリ統計は KB ではなく MB 単位になる。

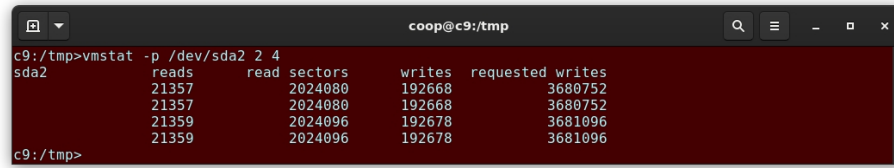
アクティブ・ページとは最近使用されたページのこと、クリーン（ディスクの内容が最新）である場合もあれば、ダーティ（いずれディスクにフラッシュする必要がある）である場合もある。対照的に、非アクティブメモリページは、最近使用されておらず、クリーンである可能性が高く、メモリプレッシャーがかかるとすぐに解放される：

メモリはアクティブなリストと非アクティブなリストの間を行ったり来たりし、新たな参照先が増えたり、使用間隔が長くなったりする。

```
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16.3: vmstatの使用

パーティション1つだけの簡単な統計を取りたい場合は、-pオプションを使う。



```
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16.4: 1つのディスクでのvmstatの使用

16.4 スワップ

スワップの使用

- 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やファイルをフォーマットする
- **swapon**: スワップパーティションまたはスワップファイルをアクティブにする

• **swapon**: スワップパーティションの使用
 /dev/sda6 8290300 0 -1
 /tmp/swapfile 102396 0 -2

- ブート時にマウント可能 (/etc/fstabを使用)

- 複数のスワップ・パーティションを持つことができる

`cat /proc/swaps`

Linuxは仮想メモリ・システムを採用しており、オペレーティング・システムが実際よりも多くのメモリを持っているかのように機能することができる。このようなメモリのオーバーミッションは、2つの方法で機能する:

- 多くのプログラムは、使用許可が与えられたメモリを実際にはすべて使用していない。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で、子プロセスは変更があったときだけ (ページ単位で) 一意のコピーを取得する。
- メモリへの負荷が重要になると、あまりアクティブでないメモリ領域はディスクにスワップアウトされ、再び必要になったときだけ呼び出される。

このようなスワッピングは通常、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対して行われる。Linuxでは複数

の**スワップ・エリア**を許可しているため、ニーズを動的に調整することができる。各領域には優先順位があり、優先順位の低い領域は優先順位の高い領域が埋まるまで使用されない。

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**RAM**の合計です。cat /proc/swapsでシステムが現在使用しているスワップ領域を確認し、freeで現在の使用量を報告することができる。いつでも、ほとんどのメモリはファイルの内容をキャッシュするために使用され、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するのを防いでいる。そのようなメモリーのページは、バッキングストアがファイルそのものであるため、スワップに書き出すことは無意味であるため、スワップアウトされることはない。代わりに、ダーティページ（保存された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ー）がディスクにフラッシュアウトされる。

Linuxでは、アプリケーション・メモリーとは対照的に、カーネル自身が使用するメモリーは**決して**スワップアウトされない。

OOM)

16.5 メモリ不足キラー (OOM)

OOMキラー

- システムは物理メモリーを使い果たすかもしれない。その可能性がある：
 1. メモリーが解放されるまで、それ以上のメモリー要求を拒否する。
 2. **スワップ**領域を使用して物理メモリーを拡張する。
 3. 選択したプロセスを（インテリジェントに）終了させ、メモリー使用量を減らし、システムを存続させる。
- **Linux**システムは、2番目と3番目の方法を実装していることが多い。
- 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。

メモリープレッシャーに対処する最も単純な方法は、空きメモリーがある限りメモリー割り当てを成功させ、すべてのメモリーを使い果たした時点で失敗させることである。

この場合、（少なくとも理論上は）利用可能なメモリーの合計は、実際のRAMにスワップ領域のサイズを加えたものになる。この場合、（少なくとも理論的には）利用可能なメモリーの合計は、実際のRAMにスワップ領域のサイズを加えたものになる。難しいのは、プレッシャーがかかったときにスワップアウトするメモリーのページを見つけ出すことだ。このアプローチでは、スワップ領域自体が一杯になると、新しいメモリーへの要求は失敗しなければならない。

つまり、RAMとスワップのサイズを超えるメモリー要求を許可することができる。これは無謀に思えるかもしれないが、（ほとんどではないにせよ）多くのプロセスは要求されたメモリーをすべて使うわけではない。

例えば、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プログラムである。もうひとつの例は、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ー空間全体のコピーを受け取るというものだ。LinuxはCOW（コピー・オン・ライト）技術を使用しているため、プロセスの1つがメモリーを変更しない限り、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かもしれないと想定しなければならない。

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、通常のオペレーションに依存することを意図していない。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か？
- プロセスの選択は、以下の**バッドネス値**に依存する。
各プロセスの `/proc/[pid]/oom_score`
- 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る。

`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る：

- 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する。
- 1: 全てのメモリーリクエストはオーバーコミットが許可される。
- 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、スワップ領域のサイズにRAMの設定可能な割合（デフォルトでは50）を加えたサイズに達すると、メモリ・リクエストは失敗する。この比率は `/proc/sys/vm/overcommit_ratio` を変更することで変更できる。

これについては、アンドリース・ブルーワー（<https://lwn.net/Articles/104185/>）が面白い見解を示している：

ある航空機会社が、航空機に搭載する燃料を減らした方が安くなることを発見した。飛行機は軽くなり、燃料の使用量も減る。しかし、まれに燃料が不足し、飛行機が墜落することがあった。この問題は、特別なOOF（燃料切れ）機構の開発によって解決された。緊急時には、乗客が選ばれ、機外に放り出された。（多くの理論が開発され、脱出させる犠牲者を適切に選ぶという問題に多くの出版物が費やされた。犠牲者は無作為に選ぶべきか？ それとも体重の重い人を選ぶべきか？ それとも年長者？ 乗客は脱出させないために金を払うべきなのか。また、例えば最も体重の重い人が選ばれたとして、それがパイロットであった場合の特例を設けるべきか？ ファーストクラスの乗客は除外すべきか？ OOFメカニズムが存在する今、それは時々作動し、燃料不足でなくても乗客を排出する。技術者たちは、この誤作動がどのようにして起こるのかを正確に研究中

16.6 ラボ



ビデオ・デモンストレーション・リソース

[swap_and_oom_demo.mp4](#)

練習16.1: スワップ領域の管理

現在のスワップ領域を調べる:

```
cat /proc/swaps
```

ファイル名	タイプ・パーティション	サイズ	使用	優先度
/dev/sda11	ディスク	4193776 0		-1

新しいパーティションかファイルを追加して、スワップ領域を増やそう。ファイルを使うにはこうする:

```
dd if=/dev/zero of=swpfile bs=1M count=1024
```

```
1024+0レコード
1024+0レコード
1073741824バイト (1.1GB) のコピー、1.30576秒、 822MB/秒
```

```
mkswap swpfile
```

```
スワップスペース・バージョン1の設定、サイズ =
1048572 KiB ラベルなし、UUID=85bb62e5-84b0-
4fdd-848b-4f8a289f0c4c
```

(本物のパーティションの場合は、パーティション名を**mkswap**に入力するだけ。ただし、そのパーティ

ション上のデータはすべて消去されるので注意!)新しいスワップ領域をアクティブにする:

```
sudo swapon swpfile
```

```
swapon: /tmp/swpfile: insecure permissions 0664, 0600 suggested.
swapon: /tmp/swpfile: insecure file owner 500, 0 (root) suggested.
```



Red Hat、Centos、Fedoraの場合

RHELは私たちが安全でないと警告している:

```
sudo chown root:root swpfile
sudo chmod 600 swpfile
```

swpfileが使用されていることを確認する:

`cat /proc/swaps`

ファイル名	タイプ	サイズ	使用	優先度
/dev/sda11	・パー	4193776 0		-1
/tmp/swpfile	ティシ	1048572 0		-2
	ョン・			
	ファイ			
	ル			

優先順位の低いスワップ・パーティションやファイルは、優先順位の高いものが満たされ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、スペースを節約するために削除する：

```
sudo swapoff swpfile
sudo rm swpfile
```

🔗練習16.2: OOMキラーの発動

- Linuxカーネルが極度のメモリ・プレッシャーにさらされると、恐ろしい**OOM**（Out Of Memory）**キラー**が起動する。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「最適な」プロセスを選択しようとする。
- システムにメモリ不足を強制し、何が起こるか見てみよう。まず最初にターミナル・ウィンドウを開き、そこにこう入力する：

```
sudo tail -f /var/log/messages
```

カーネル・メッセージが表示されるのを監視するためである。

- さらに良い方法は、次のようなものだ：

```
dmesg -w
```

カーネル以外のメッセージは表示されないからだ。

- この練習は、まずコマンドですべてのスワップをオフにした方がやりやすいだろう：

```
sudo /sbin/swapoff -a
```

を使って後でオンに戻すようにしてください。

```
sudo /sbin/swapon -a
```

- さて、これからシステムをメモリ・プレッシャーにさらします。独自の方法を見つけるのは自由だが、メモリを消費するプログラムも用意した：



ラボの廃棄物

```

1  /* メモリのデフラグを行うシンプルなプログラム、J.
2  Cooperstein 2/04
3  */
4
5
6  #include <stdio.h>
7  #include <stdlib.h>
8
9  #include
10
11  <unistd.h> #include
12  #define BS 16
13  #define CHUNK (MB*BS) /* 各ステップでBS*MBを割り当てる */
14  #define QUIT_TIME 30
15  void quit_on_timeout(int sig)
16  {
17      printf("\nTime expired, quitting\n");
18      #define MB (1024*1024)
19      exit(EXIT_SUCCESS);
20  }
21  int main(int argc, char **argv)
22  {

```



```

23 struct sysinfo si;
24 int j, m;
25 char *c;
26
27 /* システム上の総メモリを取得 */
28
29 sysinfo(&si);
30 m = si.totalram / MB;
31 printf("Total usable Memory in MB is %d MB\n", m);
32 printf("Using somewhat less: %d MB\n", m/10);
33
34 if (argc == 2) {
35     m = atoi(argv[1]);
36     printf("Choosing instead mem = %d MB\n", m);
37 }
38
39 signal(SIGALRM, quit_on_timeout);
40 printf("Will quite in QUIT_TIME seconds if no normal
41 termination\n"); 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る、
45      * そういふことだ!
46      */
47     c = malloc(CHUNK);
48     /* ブロックをjで何度も埋める */ memset(c, j,
49     CHUNK);
50     printf("%08d", j);
51     fflush(stdout);
52 }
53 printf("\n Sleeping for 5 seconds\n");
54 sleep(5);
55 printf("\n 終了してメモリを解放する; )
56 exit(EXIT_SUCCESS);

```

何MB消費するかを引数として受け取る。システムがメモリを使い果たすまで、要求されるメモリ量を徐々に増やしながら実行し続ける。



ご注意

とするだけで、プログラムをコンパイルして実行できるはずだ:

```
$ gcc -o lab_wastemem lab_wastemem.c
./lab_wastemem 4096
```

これは4GBを浪費することになる。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のは良いアイデアだろう（しばらく表示がフリーズするかもしれないが!）。

- OOM（Out of Memory）キラーが急襲し、生き残りをかけてプロセスを殺そうとするのを見るべきだ。誰が最初にやられるのか？

✔ 解答 16.2

[lab_wastemem.c](#)
[lab_waste.sh](#)

第17章

I/Oモニタリングとチューニング



17.1	I/Oモニタリング	262
17.2	イオスタット	263
17.3	イオトップ	264
17.4	研究室	265

学習目標

このセッションを終えるまでに、あなたは次のことができるようになるはずだ：

- 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となるタイミングを理解する。
- **iostat** を使って、システム I/O デバイスの活動を監視する。
- **iotop** を使用すると、現在のI/O使用量の表が常に更新されて表示される。

17.1 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

- 入出力 (I/O) アクティビティを監視することは、最高のパフォーマンスを得るために不可欠である。
- システムは、I/O バウンド、CPU バウンド、メモリ バウンドのいずれかである。
- 重要なツールをいくつか紹介しよう：
 - イオスタット
 - イオトップ

ディスク・パフォーマンスの問題は、メモリ不足や不十分なネットワーク・ハードウェアやチューニングなど、他の要因と強く結びついていることがある。これを切り離すのは難しい。

原則として、CPU が I/O の完了を待ってアイドル状態になっていたり、ネットワークがバッファのクリアを待っていたりする場合、システムは I/O バウンドしているとみなすことができる。

しかし、誤解されることもある。メモリが不足しているように見えるのは、I/O が遅すぎるためである可能性がある。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、メモリが問題であるように見えるかもしれないが、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である。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することもあります。

リアルタイムの監視とトレースは、ディスクのボトルネックを特定し、緩和するために必要なツールである。しかし、まれな問題や繰り返さない問題は、これを達成することを困難にする。

関連する変数は多く、I/O チューニングは複雑だ。

17.2 イオスタット

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ユーティリティ
- 非常に柔軟なレポートオプション

`iostat [OPTIONS] [デバイス] [間隔] [カウント]`

```

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         0.25    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         0.31     638049     6892
dm-1               0.22        213.06         0.00    4791481         60
dm-2               0.01         0.15         0.00       3357         12
dm-3               0.01         0.16         0.00       3545         12
dm-4               0.00         0.06         0.00       1404          0
dm-5               0.00         0.00         0.00         82          0
dm-6               0.01         0.15         0.00       3412         12
dm-7               0.00         0.00         0.00         82          0
dm-8               0.00         0.06         0.00       1404          0
loop0             41.24         41.29         0.00    928544          0

c8:/tmp>
  
```

図17.1: 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できる。

CPU使用率の簡単な要約の後、I/O統計が示される：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マージできる）、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）、および読み書きのブロックの合計。

情報はディスクパーティションごとに分けられます（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も）。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     0.25    0.00    0.02   0.99   16.41   73.69   127.10    0.04   662.96    3.06    1.07   0.05
sdb                2.33    5.30    81.40    90.98    0.02    7.65   0.86   59.06    0.30    1.03    0.00    34.90   17.17   0.38   0.29
sdc                1.45    1.65    67.78   202.40    0.00    1.94   5.28   54.02    1.04   13.65   0.02   46.75   122.37   0.30   0.00
dm-0               0.09    0.08    28.66     0.31    0.00    0.00   0.00   0.00   165.62   131.17   0.03   304.18    3.99   1.10   0.02
dm-1               0.21    0.00   202.69    0.00    0.00    0.00   0.00   0.00   35.93   21.90   0.01   972.10    2.86   1.21   0.03
dm-2               0.01    0.00     8.14    0.00    0.00    0.00   0.00   0.00   19.50   11.89   0.00   17.86    1.33   0.48   0.00
dm-3               0.01    0.00     8.15    0.00    0.00    0.00   0.00   0.00   19.05   20.00   0.00   18.56    1.33   0.42   0.00
dm-4               0.00    0.00     8.06    0.00    0.00    0.00   0.00   0.00   36.25    0.00   0.00   22.29    0.00   0.82   0.00
dm-5               0.00    0.00     8.00    0.00    0.00    0.00   0.00   0.00   50.36    0.00   0.00    2.93    0.00   0.86   0.00
dm-6               0.02    0.00     8.18    0.00    0.00    0.00   0.00   0.00   118.46   18.33   0.00   10.60    1.33   0.27   0.00
dm-7               0.00    0.00     8.00    0.00    0.00    0.00   0.00   0.00   78.43    0.00   0.00    2.93    0.00   0.79   0.00
dm-8               0.00    0.00     8.06    0.00    0.00    0.00   0.00   0.00   49.84    0.00   0.00   22.29    0.00   0.70   0.00
loop0             39.27    0.00    39.31    0.00    0.00    0.00   0.00   0.00    0.06    0.00   0.00    1.00    0.00   0.01   0.06

```

図17.2: iostatの使用

17.3 イオトップ

イオトップ

- **iostat**は現在のI/Oアクティビティを表示する
- **トップ**のように定期的に更新

ごちゃごちゃしないためには、-oオプションを使うと便利だ。

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	PRIO	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17.3: iostatの使用

もうひとつの非常に便利なユーティリティは**iostat**で、rootで実行する必要がある。これは現在のI/O使用量の表を表示し、定期的に更新する。PRIO列のbeはベストエフォート、rtはリアルタイムを表す。

```
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17.4: iostatのオプション

17.4 ラボ

練習問題 17.1: bonnie++

bonnie++は、ドライブやファイルシステムの性能をテストし測定する、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**の子孫です。

結果はターミナル・ウィンドウから読み込んだり、ファイルに出力したり、**csv**フォーマット（カンマ区切り値）に変換することもできる。コンパニオン・プログラムである **bon csv2html** と **bon csv2txt** を使って、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。

bonnie++には、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実行するかについて、かなり多くのオプションがあるため、使用する前に**bonnie++のマニュアルページ**を読むことを推奨する。簡単な概要は

bonnie++ --help

```
bonnie++: 無効なオプション -- '-'
の使用:
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]] とする。
        [x number-of-tests] [-u uid-to-use:gid-to-use] [-g gid-to-use].
        [q] [-f] [-b] [-p processes | -y] [-z seed | -Z random-file] [-D]
```

バージョン: 1.98

簡単なテストは次のようなコマンドでできる:

```
$ time sudo bonnie++ -n 0 -u 0 -r 100 -f -b -d /mnt
```

どこだ？

- **-n 0**は、ファイル作成テストを行わないことを意味する。
- **-u 0**はrootとして実行することを意味する。
- **-r 100**は、100MBのRAMを持っていることにする。
- **-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する。
- **-b**は、書き込みのたびに**fsync**を行うことを意味し、キャッシュへの書き込みだけでなく、ディスクへのフラッシュを強制する。
- **-d /mnt**は、作成されたテンポラリファイルを置くディレクトリを指定するだけである。

もしメモリーサイズを記入しなければ、プログラムはシステムの容量を計算し、その2-3倍の大きさのテスト用ファイルを作

成します。なぜなら、感触をつかむのに時間がかかるからだ。



レッドハットについて

RHEL/CentOS 9 については、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ます。
。98-1.el8.x86_64.rpmでインストールする：

```
sudo dnf localinstall bonnie++-*.rpm
```

RHEL/CentOS 8システムの場合

合：



```
c8:/tmp>sudo bonnie++ -n 0 -u 0 -r 100 -f -b -d /mnt
```

uid:0、gid:0を使用。

知的に書く...完了 リライト...完了

知的な読書...終了

開始...完了...完了...完了...完了...完了...完

ボ…完子…完了…完子…シーケンシャル出力-----シーケンシャル入力----

1.98 -Chr.ごとに--ブロック-- -Rewrite- -Chr.ごとに--ブロック
 --ファンタ--

300M ----- 271m 20 288m 15 ----- ++++++ 391622

[illegible]

```

ズなど
198 198 c8 1 1616174245 300M 8100 5 277162 20 294543 15 +++++ + 3916 22

```

30us, 20us, 12us, 18075us, , , , , ,



Ubuntuの場合

Ubuntu20.04システム上で、同じ物理的マシンの**VMware**ハイパーバイザー下で仮想マシンとして稼動:

```
sudo bonnie++ -n 0 -u 0 -r 100 -f -b -d /mnt
```

uid:0, gid:0.

知的に書く...完了 リライト...完了

知的な読書...終了

スタート...完了...完了...完了...完了...完了...完了...完了...完了...完了...

バージョン -----シーケンシャル出力----- --シーケンシャル入力----

1.98 ニンダノ

300M 287m 88 ++++++

名前: サイ

3805us	4650円	2136年	1852年
			シーク

```
1.98,1.98,ubuntu,1,1616156268,300M,8192,5,293955,88,+++++,+++ ,+++++ +++,+++++,+++ \
```

ubuntu 3805us, 4650us, 2136us, 1852us,,,,,,秒 %CP /秒 %CP /秒 %CP /秒 %CP /秒 %CP /秒 %CP /秒 %CP

— *Journal of the American Medical Association*, 1997

テンソー

明らかにパフォーマンスが低下しているのがわかります。以前の出力を**bonnie++.out**というファイルとして保存したと仮定すると、出力をhtmlに変換することができます：

```
bon csv2html < bonnie++.out > bonnie++.html
```

またはプレーンテキストに変換する:

```
bon csv2txt < bonnie++.out > bonnie++.txt
```

ドキュメントを読んだ後、より長く、より大きく、より野心的なテストを試してください。私たちがオフにしたテストのいくつかを試してみてください。システムの動作が良好であれば、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きましょう。

📝練習問題 17.2: fs マーク

fs markベンチマークは、複数のディレクトリとドライブにわたって非同期I/Oを多用し、ファイルシステムを低レベルでバッシングする。これはRick Wheeler によって書かれたかなり古いプログラムで、時の試練に耐えてきた。

<http://sourceforge.net/projects/fsmark/> からダウンロードできる。tarballを入手したら、解凍してコンパイルする：

```
tar zxvf fs_mark-3.3.tgz
cd fs_mark
作る
```

READMEファイルを読んでください。もしコンパイルに失敗し、次のようなエラーが表示されたら、READMEファイルを読んでください:

作る

```
-----
/usr/bin/ld: -lc が見つからない。
```

それは、静的バージョンの **glibc** がインストールされていないからだ。Red Hat ベースのシステムでは、こうすればインストールできる:

```
sudo dnf install glibc-static
```

と SUSE 関連システムで

```
sudo zypper install glibc-devel-static
```

Debian ベースのシステムでは、関連するスタティック・ライブラリーは共有ライブラリーと一緒にインストールされるので、追加のパッケージを探す必要はない。



Red Hat、Centos、Fedora の場合

- **RHEL 8** では、glibc-static が codeready-builder レポで利用可能で、これを有効にする必要があるかもしれない。
- しかし、**CentOS 8** では **PowerTools** レポに含まれている。すでに `/etc/yum.repos.d/CentOS-PowerTools.repo` がシステムにある場合は、`enabled=1` が設定されていることを確認してください。そうでなければ、このファイルのコピーを作成し、このコースの SOLUTIONS セクションに配置します:

テストとして、2500 個のファイルを作成し、それぞれ 10KB の大きさに、書き込みのたびに **fsync** を実行してディスクにフラッシュアウトする。これは、`/tmp` ディレクトリのコマンドで実行できる:

```
$ fs_mark -d /tmp -n 2500 -s 10240
```

これを実行している間に、拡張 **iostat** 統計情報を収集する:

```
iostat -x -d /dev/sda 2 10
```

を別のターミナル・ウィンドウで実行する。

注意すべき数値は、**fs mark** が報告する 1 秒あたりのファイル数と、**iostat** が報告する使用帯域幅の割合である。これが 100% に近づいている場合は、I/O バウンドしている。

第18章

コンテナの概要



18.1	容器	270
18.2	コンテナと仮想マシン	271
18.3	ドッカー	272
18.4	カウセイの例	275
18.5	再現可能なビルド	279
18.6	研究室.....	283

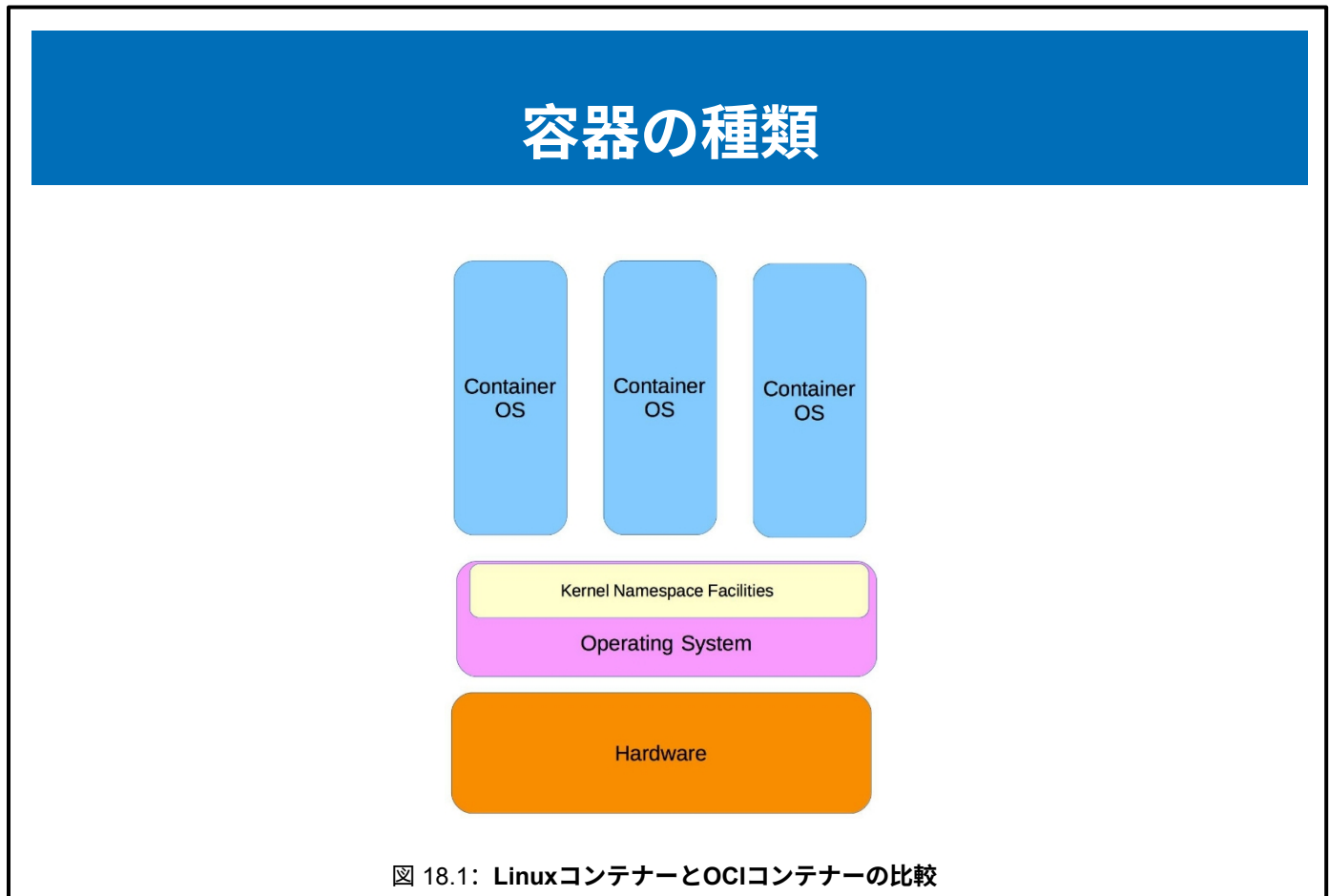
学習目標

このセッションを終えるまでに、あなたは次のことができるようになるはずだ：

- 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。
- 小さなユーティリティ・プログラムをシステムに追加する詳細な例で、**Docker**の使い方を学ぶ。
- 再現可能なビルドの入門として、バージョン仕様の問題点を分析する。

269

18.1 容器



名前空間や**cgroups**を含む**Linux**カーネルの機能を使うことで、ホスト・**カーネル**を共有するOSやアプリケーションの軽量のインスタンスを提供することができる。当初、**Linux Containers**または**LXC**は、**init**、システムツール一式、およびアプリケーションを含む仮想OS全体をコンテナとして仮想化できるようにするために作成された。

次に登場したのが**アプリケーション・コンテナ**で、**Docker**から始まり、一般的にOCIコンテナと呼ばれている。これらのコンテナは通常、OS全体を含まない（ただし、含むことはできる）。

今日、このようなアプリケーション・コンテナを実行するためのOCI互換技術は、*podman*、*runc*、*firecracker*、*apptainer*、*rkt*など数多くある。

この章では**Docker**と**docker compose**機能に焦点を当てますが、ほとんどのテクニックは他のOCIランタイムにもそのまま適用できます。一般的に、**Docker**を指定した**コンテナ・レジストリ**のイメージは、少し修正するだけで他のランタイムでも使用できます。

18.2 コンテナと仮想マシン

コンテナと仮想マシン

- 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できる
 - コンテナは通常1つのことを実行する
- VMはコンテナよりも多くのリソースを使用する
- 複数のコンテナは1つのOSカーネルを共有し、各VMは独自の
- コンテナはより持ち運びやすい
- コンテナはVMの中で実行できる
- コンテナは安全確保が難しい
- コンテナは通常、より速くスタートする
- しかし、仮想マシンは依然として最良の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る。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た。

どちらも長い歴史がある：

- メインフレームコンピューターには何十年も前から、かなり特殊な方法でソフトウェア・パーティショニングと仮想マシンが搭載されてきた。
- オペレーティング・システムには、何年も前から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る。

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は、サービスとして機能する仮想マシンが最適

実行す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ている場合は、仮想マシンが最適かもしれない。

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などの**オーケストレーションシステム

必要なコンテナの適切な数量を決定し、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る。

18.3 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、多くの個別のイメージを使用するアプリケーションレベルの仮想化である。これらのイメージはコンテナにパッケージ化される。

- 画像はコンテナ内のコンポーネント
- 画像には以下が含まれる場合がある。
 - 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
 - .アプリケーションに必要なものなら何でも。
- イメージは**Docker Hub**またはレジストリ・サーバに置くことができます。

Docker には <https://docs.docker.com> に膨大なドキュメントがある。

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る。

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コードやサービスすべてと一緒にパッケージ化し、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だ。このデプロイは何度でも簡単に繰り返すことができる。これにより、エンド・アプリケーションをサポートするためにレイヤー化されたサービスを備えたサーバーを構築する必要性が減少する。

Dockerイメージを探す

Dockerコンテナ化されたアプリケーションの立ち上げには、いくつかのステップしかないかもしれない。

- 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てください。
 -
- **Docker**サービスを開始する。
- **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する。
- 画像を引っ張る。
- イメージを実行する
- そして最後にアプリケーションをテストする。

上記のステップは、**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ない。

もちろん、イメージを作成し、システム変数やコンフィギュレーション・パラメーターを設定し、その結果を新しいイメージとして保存する関数など、使用できるオプションはたくさんあります。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場合もあります。

ほとんどの**Docker**コマンドには個別の**マニュアル**ページがある。例えば `docker(1)`、`docker-search(1)`、`docker-pull(1)`、`docker-create(1)`、`docker-run(1)`などです。

Dockerイメージを拡張する

既存のコンテナ・イメージを変更して、異なる設定値を提供したり、より多くの機能を追加したい場合がよくあります。これにはいくつかの方法がある：

- DockerfileのFROMコマンドを使って、別のイメージをベースに新しいイメージを作成する。
- **docker-compose** の使用
- あるイメージのソースDockerfileを見つけ、それをあなたのプロジェクトのベースとして修正する。

ほとんどの場合、FROMコマンドで他のイメージを継承することになります。セキュリティや内部コンプライアンス上の理

由から、ローカル・ソースや複数のベース・オペレーティング・システム・ステップを継承することもあります。

18.4 カウセイの例

牛が言う

ここでは、`cowsay`コマンドを使った非常に基本的な例を示す。

cowsay-minimal.Dockerfile

```
# モー
FROM アルパイン:エッジ
RUN apk add --no-cache -X http://dl-cdn.alpinelinux.org/alpine/edge/testing cowsay
ENTRYPOINT ["cowsay"] (カウセイ)
```

`docker run --rm cowsay-minimal 'and....アクション!'`

<そして...アクション!

```
  _
 / \
(  o  )ノ_
 (  )  )\
 ||----w |
 ||     ||
```

この例では、**Alpine Linux**を使用している。**Alpine Linux**はサイズが小さいため、コンテナのベースとしてよく使用される。

アルパイン・リナックスは、これまで取り上げてきたメインラインのディストリビューションとは、以下のようないくつかの点で異なっている：

- デフォルトで読み取り専用メディアから実行するように設計されている
- 標準Cライブラリとしてglibcの代わりに**musl**を使う
- はまったく別のパッケージ・マネージャーを使用している。



もっと知りたい？

このディストリビューションについて詳しく説明するつもりはないが、より詳しい情報を得ることができる：

1. アルパイン・リナックス - <https://alpinelinux.org/about/>
2. musl C ライブラリ - <https://musl.libc.org/>

3. apk パッケージマネージャ - https://wiki.alpinelinux.org/wiki/Alpine_Package_Keeper

どのように牛

これを分解してみよう：

```
# モー  
FROM アルパイン:エッジ
```

コンテナのベースには、OSのローリングリリースを使用する。

```
RUN apk add --no-cache -X http://dl-cdn.alpinelinux.org/alpine/edge/testing  
cowsay
```

cowsay をインストールするにはシステムパッケージコマンドを使います。

```
ENTRYPOINT ["cowsay"] (カウセイ
```

docker runがcowsayコマンドとして使うデフォルトのコマンドを設定する。

上のRUNステップでは、非常に複雑なコマンドを使った。わかりやすくするために、同じことを複数のコマンドで行うことはできなかったのだろうか？

Dockerイメージのサイズと構成は非常に重視されており、ほとんどの人がそれに同意している：

- 小さい方が良い
- レイヤーは少ない方が良い

Dockerfileの各RUN行は、**アトミック**に実行されるコマンドを表している。これらのアクションはそれぞれ**レイヤー**と呼ばれるスナップショットに保存され、最終的なイメージを作成するためには、これらすべてのレイヤーをダウンロードする必要がある。

レイヤーを少なくするテクニックはいくつかある：

1. シェル演算子&&を使用することもある。
2. この詳細を自動的に処理するシステムを構築する

3. 多段階ビルド

ビルド・カウ

Dockerファイルを作成したら、それをビルドする必要がある：

```
docker build -f cowsay-minimal.Dockerfile -t cowsay-minimal .
```

Dockerデーモンにビルドコンテキストを送信す

る 3.072kB Step 1/3 : FROM alpine:edge

---> 855f10c27d71

ステップ 2/3 : RUN apk add --no-cache -X http://dl-cdn.alpinelinux.org/alpine/edge/testing

└─カウセイ

---> キャッシュの使用

---> 8d2935a86a26

ステップ3/3: ENTRYPOINT ["cowsay"] (エントリーポイント

---> キャッシュの使用

---> 90a4fc4e505d

90a4fc4e505dのビルドに成功しました

cowsay-minimal:latestのタグ付けに成功し

ました

docker buildのデフォルトの動作を変更するために、いくつかの引数を指定した：

-f, -file 使用するDockerfileのパス/ファイル名

-t, -tag ビルド成功時に作成されるタグの名前。

.

最後の必須引数は作業ディレクトリである。この場合、カレント・ディレクトリである。

牛を走らせる

ドッカー・イメージを手に入れたら、それを使ってシステムからの情報とやりとりすることができる。

```
docker run --rm cowsay-minimal "$(cat /etc/debian_version)"
```

```
-----
< 11.7 >

  \      ^
   \    (•o•)ノ_
    (  )      )\
     ||-----w ||
     ||           ||
```

docker run コマンドの内訳:

-rm イメージから作成されたコンテナを削除し、毎回新しくスタートできるようにする。

カウセー・ミニマル

画像名、またはこの場合は**タグ**で、実行する特定の画像を指定します。

"\$(cat /etc/debian_version)"

には、docker ENTRYPOINT プログラムに渡される残りの引数を指定します。この場合、ホスト上で実行されたサブシェルの結果として、引数は1つだった。



非常に重要

動的引数はdockerに渡す前に処理される。



ご注意

別のコマンドを実行するコマンドを実行する場合、最初のプログラムへの引数リストの最後を -- 引数で指定するのは、しばしば良い習慣である。

つまり、この実行コマンドは次のように書くこともできる。

```
docker run --rm cowsay-minimal -- "$(cat /etc/debian_version)"
```


18.5 再現可能なビルド

間違った牛

コンテナを使う論拠のひとつは、複数のマシンで同じ環境を構築できることだ。しかし、dockerや他のコンテナ・システムを使うだけでは、この保証は完全には得られない。例に戻ろう：

cowsay-minimal.Dockerfile

```
# モー  
FROM アルパイン:エッジ  
#
```

● **Debian Testing**や● **CentOS Stream**に似ている。これらのタイプのディストリビューションは頻繁に、時には毎晩のようにアップデートが行われます。同じ**dockerイメージ**を明日ビルドするとしたら、どの程度違うだろうか？

エッジではなく、上記のように異なるバージョンを指定したとしても、セキュリティアップデートやその他の変更が公開される可能性があります。ソフトウェア開発のベースとして **docker** を使っている場合、その **docker イメージ**の予期せぬ変更が、プロジェクトのコードや設定に加えた変更のように見えるかもしれない。

より具体的に

FROM行はバージョンだけでなく、sha256ハッシュによるダイジェストも指定できる:

cowsay-digest.Dockerファイル

モー

FROM alpine:edge@sha256:2d01a16bab53a8405876cec4c27235d47455a7b72b75334c614f2fb0968b3f90

RUN apk add --no-cache -X http://dl-cdn.alpinelinux.org/alpine/edge/testing cowsay **ENTRYPOINT** ["cowsay"].

Dockerfileのコピーを作成し、現在のイメージのダイジェストを指定し、ビルド・ステップを再度起動した:

`docker build -f cowsay-digest.Dockerfile -t cowsay-digest .`

よく似た出力が続く

通常、新しいイメージのダイジェストハッシュを見つけるには、docker コマンドラインツールの代わりにウェブブラウザを使うのが一番早い。

ここではdockerタグまたはラベルの両方を指定しているが、OCIコンテナプログラムによっては、sha256だけ、またはsha256のみで動作するものもある。

ハッシュ

画像を比較する

docker images

リポジトリ	TAG	画像ID	作成	サイズ
カウセー・ダ	最新	90a4fc4e505d	13分前	43.2MB
イジェストカ	の最	90a4fc4e505d	13分前	43.2MB
ウセー・ミニ	新			

余分なハッシュ情報を使うことで、dockerが懸念する限り同じイメージを作成したことがわかる。これで、新しいバージョンがalpine:edgeに公開されても、誤ってアップグレードしてしまう心配はなくなった。



非常に重要

特定のバージョンを宣言 (pin) しておけば、誤ってアップグレードすることはなくなりますが、重要なセキュリティパッチを受けられるように、何らかのアップグレード手順を用意しておくことが重要です。

いくつかの情報セキュリティ企業が、公表されている公開コンテナの約半数にセキュリティ上の脆弱性があり、中には非常に深刻なものもあるという報告書を発表している。

しかし、どうだろう？

この例で変更できるのは、ソフトウェアのベースイメージだけではないことにお気づきだろうか？

`cowsay-digest.Dockerファイル`

```
RUN apk add --no-cache -X http://dl-cdn.alpinelinux.org/alpine/edge/testing cowsay
```

ベースシステムの一部ではないプログラムをインストールしているのですが、それも変更できないのでしょうか？

そうだ。実際、画像の**ビット単位**で同じバージョンを作りたい場合、一時ファイルやファイルシステム内のファイルのタイムスタンプなども考慮しなければなりません。

難しい問題ではあるが、その利点は多く、この問題領域ではすでに重要な研究がなされている。

もっと知りたい？



詳しくは<https://reproducible-builds.org/>。

タイムスタンプが変更された同じソースからリビルドした後でも、同じパッケージのバージョンを提供する方法の一つは、「スナップショットサーバ」を利用することです。

🔴 **Debian** での実現方法については、こちらのブログ記事 (<https://www.qubes-os.org/news/2021/10/08/reproducible-builds-for-debian-a-big-step-forward/>) をご覧ください。

18.6 ラボ

✎練習18.1: 小さなユーティリティをdockerで利用可能にする

cowsayの例を参考に、**httpie**ユーティリティ (<https://github.com/httpie/httpie>)用に同様の小さなDockerコンテナを構築してみよう。

✔解答 18.1

アルパインのLinuxソリューション:

1. パッケージ名を確認する

httpie "だと推測したのであれば、それで問題ないだろう。また、

https://pkgs.alpinelinux.org/package/edge/community/x86_64/httpieのパッケージ検索を使うか、アップストリームのREADMEの情報をチェックすることもできる。

2. Dockerファイルを作成する

これにはミニマル・バージョンでもダイジェスト・バージョンでも構わないが、この例では、各ビルドで最新バージョンが望まれるため、おそらく実際にはミニマル・バージョンを使うだろう。加えて、テストのために余計なrepo引数は必要ないが、あっても何の支障もない。

httpie.Dockerファイル

```
FROM アルパイン:エッジ
RUN apk add --no-cache httpie
CMD ["httpie"]
```

3. イメージを構築する

`docker build -f httpie.Dockerfile -t httpie:latest .`

```
Dockerデーモンへのビルドコンテキストの送信
30.72kB Step 1/3 : FROM alpine:edge
---> 855f10c27d71
ステップ 2/3 : RUN apk add --no-cache httpie
---> 2f6b1436e5e3 で実行中
fetch https://dl-cdn.alpinelinux.org/alpine/edge/main/x86_64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/edge/community/x86_64/APKINDEX.tar.gz
(1/57) libbz2 (1.0.8-r6) のインストール
(2/57) libexpat (2.5.0-r2) のインストール
(3/57) libffi (3.4.4-r3) のインストール
(4/57) gdbm (1.23-r1) のインストール
(5/57) xz-libs (5.4.3-r1) のインストール
(6/57) libgcc (13.1.1_git20230617-r0) のインストー
```

ル (7/57) libstdc++ (13.1.1_git20230617-r0) のインストール (8/57) mpdecimal (2.5.1-r2) のインストール (9/57) ncurses-terminfo-base (6.4_p20230625-r0) のインストール (10/57) libncursesw (6.4_p20230625-r0) のインストール (11/57) libpanelw (6.4_p20230625-r0) のインストール (12/57) readline (8.2.1-r2) のインストール (13/57) sqlite-libs (3.42.0-r2) のインストール (14/57) python3 (3.11.4-r0) のインストール (15/57) python3-pycache-pyc0 (3.11.4-r0) のインストール (16/57) pyc (0.1-r0) のインストール (17/57) py3-multidict のインストール (6.0.4-r2) (18/57) py3-multidict-pyc のインストール (6.0.4-r2) (19/57) py3-parsing のインストール (3.1.0-r0) (20/57) py3-parsing-pyc のインストール (3.1.0-r0) (21/57) py3-packaging のインストール (23.1-r1) (22/57) py3-packaging-pyc (23.1-r1) (23/57) py3-setuptools のインストール (68.0.0-r0) (24/57) py3-setuptools-pyc (68.0.0-r0) のインストール

(25/57) py3-pip (23.1.2-r0) のインストール
(26/57) py3-pip-pyc (23.1.2-r0) のインストール
(27/57) py3-pygments (2.15.1-r0) のインストール
(28/57) py3-pygments-pyc (2.15.1-r0) のインストール
(29/57) py3-pysocks (1.7.1-r5) のインストール
(30/57) py3-pysocks-pyc (1.7.1-r5) のインストール
(31/57) py3-certifi (2023.5.7-r0) のインストール
(32/57) py3-certifi-pyc (2023.5.7-r0) のインストール
(33/57) py3-charset-normalizer (3.1.0-r1) のインストール
(34/57) py3-charset-normalizer-pyc (3.1.0-r1) のインストール
(35/57) py3-idna (3.4-r4) のインストール
(36/57) py3-idna-pyc (3.4-r4) のインストール
(37/57) py3-urllib3 (1.26.15-r2) のインストール
(38/57) py3-urllib3-pyc (1.26.15-r2) のインストール
(39/57) py3-requests (2.31.0-r0) のインストール
(40/57) py3-requests-pyc (2.31.0-r0) のインストール
(41/57) py3-requests-toolbelt (1.0.0-r0) のインストール
(42/57) py3-requests-toolbelt-pyc (1.0.0-r0) のインストール
(43/57) py3-attrs (23.1.0-r1) のインストール
(44/57) py3-attrs-pyc (23.1.0-r1) のインストール
(45/57) py3-mdurl (0.1.2-r2) のインストール
(46/57) py3-mdurl-pyc (0.1.2-r2) のインストール
(47/57) py3-markdown-it-py (3.0.0-r0) のインストール
(48/57) py3-markdown-it-py-pyc (3.0.0-r0) のインストール
(49/57) py3-rich (13.4.2-r0) のインストール
(50/57) py3-rich-pyc (13.4.2-r0) のインストール
(51/57) py3-wheel (0.40.0-r1) のインストール
(52/57) py3-wheel-pyc (0.40.0-r1) のインストール
(53/57) httpie-pyc (3.2.2-r0) のインストール
(54/57) py3-defusedxml-pyc (0.7.1-r4) のインストール
(55/57) python3-pyc (3.11.4-r0) のインストール
(56/57) py3-defusedxml (0.7.1-r4) の

インストール (57/57) httpie (3.2.2-r0) のインストール

busybox-1.36.0-r5.trigger の実行 OK:

72パッケージで95MB

中間コンテナの削除 2f6b1436e5e3
---> 227f4a0c96ab

ステップ3/3: ENTRYPOINT ["httpie"]

---> f19a66bab6cb で実行中

中間コンテナf19a66bab6cbの削除
---> c0b613745c84

c0b613745c84 のビルドに成功しました。

httpie:latest のタグ付けに成功しました。

4. イメージを実行する

`docker run --rm httpie`

usage: httpie [-h] [--debug] [--traceback] [--version] {cli,plugins} ... httpie:
エラーです: これらのいずれかを指定してください: cli、'plugins'

このコマンドはHTTPIeプラグインを管理するためだけのものです。

リクエストを送信するには、http/httpsコマンドを使用してください:

`http POST pie.dev/post hello=world`

`https POST pie.dev/post hello=world`

そして、提案されたコマンドを試す:

```
docker run --rm httpie https
```

を使用している:

```
http [METHOD] URL [REQUEST_ITEM ...].
```

エラー:

には以下の引数が必要です: URL

をご覧ください:

[http --help](#) を実行するか、<https://httpie.io/docs/cli>。

5. (オプション) エイリアスに追加する

```
https='docker run --rm httpie https -- '
```

```
https httpie.io/hello
```

```
{"ahoy":["Hello, World! HTTPieをお試しいただきありがとうございます。
```

```
  ↳友情。"],"
```

```
  ↳リンク": {"homepage": "https://httpie.io", "twitter":
```

```
    "https://twitter.com/httpie", "discord": "https://httpie.io/discord", "github":
```

```
    "https://github.com/httpie"}}}
```


第19章

LinuxファイルシステムとVFS



19.1	ファイルシステムの基本	288
19.2	ファイルシステム の概念	289
19.3	仮想ファイルシステム (VFS)	291
19.4	利用可能なファイルシステム	292
19.5	ファイルシステムのジャーナリング	294
19.6	特殊ファイルシステム	295
19.7	研究室	296

学習目標

このセッションを終えるまでに、あなたは次のことができるようになるはずだ：

- 基本的なファイルシステムの構成について説明する。
- VFS（仮想ファイルシステム）の役割を理解する。
- **Linux**で利用可能なファイルシステムと、実際のシステムで利用可能なファイルシステムを知る。
- ジャーナリングファイルシステムが大きな進歩を遂げた理由を理解する。
- **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。

287

19.1 ファイルシステムの基本

ファイルシステムの基本

- **Linux**プログラムはディスクセクタではなくファイルを読み書きする
- **ファイル**は、実際には物理的なI/Oレイヤーをカモフラージュする抽象化である。
- **ファイルシステム**は物理パーティションに使用可能なフォーマットを作成する
- **UNIXライクなファイルシステム**は、ツリー階層を使用する。
 - 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
 - すべてのパスまたはノードはルート(/)ディレクトリの下にある
- 複数のファイルシステムを1つのツリー構造にマージすることができる（そして通常はそうする）。
- Linuxは仮想ファイルシステム・レイヤー（**VFS**）を使ってファイルシステム・ソフトウェアと通信する。

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**を読み書きする。

ファイルとその名前は、物理的なI/Oレイヤーをカモフラージュする抽象化である。コマンドラインから（**ファイルシステム層を無視して**）ディスクに直接書き込む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・アプリケーションでは行われない。例外は、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**cmd**アクセスを行う、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であっ

たり、**論理ボリューム・マネージャー（LVM）**によって管理される論理パーティションであったりします。また、ファイルシステムはネットワーク的な性質を持ち、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ります。

19.2 ファイルシステムの概念

イノード

- ファイル名は、より基本的なオブジェクトである**inode**のプロパティにすぎない。
- Inodeはファイルシステム上に保存されるだけでなく、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、保存する：
 - 許可
 - 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
 - * 最終修正時間
 - * 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る。

inodeはディスク上のデータ構造で、ファイルの場所などの属性を記述して保存する。

Linuxファイルシステムに含まれるすべてのファイルは、それ自身のinodeに関連付けられている。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、所有者など）、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る。このため、ファイルに関するすべてのI/Oアクティビティは通常、ファイルのinodeにも関与する。

タイムスタンプは3つある：

1. **アクセス時間**：ファイルが最後にアクセスされた時間。

2. **変更時間**: ファイルの**内容**が最後に変更された時間。
3. **変更時間**: パーミッション、所有者、ファイル名、ハードリンクなどの変更によって、ファイルのinodeが最後に変更された時間。

ハードリンクとソフトリンク

- **ハードリンク**はinodeを指す。
 - 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なければならない。
- **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りま
 - ターゲットが削除されていなかったり、ぶら下がっていたりする。
- ハードリンクされたファイルには注意が必要だ：
 - ある場所で内容を変更しても、他の場所では変わらないことがある

```

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 -liF file*
262156 -rw-r--r-- 2 student lfstudent 0 Mar 24 12:53 file
262156 -rw-r--r-- 2 student lfstudent 0 Mar 24 12:53 file-hard
262165 lrwxrwxrwx 1 student lfstudent 4 Mar 24 12:53 file-soft -> file
student@debian:/tmp$
  
```

図19.1: ハードリンクとソフトリンク

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われる特殊なタイプのファイルである。関連付ける（あるいは

link)はinodeを持つファイル名である：

- **ハードリンク**はinodeを指す。**ln**をオプションなしで使用することで作成される。
- **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す。ソフト・リンクは **ln** に **-s** オプションがある。

ディレクトリ・ファイルのコンテンツとinodeの各関連付けは、リンクとして知られている。**In**を使えば、さらにリンクを作ることができる。

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）を指すことは可能であり（かなり一般的である）、ファイルは複数の名前で見られることができ、それぞれがディレクトリ構造の中で独自の場所を持つ。しかし、どの名前が使われていても、inodeは1つしか持つことができない。

プロセスがパス名を参照するとき、カーネルは対応するinode番号を見つけるためにディレクトリを検索する。名前がinode番号に変換された後、inodeはメモリにロードされ、後続のリクエストで利用される。

通常、ファイルを変更しても、同じinodeを参照するハード・リンクは壊れません。しかし、ファイルをコピーして変更し、それを置き換えたり、ファイルを削除して置き換えたりして、その過程でリンクされなくなった新しいファイルを作成する（ひどく書かれた）アプリケーションがあります。そのため、このような動作が意図されたものでない場合は、注意してください。

19.3 仮想ファイルシステム (VFS)

ブイエフエス

仮想ファイルシステム (VFS) :

- 抽象化レイヤー
- カーネルと実ファイルシステム (すべてのストレージ・メディア上) 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)** を実装している。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**抽象化レイヤーとやりとりし、VFS抽象化レイヤーがすべてのI/Oシステム・コール (読み込み、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する。

そのため、アプリケーションは、特定の実際のファイルシステムや、それが存在する物理メディアやハードウェアを考慮する必要はない。さらに、ネットワーク・ファイルシステム (**NFS**など) は透過的に扱うことができる。

これにより**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で動作することができる。この民主的な特性がLinuxの成功の大きな要因となっている。

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のアクセスを持つが、いくつかのファイルシステムは、読み取りアクセスと、おそらく実験的な書き込みアクセスしか持たない。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種類のユーザーに対して、どのようにパーミッションを指定するかについて想定する必要があり、そのような動作は、マウント操作によって影響を受ける可能性があります。カーネル・ファイルシステム以外の実装として、読み書き可能な **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もパフォーマンスが劣る。

19.4 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種類のファイルシステムに対応している。
- この民主的な柔軟性が、成功の大きな要因となっている。
- ネイティブ（カーネル）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つ傾向があるが、他のオペレーティングシステムや特殊なオペレーティングシステムからのものは、読み取りアクセスのみ、または実験的な書き込みアクセスを持つかもしれない。
- S3互換のような）**キーバリュ型**ファイルシステムの中には、すべてのファイル操作を許可しないものもあり、ユーザー空間のFUSE経由で利用できることも多い。

- よく使わ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カーネルファイルシステムの一覧は [/proc/filesystems](#) にあります。

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・タイプのリストは、こうすることで見るができる：

`cat /proc/filesystems`

ノード ブ	シスフ	ノードブ	bpf	ノードブ	オート フス
ノード ブ	tmpfs	ノードブ	パイピング	ノードブ	待ち行 列
ノード ブ	bdev proc cgroup	ノード ブ	ランプス hugetlbfs rpc_pipefs	ノード ブ	ピース トア
ノデブ		ノデブ		ノード ブ	ヒュー ズ
ノデブ		ノデブ		ノード ブ	フセ クル
ノード ブ	cgroup2	ノードブ	デブス	ノードブ	エク ストス リ
ノード ブ	クープセッ ト		スカッシュ		ー
ノード ブ	devtmpfs		ブイファッ ト		エク スト2
ノデブ	configfs debugfs		msdos exfat		ext4 ntfs3
ノデブ	トレース	ノードブ	エヌエフエ ス	ノードブ	オーバ ーレイ
ノード ブ					
ノード ブ	セキュリティ ファイル	ノードブ	エヌエフエ スフォー		
ノード ブ	ソックス	ノードブ	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**特別なファイルシステム**である。) 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のみ、そのコードがモジュールとしてロードされる。

ファイルシステムの種類

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、そのほとんどが完全な読み書きアクセスを備えている:

- **ext4: Linux** ネイティブファイルシステム (以前の **ext2** と **ext3** も)
- **XFS**: 元々は**SGI**によって作られた高性能ファイルシステム。
- **JFS**: **IBM**が開発した高性能ファイルシステム
- **Windows**ネイティブ: **fat12**、**fat16**、**fat32**、**vfat**、**ntfs**
- **proc**、**sysfs**、**de-vfs**、**debugfs**を含む、メモリ上にのみ常駐する疑似ファイルシステム。
- **NFS**、**coda**、**afs**などのネットワークファイルシステム
- その他

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、常に追加されている。以下のようなものがある:

- **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、他のオペレーティングシステム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ストラクチャを使用する特殊なファイルシステム。

デバッグファイル

Linuxは非常にモジュール化された構造になっており、この柔軟性が有効に活用されている。

19.5 ファイルシステムのジャーナリング

ファイルシステムのジャーナリング

- クラッシュや異常シャットダウンから優雅に回復
- リカバリーと修復を極めて迅速に行う
- 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**を扱う
- 多くの高度な機能を含む
- 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スムーズでないシャットダウンから、ほとんど、あるいはまったく破損することなく回復する。このことは、より多くの処理を行うという代償を伴いますが、追加的な機能強化により、その代償を相殺することができます。

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**にグループ化される。トランザクションはエラーなく、**アトミック**に完了しなければならない。そうでなければファイルシステムは変更されない。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る。

Linux では、以下のジャーナリングファイルシステムが自由に利用できる：

- **ext3**は、以前の非ジャーナル**ext2**ファイルシステムの拡張である。
- **ext4**は**ext3**を大幅に拡張したものである。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る。
ほとんどの場合
Linuxディストリビューションは、数年前からデフォルトのファイルシステムとして**ext4**を使っている。
- **reiserfs**は**Linux**で最初に使われたジャーナリング実装だったが、主導権を失い、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。
- **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。**RHEL 7はXFSをデフォルト**のファイルシステムとして採用した。
- btrfsはジャーナリングファイルシステムの中で最も新しく、現在も急速に開発が進んでいます。SUSEと**openSUSE**システム。

19.6 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チューニング、あるいは特定の機能を実装するために使用される。
- マウントポイントを持つものもある（`/proc`の`proc`や`/sys`の`sys`など）。
- マウントポイントがないものもある（`sockfs`や`pipefs`など）。
- これらの特殊なファイルシステムは、実際には真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抽象化がデータや機能を整理するのに有用であると見出したカーネル機能やサブシステムである。

Linuxは、特定のタスクのために特殊なファイルシステムを広く採用している。これらは、さまざまなカーネル・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、あるいは特定の関数の実装に特に有用である。一部を紹介する：

表 19.1: 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
ルートファイル	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する。
ヒュージェットエルビーエス	どこでも	拡張ページアクセスを提供（x86では2MBまたは4MB）
bdev	なし	ブロックデバイスに使用される。
ブロック	<code>/proc</code>	多くのカーネル構造とサブシステムへの疑似ファイルシステムアクセス。
ソックス	なし	BSDソケットで使用

tmpfs	どこでも	スワッピング、リサイズが可能なRAMディスク。
shm	なし	System V IPC Shared Memory で使用される。
パイピング	なし	パイプに使用。
binfmt misc	どこでも	様々な実行形式によって使用される。
デブス	/dev/pts	Unix98 擬似ターミナルで使用
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る。
シスフ	/シス	デバイスツリー として使用。
デバッグファイル	/sys/kernel/debug	単純なファイルアクセスのデバッグに使用する。

これらの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ほしい。つまり、ユーザー・アプリケーションはこれらのファイルシステムと対話しないが、カーネルは**VFS**レイヤーとコードを利用してそれらを使用する。

19.7 ラボ



ビデオ・デモンストレーション・リソース

使用可能なファイルシステムデモ.mp4

練習問題19.1: tmpfs特殊ファイルシステム

tmpfsは、Linuxで使われる多くの特殊なファイルシステムの1つである。これらの中には、実際にはファイルシステムとして使われるのではなく、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る。しかし、**tmpfs**はアプリケーションがI/Oを実行できる本物のファイルシステムである。

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実装にはない、いくつかの優れた特性がある：

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）を動的に調整する。ゼロから始まり、マウントされた最大サイズまで必要に応じて拡張する。
2. RAMを使い果たした場合、**tmpfs**はスワップ領域を利用できる。(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ない)。
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い。tmpfsは、ファイルやI/Oを処理するための独自のメソッドを持っており、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ている（実際にはブロックデバイスではない）ため、スピードのために最適化されている。

したがって、mkfsコマンドでファイルシステムを事前にフォーマットする必要はない。

tmpfsの新しいインスタンスを、以下のようなコマンドでディレクトリ構造の任意の場所にマウントする：

```
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する：

```
df -h /mnt/tmpfs
```

デフォルト値としてRAMの半分が割り当てられていることがわかるはずだ。しかし、使用量はゼロで、**/mnt/tmpfs**にファイルを置くにつれて増えていくだけだ。

というように、マウントオプションとして割り当てサイズを変更することができる：

```
sudo mount -t tmpfs -o size=1G none /mnt/tmpfs
```

容量が一杯になるまで入れてみて、どうなるか試してみるといいだろう。使い終わったらアンマウントするのを忘れずに：

```
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する：

```
df -h /dev/shm
```

ファイル	タイプ tmpfs にマウント 3. 9G24M 3.
システム	9G1% /dev/shm
tmpfs	

多くのアプリケーションは、プロセス間通信の仕組みとして**POSIX**共有メモリーを使用する場合などにこれを使用する。どのユーザーも**/dev/shm**にファイルを作成し、読み書きできるので、メモリ上に一時ファイルを作成するには良い場所である。

`/dev/shm`にいくつかファイルを作成し、**df**でファイルシステムがどのように埋まっていくかを確認する。

さらに、多くのディストリビューションは、**tmpfs**の複数のインスタンスをマウントする：

```
df -h | grep ' tmpfs '
```

tmpfs	tmpfs	7.8G	38M	7.8G	1% /dev/shm
tmpfs	tmpfs	7.8G	18M	7.8G	1% /run
tmpfs	tmpfs	7.8G	0	7.8G	0% /sys/fs/cgroup
tmpfs	tmpfs	1.6G	1.2M	1.6G	1% /run/user/42
tmpfs	tmpfs	1.6G	56K	1.6G	1% /run/user/1000

これは16GBのRAMを搭載したシステムで実行されたものであるため、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～8GBを実際に使用しているわけではないことは明らかだ！



ご注意

ディストリビューションによっては（**Fedora**など）、**/tmp**を（デフォルトで）**tmpfs**システムとしてマウントするものもある。そのような場合は、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らない。あるいは、先に**/tmp**について説明したように、この動作を無効にすることもできる。

第20章

ディスク・パーティショニング



20.1	一般的なディスクの種類	300
20.2	ディスク・ジオメトリー	301
20.3	パーティショニング	302
20.4	パーティション・テーブル	304
20.5	ディスクデバイスの命名	306
20.6	blkidとlsblk	307
20.7	パーティションのサイズアップ	309
20.8	パーティションテーブルのバックアップと復元	310
20.9	パーティションテーブルエディター	312
20.10	エフディスク	313
20.11	研究室	314

学習目標

このセッションを終えるまでに、あなたは次のことができるようになるはずだ：

- 最も一般的なタイプのハードディスクとデータバスについて説明し、対比させる。
- ディスクのジオメトリと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と、関連するデバイスノードの識別方法を理解する。
- fdisk、blkid、lsblkなどのユーティリティを使う。
- 異なるパーティショニング戦略を区別し、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。

20.1 一般的なディスクの種類

一般的なディスクの種類

- **SATA** (シリアルATアタッチメント)
 - より高速なデータ転送、**IDEより**小さなケーブル
 - **SCSI**デバイスとして見られる
- **SCSI** (小型コンピュータ・システム・インターフェース)
 - 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
- **SAS** (シリアルアタッチドSCSI)
 - 新しいポイント・ツー・ポイントのシリアル・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とフロッピーを含む
 - SCSIデバイスとして見られる
- **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**および**EIDE** (統合ドライブ・エレクトロニクス、エンハンスドIDE)
 - 廃止

ハードディスクにはさまざまなタイプがあり、それぞれ接続される**データバス**のタイプや、速度、容量、複数のドライブが同時にどの程度動作するかなどの要素によって特徴付けられる。

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る。その違いについては、<https://store.hp.com/us/en/tech-takes/sas-vs-sata> を参照されたい。

SCSIディスクには、ナロー (8ビットバス) からワイド (16ビットバス) まであり、転送速度は毎秒約5MB (ナロー、標準**SCSI**) から毎秒約160MB (**Ultra-Wide SCSI-3**) までである。

最新の**SSD**ドライブ (ソリッド・ステート・ドライブ) 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

ブよりも消費電力が少なく、転送速度が速くなっている。内蔵SSDは、従来のドライブと同じフォーム・ファクターで、同じ筐体に取り付けることさえできる。

SSDはまだかなり高価だが、価格は下がっている。SSDと回転式ドライブの両方を同じマシンに搭載し、頻繁にアクセスされ、パフォーマンスが重要なデータ転送は**SSDで行うのが一般的**です。

20.2 ディスクの形状

ディスクの形状

- 回転ディスクは1つまたは複数のプラッターで構成され、各プラッターは1つまたは複数のヘッドで読み取られる。
- ディスクが回転すると、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
- 元来、セクタは512バイトのデータだったが、現在は通常4096バイト
- シリンダーとは、すべてのプラッター上の同じトラックで構成されるグループのこと。
- SSDでは、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`

ディスク /dev/sdc: 1.8TiB、2000398934016バイト、3907029168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/4096バイト

物理的な構造イメージは、ドライブの内部エレクトロニクスがその多くを実際に見えなくしているため、あまり意味を持たなくなっている。さらに、SSDには可動部品もなければ、上記のような成分もない。

`fdisk`でジオメトリを見ることができる:

```
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>
```

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の一覧を表示します。

歴史的にディスクは512バイトのセクタで製造されてきたが、現在では4KBが圧倒的に一般的である。**Linuxでは、**後方互換性のために今でも512バイトの論理セクタ・サイズを使用しているが、これは単にソフトウェアでふりをするためである。

20.3 パーティショニング

パーティション組織

- ディスクはパーティションに分割される
- パーティションとは、ディスク上の物理的に連続した領域のこと。
- 最も一般的なアーキテクチャでは、2つのパーティショニング・スキームが使われている：

- **MBR**（マスターブートレコード）
- **GPT**（GUIDパーティションテーブル）

- **MBRはMSDOSの初期にさかのぼる**

最大4つのプライマリパーティションを持つことができ、プライマリパーティションの1つを**拡張**パーティションとして指定し、15パーティションの**論理**パーティションに細分化することができる。

- 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）に基づいています。

デフォルトでは、プライマリ・パーティションは最大128個まで設定可能

- **MBRスキームを使用する場合：**

SCSIドライブ、例えば`/dev/sda`がある場合、`/dev/sda1`が第1プライマリパーティションで、`/dev/sda2`が第2プライマリパーティションになります。拡張パーティション`/dev/sda3`を作成すると、論理パーティションに分割されます。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。

メモ: **Linux**ではパーティションがシリンダー境界で始まったり終わったりする必要はありませんが、他のオペレーティング・システムでは、パーティションがシリンダー境界で始まったり終わったりしないと文句を言われることがあります。このため、広く普及している**Linux**のパーティション分割ユーティリティは、境界で終了するように努めて

います。もちろん、パーティションが重なってはいけません。

- **GPT**スキームを使用する場合:

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。パーティションのサイズは最大²³³TBまで！（MBRの場合は2TBまで）。

なぜ分割なのか？

- 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**
- 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と権限を課すことによる**セキュリティ強化**
- **サイズへの懸念**：可変ストレージと揮発性ストレージを静的ストレージから分離する。
- 使用頻度の高いデータをより高速なストレージメディアに置くことによる**パフォーマンス向上**
- **スワップ領域**はデータから分離でき、ハイバネーション・ストレージとしても使用できる。

何をパーティション分割し、どのようにパーティション分割するかは、考えどころです。パーティションを分ける理由には、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などがあります。また、データ保護の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には、`/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があります。

パーティションのサイズを後から変更するのは、インストール中や作成中よりも難しいことを覚えておってください。それなりの計画を立ててください。

ング

20.4 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、最初のセクターの最後の64バイト、**MBR**（マスター・ブート・レコード）の512バイトにある。

- 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つを記述する。それぞれの情報は以下の通りである：
示す：

Linux、Linux LVM、ntfs、swap

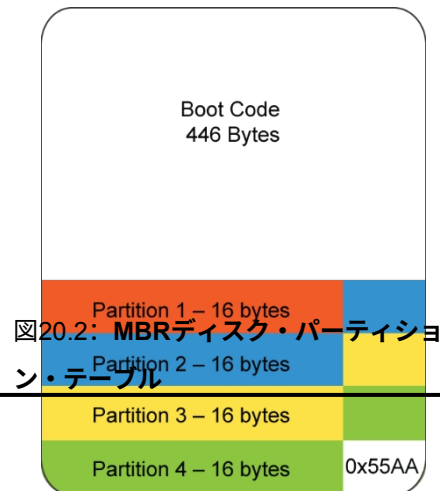
— 各タイプ・ビット

— 開始の終了住所レシ で シリンダ

— シ・ヘッド/セクタ(CHS)フォーマット
— スタート・セクター、0からリニ
ット
アにカウント

— パーティション内のセクタ数

Linuxは、リニア・ブロック・アドレスリング（**LBA**）方式によるアドレス指定に最後の2つのフィールドのみを使用する。



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、アクティブとマークされているかもしれません。システムが起動すると、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となる。

拡張パーティションは1つしか存在できませんが、そのパーティションにはいくつもの論理パーティションが含まれる可能性があることを覚えておいてください。

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る。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エントリーのパーティションテーブル用のスペースです。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する。

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る。

(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・マーカースとして知られる**MBR**の末尾にはさらに2バイトあり、常に0x55AAという値を持っていることに注意してほしい)。

ーブル

GPTパーティションテーブル

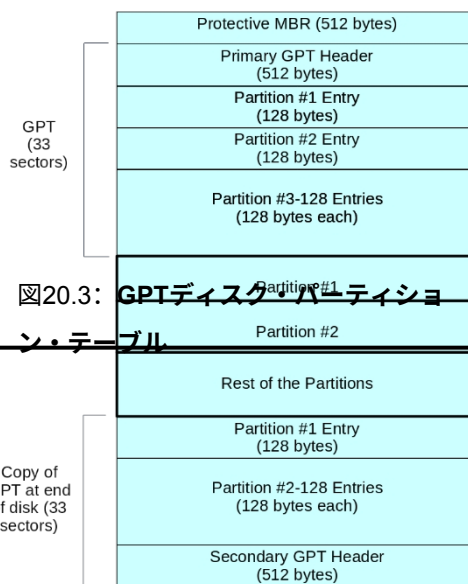
- 最近のハードウェアは**GPT**をサポートしている。
- 保護MBRは後方互換性のためで、UEFIシステムは古い方法で起動できる。
- ディスクの最初と最後に、メタデータを記述する**GPT**ヘッダーのコピーが2つある：

– ディスク上の使用可能ブロックのリスト

– パーティション数

– パーティション・エントリーのサイズ

- 各パーティション・エントリーの最小サイズは128バイト



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する：

- 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システムの場合：

```
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^グシステムを表す一意なUUIDを与えてい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パーティションはパーティションを記述するPARTUUIDも与え、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す。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させることは難しくありません。そのため、通常はリスクに見合うメリットがありません。

20.5 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

- `dev`ディレクトリにデバイスファイルが見つかった
- 名前の形式: `xx[y][z]`
 - `xx` はデバイスのタイプ（通常は `sd`）です。
 - `y` はドライブ番号の文字（`a, b, c` など）。
 - `z` はパーティション番号
- サンプル:
 - `/dev/sda` が最初に見つかったドライブです
 - `/dev/sdb2` は2番目のドライブの2番目のパーティションである。

Linuxカーネルは、通常`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やりとりする。通常、デバイス・ノードへのアクセスは、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行われる。

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る：

```
sudo mkfs.ext4 /dev/sda9
```

SCSIおよびSATAディスクのデバイスノードは、単純な命名規則に従う：

- 最初のハードディスクは`/dev/sda`である。
- 番目のハードディスクは`/dev/sdb`である。
- その他

のように、パーティションも簡単に列挙できる：

- `/dev/sdb1`は2番目のディスクの最初のパーティションである。

- `/dev/sdc4`は3番目のディスクの4番目のパーティションである。

グ

上記のsdは**SCSI**または**SATA**ディスクを意味する。**IDE**ディスクが見つかった時代には、`/dev/hda3`だったはずだ、`/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・ノードが表示される。

20.6 blkidとlsblk

ブロックキッド

- 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
- ブロック装置の内容を報告
- ブロックデバイスの属性に関するレポート
- サンプルコマンド:

```
sudo blkid
sudo blkid /dev/sda*
sudo blkid -L root
```

blkid は、ブロック・デバイスを検索し、その属性を報告するユーティリティです。**libblkid** ライブラリで動作する。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のタイプ（ファイルシステムやスワップなど）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る。

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、ブロックを識別する**UUID**を生成しません。

blkidには、特定のNAME=値のペアを持つデバイスを検索するか、1つ以上のデバイスのNAME=値のペアを表示するかの2つの主な操作形態がある。

引数がない場合は、すべてのデバイスについてレポートする。デバイスをどのように指定し、どのような属性を報告するかを指定するオプションがかなりある。

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"
v /dev/sda6: LABEL="SWAP" UUID="6a045706-62dc-4ca4-bc54-e07e41141585" TYPE="swap"
c7:/tmp>
```

図20.4: blkidの使用

グ

エクスブロック

- ブロックデバイス情報をツリー形式で表示

```

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]
c7:/tmp>

```

図20.5: lsblkの使用

lsblk -h

使い方:

lsblk [オプション] [<デバイス> ...]。

ブロックデバイスの情報を一覧表示し

ます。オプション

- a, --all 全てのデバイスを表示
- b, --bytes SIZEを可読形式で表示するのではなく、バイト数で表示する。
- d, --nodeps スレーブやホルダーを表示しない
- D, --disk-cards 印刷破棄機能
- z, --zone 印刷ゾーンモデル
- 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 アスキー文字のみを使用
- I, --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出力形式を使う
- l, --list リスト形式の出力を使う
- T, --tree ツリー形式出力を使う
- m, --perms パーミッションに関する情報を出力する
- n, --no-show 見出しを印刷しない
- o, --output <リスト> 出力列
- O, --output-all 全ての列を出力する。
- p, --paths デバイスパス全体を表示する
- P, --pairs key="value"の出力形式を使う
- r, --raw 生の出力フォーマットを使う
- s, --inverse 逆依存
- S, --scsi SCSI デバイスに関する情報を出力する。
- t, --topology トポロジーに関する情報を出力する
- x, --sort <column> 出力を <column> でソートする。

- h, --help ヘルプを表示する
- V, --version バージョンを表示

利用可能な出力列:

.....
詳細は lsblk(8) を参照。

20.7 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する：

ルート

- ファイルシステムに使用
- ほとんどのLinuxは、複数のファイルシステムを持っている。
- サイズの変更がやや不便（可能であればLVMを使用できる）

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サイズに等しい
- 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ってもよい。

- Ubuntuを含むいくつかのディストリビューションでは、デフォルトでパーティションではなくスワップファイルを使用する：

- より柔軟性がある（例えば、サイズ変更がより簡単になる）
- しかし、エラーやバグが蔓延すると、より危険な状態になる可能性がある。

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のは困難です。

ルート・パーティションだけでLinuxを実行することは確かに可能だが、ほとんどのシステムでは、バックアップを簡単にし、ディスクドライブをより効率的に使用し、セキュリティを向上させるために、より多くのパーティションを使用している。

通常、スワップ・サイズは物理メモリと同サイズが推奨されるが、その2倍が推奨されることもある。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に関連する問題に依存する。

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、必ずしも助けにはならない。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る。

古い回転式ハードディスク・ドライブ・メディアでは、スワップ・パーティションを別にする方が理にかなっているかもしれないが、SSD**タイプ**のメディアでは、これは重要ではない。しかし、より低速でおそらくより安価なハードウェアにスワップを置きたい場合もあるだろう。これは、パーティションを使用する場合でも、ファイルを使用する場合でも同じです。

現在、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）を使用していることに注意してください。zram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。これは簡単にメモリ不足に陥りますが、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。

20.8 パーティションテーブルのバックアップと復元

パーティションテーブルのバックアップと復元

- マスタブートレコード

- **dd**プログラムを使う：
- 以下のコマンドは、**MBR**を（パーティションテーブルとともに）バックアップする。

```
dd if=/dev/sda of=mbrbackup bs=512 count=1
```

- **MBR**は以下の方法で復元できる。

```
sudo dd if=mbrbackup of=/dev/sda bs=512 count=1
```

- ジーピーティー

- **MBR**システムでも使用できる**sgdisk**を使用する。

```
sudo sgdisk --backup-file=gptbackup /dev/sda  
sudo sgdisk --load-backup-file=gptbackup /dev/sda
```

20.8. パーティションテーブルのバックアップと復元

グ

311

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すべて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（そんなことはないはずですが、用心してください！）。

したがって、この種の作業を行う前に、（まだバックアップされていない）すべてのデータのバックアップをとっておくことが常に賢明である。**MBRシステム**の場合、**dd**はファイルの変換とコピーに使用できる。**dd**の使用には注意が必要である：単純なタイプミスやオプションの使い方を誤ると、ディスク全体が破壊される可能性がある。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のみをコピーします。他のパーティション（拡張パーティションなど）に格納されているパーティションテーブルは扱いません。

GPTシステムの場合は、以下のように**sgdisk**ツールを使うのがベストだ：

x8:/tmp>sudo sgdisk -p /dev/sda

```
ディスク /dev/sda: 1000215216 セクタ、476.9 GiB
モデル: サムスンMZNLN512
セクタサイズ (論理/物理) : 512/512バイト
ディスク識別子 (GUID) : DBDBE747-8392-4B62-97AA-
6214490073DC
パーティション・テーブルは最大128エントリ
.....
1          2048          534527    260.0 MiB  EF00 EFIシステムパーティ
番号 開始 (セクタ)      サイズ      ション 567295      16.0 MiB      0C01 マイ
.....
          クロソフト予約済み...
```

純粋な**MBRシステム**で実行する場合の注意：

c8:/tmp>sudo sgdisk -p /dev/sda

```
*****
*** 無効なGPTと有効なMBRが見つかりました。メモリ内でMBRを
GPT形式に変換します。
*****
***** ディスク /dev/sda: 500118192セクター、238.5GiB
モデル: クルージャ CT256MX1
セクタサイズ (論理/物理) : 512/4096バイト
.....
```


20.9 パーティション・テーブル・エディター

パーティションテーブルエディター

- **fdisk**: 標準、メニュー指向
- **sfdisk**: スクリプトまたはコマンドラインインターフェイス
- partedGNU パーティション操作プログラム
- **gparted**: **parted**のGUIベースバージョン
- **gdisk**: **GPT** システム用; **MBR** システムでも動作可能
- **sgdisk**: スクリプトまたはコマンドラインインターフェイス

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の中で最も標準的で、最も柔軟なエディタです。他のパーティションテーブルエディタと同様、変更を加える前に、現在のパーティションテーブル設定を書き留めるか、現在の設定のコピーを取るようにしてください。

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、スクリプト作成に便利です。**sfdisk** ツールは慎重に。

GPTシステムでは、**gdisk**と**sgdisk**が同様の役割を果たす。どちらも**MBR**システムでも動作する。

parted は GNU 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、移動ができます。**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、**gparted**のコピーを含むCDROMやUSBメモリから実行できる**ライブインストール**バージョンがあり、パーティション分割プログラムの実行中に実際に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ン分割ツールとして簡単に使用できます。

20.10 エフディスク

fdiskの使用

- ベース・インストールの一部で、使いやすい
- メニュー指向で、プロンプトに従うだけ：
 - m:メニューを表示します。
 - p:パーティションテーブルをリストアップする。
 - 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情報を書き込み、終了する。
 - 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になる必要がある。扱いがやや複雑になることがあるので、注意が必要である。**fdisk**のインターフェースはテキスト・メニュー駆動である。のように特定のディスクで起動した後

```
sudo fdisk /dev/sdb
```

という一文字コマンドを発行することができる。

幸いなことに、wを入力してパーティションテーブルをディスクに書き込むまでは、実際の変更は行われない。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ことを（pで）確認することが重要である。

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。しかし、以下のコマンドを使うことができます：

```
sudo partprobe -s
```

314 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

第20章 ディスク・パフォーマンス

なく、パーティションを取り違えると大惨事になる可能性があるため、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。

いつでもできる：

[cat /proc/partitions](#)

をクリックして、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ます。

20.11 ラボ



ビデオ・デモンストレーション・リソース

[_fdisk_demo.mp4](#)を使って

練習20.0: ループバックデバイスを使っ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を簡単に使用します。これらのコマンドは次のセッションで詳しく説明します。

このセクションの最初の3つの練習では、パート分けされたプログラムの有無にかかわらず、**ループ装置**メカニズムを



非常に重要

このコースの後の練習のために、パーティション分割されていないディスク領域が必要です。GB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、それがあつかないかです。ない場合は、パーティションとそ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手順やこれから説明する手順を使って、利用できる



ご注意

実際の物理的な未パーティションディスク領域がある場合は、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20.1: ファイルをディスクパーティションイメージとして使う

この最初のエクササイズでは、フルハードディスクパーティションイメージのコンテナとして使用するファイルを作成します。次のエクササイズでは、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のゼロばかりのファイルを作る：

```
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空き容量がない場合などには、もっと小さなファイルを作成することもできます。

2. その上にファイルシステムを置く：

```
mkfs.ext4 イメージファイル
```

```
mke2fs 1.42.9 (2013-12-28)
imagefileはブロック特殊デバイスではな
い。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、別のファイルシステムでフォーマットすることもできる。

グ

3. それをどこかに取り付ける:

```
mkdir mntpoint
sudo mount -o loop imagefile mntpoint
```

これで思う存分、ファイルなどを置いて使うことができる。

4. 終わったら、アンマウントする:

```
sudo umount mntpoint
```

ループ・オプションを使ってマウントする以外の方法としては、次のようなものがある:

```
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
```

また、`/dev/loop[0-7]`を使うこともできるが、これから説明するように、それらがすでに使われていないことに注意しなければならない。

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うのは便利ですが、計測やベンチマークを行う上ではかなり無価値であることに注意する必要があります。というのも、ファイルシステムのレイヤーを別のレイヤーの上に重ねることになり、パフォーマンスに悪影響を及ぼすだけだからです。

🔪練習20.2 ディスクイメージファイルのパーティション分割

次のレベルの複雑さは、コンテナファイルを複数のパーティションに分割することで、各パーティションはファイルシステムやスワップ領域を保持するために使用できる。

前の練習で作成した画像ファイルを再利用するか、新しく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 imagefile
```

デバイスに認識された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、書き込むことを決定するまで、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 (ヘルプは m) :

C 130は、ドライブの偽シリンダーの数を設定するもので、古いバージョンの**fdisk**でのみ必要で、残念ながら**RHEL 6**

で見つかる。

2. mと入力すると、コマンドのリストが表示される：

m

コマンド（ヘルプはm）：

m コマンドアクション

a ブート可能フラグを切り替える

b ディスクラベルを編集する

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

グ

G IRIX (SGI) パーティションテーブルの作成
 l 既知のパーティション・タイプのリスト
 m このメニューを印刷する
 n 新しいパーティションを追加する
 o 新しい空のDOSパーティションテーブルを作成する
 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 空のサン・ディスクラベルを新規作成する
 t パーティションのシステムIDを変更する
 u 表示／入力単位を変更する
 r 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み、終了する

x コマンド (m for help):

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）にする：

コマンド (mはヘルプ) : n

パーティションタイプ:

p プライマリ (プライマリ0、エクステン

デッド0、フリー4) e 拡張

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パーティション1、タイプはLinux、サイズは256MiB。

4. 2番目のプライマリパーティションも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:

p プライマリ (プライマリ1、エクステン

デッド0、フリー3) e 拡張

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

れている。

コマンド (ヘルプはm) : p

イメージファ イル1	2048	526335	262144 ス	83 リナック
イメージファ イル2	526336	1050623	262144 クス	83 リナッ

5. パーティションテーブルをディスクに書き込み、終了する：

コマンド (ヘルプは m) : 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、作成したばかりの2つのパーティションを使う方法はまだ見ていません。次の練習では、それを可能にする方法からやり直そう。

練習20.3: `losetup`と`parted`の使用

これからもっと実験していくつもりだ:

- ループ装置と`losetup`
- `parted`はコマンドラインでパーティション分割を非対話的に行う。

以下の手順を実行する前に、`losetup`と`parted`のマニュアル・ページを読んでいただきたい。もう一度言うが、

イメージ・ファイルを再利用することもできるし、ゼロにして新たに、あるいは別のファイルから始めることもできる。

1. 画像ファイルをループ装置に関連付ける:

```
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・デバイスを見つける。このようにする理由は、すでに1つ以上のループ・デバイスを使用している可能性があるからです。例えば、上記のコマンドが実行される前に、この文章が書かれているシステム上で:

```
losetup -a
```

```
/dev/loop0: []: (/usr/src/KERNELS.sqfs)
```

もしこれを無視して`/dev/loop0`に`losetup`を使ったら、ほぼ間違いなくファイルが壊れてしまうだろう。

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する:

```
sudo parted -s /dev/loop1 mklabel msdos
```

3. ループデバイス上に3つのプライマリパーティションを作成する:

```
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する:

```
fdisk -l /dev/loop1
```

```
ディスク /dev/loop1: 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
```

グ

デバイス識別子	スタート	終了	ブロック IDシステム
/dev/loop1p1	0x00050c11	500000	250000 83 リナックス
/dev/loop1p2	500001	1000000	250000 83 リナックス
/dev/loop1p3	1000001	2000000	500000 83 リナックス

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よって異なります。例えば、RHELやUbuntuでは、新しいデバイスノードが作成されていることがわかる：

```
$ ls -l /dev/loop1*
```

```
brw-rw---- 1 root disk    7, 1 10月 7 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
```

以下ではそれらを使用する。

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すべてをマウントし、利用可能であることを示す：

```
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
/dev/sda1	ext4	29G	8.210万ドル	5G19G 32%/インチ

/dev/loop1p1	ext3	233M	億1900万ドル	1% mnt1
/dev/loop1p2	ext4	233M	210万ドル	1% mnt2
/dev/loop1p3	vfat	489M	215万ドル	0% mnt3
			0 489M	

8. 心ゆくまでファイルシステムを使った後は、すべてを解き放つことができる：

```
sudo umount mnt1 mnt2 mnt3
rmdir mnt1 mnt2 mnt3
sudo losetup -d /dev/loop1
```

練習問題 20.4: 実ハードディスクのパーティション分割

ハードディスクにパーティション未設定の領域がある場合、**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してみてください。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21章

ファイルシステムの機能：属性、作成、チェック、使用、マウント



21.1	拡張属性	320
21.2	ファイルシステムの作成とフォーマット	321
21.3	ファイルシステムのトラブルシューティング	322
21.4	ファイルシステムのチェックと修復	324
21.5	ファイルシステムの使用	325
21.6	ディスク使用量	326
21.7	ファイルシステムのマウント	327
21.8	NFS	331
21.9	ブートと/etc/fstabでのマウント	332
21.10	オートマウント	334
21.11	ネットワーク・ブロック・デバイス	336
21.12	研究室	340

学習目標

このセッションを終えるまでに、あなたは次のことができるようになるはずだ：

- **lsattr**と**chattr**を使った拡張ファイル属性の使い方を学ぶ。
- **mkfs**と**fsck**を使って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のトラブルシューティング
- ファイルシステムのエラーをチェックし、修正する。
- ディスク容量の使用状況を確認するには、ユーティリティの**df**と**du**を使用する。
- ファイルシステムのマウントとアンマウント。
- **NFS**などのネットワークファイルシステムを使用する。
- 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。

21.1 拡張属性

lsattrとchattr

- ファイルに拡張属性を設定できる
 - i: 不変フラグ
 - a: 追加のみフラグ
 - d: ダンプフラグなし
 - A: 時刻更新フラグなし
- フラグは **lsattr** コマンドで確認できる。
- フラグは **chattr** コマンドで設定
- フラグはファイルのinodeフラグに格納される。
- ルートによってのみ設定可能

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る。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**で使われる。

フラグ値はファイルのinodeに保存され、rootユーザーのみが変更・設定できる。これらは **lsattr** で表示され、**chattr** で設定される。フラグはファイルに対して設定することができる：

- **i（不変）**：i（不変）：（rootであっても）変更、削除、名前の変更ができない。このファイルへのハード・リンクを作成したり、データを書き込んだりすることはできません。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のみです。
- **a（追記専用）**：（追記専用）：書き込みのために追記モードでのみオープンできる。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る。バックアップに時間を費やしたくないスワップファイルやキャッシュファイルに便利である。

- A (**atime更新なし**) : A (atime更新なし) : ファイルがアクセスされても変更されない場合、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の量が減るため、システムによってはパフォーマンスが向上する。

他にも設定できるフラグがあることに注意。man chattrと入力すると、すべてのリストが

表示される。**chattr**のフォーマットは

chattr [+|-|=mode] ファイル名

lsattrはファイルの属性を表示するのに使われる:

lsattr ファイル名

21.2 ファイルシステムの作成とフォーマット

mkfs

- **mkfs**の一般的なフォーマット:

```
mkfs [-t fstype] [オプション] [デバイスファイル]
```

- **[device-file]**は通常、**/dev/sda3**や**/dev/vg/ lvm1**のようなデバイス名である。
- 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・オプションがある。
- 各ファイルシステムは、それぞれ独自の**mkfs**プログラムを持っている。
- 同等のコマンド:

```
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である。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すぎず、それぞれのプログラムには特定のオプションがあるかもしれない:

```
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 20 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*
student@opensuse:~>
```

図 21.1: mkfs

詳細については、各mkfs.*プログラムのマニュアル・ページを見てほしい。

21.3 ファイルシステムのトラブルシューティング

ファイルの完全性のチェック

- 破損した設定ファイルやバイナリをチェックする
- 多くのメソッドを使用可能

- 回転数

`rpm -V some_package`

単一のパッケージをチェックする

`rpm -Va`

すべてのパッケージをチェックする

- 補佐官：侵入検知

`$ aide --check`

- デブサムズDebian

`debsums オプション 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る。一つの方法は、`rpm -V` コマンドを使って特定のパッケージをチェックすることである。`rpm -Va` を使ってすべてのパッケージの完全性をチェックすることもできます。

ファイルの変更をチェックするには、**aide**を使う方法もある。コマンド `aide --check` はファイルのスキャンを実行し、前回のスキャンと比較する。

Debianでは、完全性チェックを行う唯一の方法は**debsums**です。`debsums somepackage`を実行すると、そのパッケージ内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われる。
- `etc/fstab` に設定ミスがないかチェックする。
- ランニングが必要な場合がある

`マウント -o リマウント,rw /`

修正を試みる

- ファイルシステムを手動でマウントしてみる

fsckを使って修復を試みることができる。しかし、その前に、`/etc/fstab`が誤って設定されたり破損したりしていないことを確認すべきである。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、もう一度注意してほしい。

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るが、`/`は読み取り専用でマウントされている可能性がある：

```
sudo mount -o remount,rw /
```

を書き込み許可で再マウントする。

`etc/fstab`が正しいようであれば、**fsck**に移ることができる。まず最初に

```
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る。これが完全に成功しない場合は、問題のあるものを手動でマウントしてみることができる。最初に**fsck**を実行し、調べるだけにしてください。その後、再度fsckを実行し、見つかったエラーを修正することができます。

21.4 ファイルシステムのチェックと修復

ファスク

- 設定した期間またはマウント数の後に自動的に実行する
- 事前にクリーンマウントされていない場合、ブート時に実行されます。
- マウントされたファイルシステム上で実行してはならない
- **fsck**の一般的なフォーマット:

`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`のようなデバイス名である。

- パーティションのスーパーブロックが検査されるため、ファイルシステムタイプは通常必要ない
- 同等のコマンド:

`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かったエラーを修正する）よう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である。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*
```

用法

図 21.2: **fsck**

こうすることで、ブート時にマウントされている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る:

```
sudo touch /forcefsck
```

```
sudo リブート
```

チェックが成功すると、`/forcefsck` は消えます。

見つかったエラーをひとつずつ手動で修正するかどうかは、`-r` オプションで制御できる。さらに、ファイルシステムの種類ごとに、独自の **fsck** オプションがある。

21.5 ファイルシステムの使用

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のに使われる。
- ファイルシステムの使用量を表示する (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する:
`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21.3: dfの使用

21.6 ディスク使用量

デュディスクの使用状況

- du（ディスク使用量）は、ディスク容量と使用量の両方を調べるために使用される。

- 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する：

\$ 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する：

\$ du -h somedir

次のコマンドを試してみよう：

\$ find .-maxdepth 1 -type d -exec du -shx {} \; | ソート -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21.4: duの使用

21.7 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る：

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ない。
- 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`）にあります。
- ファイルシステムは現在のディレクトリツリーの`/home`にマウントされている。
- パーティションがアンマウントされるまで、元の`/home`ディレクトリにあるファイルはすべて隠される。

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つの大きな階層ツリー構造に編成されている。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（それぞれ独自のファイルシステム・タイプを持つことができる）が結合されていることがよくあります。これらのパーティションは、ネットワーク上であっても、異なる物理デバイス上にあることもあります。

mountプログラムは、ツリー構造のどのポイントでも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ーである。**mkdir**を使用して空のディレクトリを作成することができます。**mkdir**を使用して空のディレクトリを作成することができます。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、それらはマウント後に非表示になります。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見えるようになります。

ファイルシステムのマウントとアンマウントができるのはスーパーユーザーのみである。

マウント

• マウントの一般的なフォーム

`mount [options] <ソース> <ディレクトリ>`

• を使用してマウントできます：

デバイスノード、ラベル、UUID

• ファイルシステムにはそれぞれ独自のオプションがある。よくある例としては

`sudo mount -o remount,ro /myfs`

```

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
....
c7:/tmp>
  
```

図21.5: マウント

これは、読み取り専用属性を持つファイルシステムを再マウントする。

以下の取り付け方法はすべて同等である：

```

sudo mount /dev/sda2          /ホーム
sudo mount LABEL=home /home
sudo マウント    -ホーム/ホーム
sudo mount UUID=26d58ee2-9d20-4dc7-b6ab-aa87c3cfb69a /home
sudo マウント    -U 26d58ee2-9d20-4dc7-b6ab-aa87c3cfb69a /home
  
```

ラベルは**e2label**のようなファイルシステムタイプに特化したユーティリティによって割り当てられ、UUIDはパーティションがファイルシステムのコンテナとして作成され、**mkfs**でフォーマットされるときに割り当てられる。

デバイスを指定するこれら3つの方法のどれを使ってもよいが、最近のシステムでは、システムの起動方法やどのハードデ

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・ノード形式を使うことは推奨されていない。

ラベルは改善されましたが、まれに2つのパーティションが同じラベルになってしまうことがあります。しかし、**UUID**は常に一意であるべきで、パーティションの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ステム

```

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/systemd/systemd-cgrou
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21.6: 現在マウントされているファイルシステム

上記では、3種類の基本的なファイルシステムを見てきた:

- カーネルスペース - ブロックデバイスによってバックアップされる。

エクステンデッドフォー

- ユーザー空間 - ブロック・デバイスによってバックアップされる

vmware-vmblock - VMWare ボリュームを共有するための **FUSE** ファイルシステム。

gvfsd-fuse - GNOME でマウントを扱うための **FUSE** ファイルシステム

- **psuedo** - ブロックデバイスによってバックアップされていない（しかし、一般的にカーネル空間でもある）。

sysfs、**procfs**、**devtmpfs**、**devtmpfs** - コア機能の提供

debugfs、**hugetlbfs**、**cgroup**、**sunrpc** - ネットワークファイルシステムのサポートを含む追加機能を提供します。

アンマウント

- 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:

`umount [デバイス・ファイル | マウント・ポイント]`。

- 例を挙げよう:

- **home** ファイルシステムをアンマウントする:

```
$ 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**であることに注意（**unmount**ではない!）。

mountと同様、**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固有である。繰り返しますが、具体的なオプション情報については **man** ページが最良の情報源です。

ファイルシステムをアンマウントするとき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を行おうとした場合である。

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だ。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、アンマウントできるようになる。

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、そのユーザーを殺すことができる（これには注意が必要で、最初にユーザーに警告を出したほうがよいかもしれない）。**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る。

21.8 ネットワークファイルシステム

ネットワーク共有 (NFS)

- 最も一般的なネットワーク共有は**NFS**
- その他、**AFS**、**SMB (CIFS)** などがある。
- のようにマウントする：

```
$ sudo mount -t nfs myserver.com:/shdir /mnt/shdir
```
- この行を`/etc/fstab`に記述し、ブート時または`mount -a`でマウントする：

```
myserver.com:/shdir /mnt/shdir nfs rsize=8192,wsiz=8192,timeo=14,intr 0 0
```
- ネットワークが立ち上がる前に、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かもしれない。netdevオプションとnoautoオプションが使えます（man nfs, examine **mount** options）。
- **autofs**または**automount**を使用しても解決できます。
- `mount`には多くのオプションがあり、**nfs**に特有のものもある。詳細については、nfsと**mount**の両方のマニュアルページを参照のこと。

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し、あたかもローカルマシンにあるかのように見せるのが一般的だ。歴史的に最もよく使われてきたのは、おそらく**NFS**（ネットワーク・ファイル・システム）だろう。

NFSはもともと1989年にサン・マイクロシステムズ社によって開発され、継続的に更新されてきた。最近のシステムでは**NFSv4**は2000年以降、継続的に更新されている。

その他のネットワーク・ファイルシステムには、**AFS**（アンドリュー・ファイル・システム）や**SMB**（サーバー・メッセージ・ブロック）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かなければならない。

したがって、このような状況では、指定された期間以上待機している間、システムがハングしないように、あるいはブロックされないように指示すべきである。これらはマウントコマンドが`/etc/fstab`で指定できる。

21.9 ブートと/etc/fstabでのマウント

ブーツの取り付け

- システムの初期化中に

マウント -a

は、`/etc/fstab` にリストされているすべてのファイルシステムをマウントするために実行される。

- 起動時にマウントするファイルシステムとして、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を指定できる。

`/etc/fstab` は、起動時にマウント可能なファイルシステムとデバイスを定義するために使用される。そこで、どのファイルシステムがマウント可能か、ローカルでどこにあるか、誰がどのようなパーミッションでマウントできるかを確認する。 `etc/fstab` にファイルやディレクトリが指定されていれば、起動時にマウントされる可能性がある。

これには、NFSやsambaファイルシステムなど、ローカルとリモートの両方のネットワークマウントされたファイルシステムが含まれる。

/etc/fstab

- ファイルシステム、その標準マウントポイント、およびそれらをマウントする際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。
- ファイルの各レコードは、マウントされるファイルシステムに関する情報を空白で区切られたフィールドで含んでいる。
 - 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
 - カンマ区切りのオプションリスト
 - ダンプ周波数（またはゼロ）
 - **fsck**パス番号（またはゼロ）
- **mount** コマンドと **umount** コマンドは **/etc/fstab** の情報を使用できます。

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る。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、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するか）が続く。続いて、ファイルシステムのタイプ、オプションのカンマ区切りリスト、ダンプ回数（`dump -w`コマンドで使用）またはゼロ（**dumpでは無視**）、**fsck**パス番号またはゼロ（このパーティションを**fsckしない**という意味）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21.7: /etc/fstab の例

21.10 オートマウント

ファイルシステムの自動マウント

- 使用時にファイルシステムを利用可能にし (**mount**)、アイドル時に削除する (**umount**)。
- **autofs**である：
 - 長年にわたって利用できる、
 - **autofs**パッケージのインストールが必要
 - 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**を変更し、**systemd**をリブートまたは再起動する：


```
sudo systemctl daemon-reload
sudo systemctl restart local-fs.target
```

Linuxシステムには昔から、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あった。歴史的に、これは **autofs** を使って行われてきた。このユーティリティを使用するには、適切なパッケージ・マネージャを使用して**autofs**パッケージをインストールし、ファイルを **/etc**

autofsは非常に柔軟でよく理解されているが、**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

```
LABEL=Sam128 /SAM ext4 noauto,x-systemd.automount,x-systemd.device-timeout=10,x-systemd.idle-timeout=30
└─ 0 0
```

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**自動マウント**機能がある。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、マウントオプションを指定する行を追加するだけでよい：

その後、再起動するか、コマンドを発行する：

```
sudo systemctl daemon-reload
```

```
sudo systemctl restart local-fs.target
```

次のページでは、その例を挙げ、オプションについて説明する。

自動マウントの例

```

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21.8: オートマウントの例

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

る。**etc/fstab**のオプション:

- ノオート

ブート時にマウントしない。ここで `auto` は**自動マウント**を意味しない。

- `x-systemd.automount`

systemdの**自動マウント**機能を使う

- `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**でアクセス可能なネットワーク・デバイスの場合、ハングアップする代わりに10秒後にタイムアウトする。

- `x-systemd.automount.idle-timeout=30`

デバイスが30秒間使用されなければ、アンマウントする。

ブート中にデバイスがマウントされることがあるが、指定されたタイムアウト後にアンマウントされることに

注意。スクリーンショットは、デバイスが使用されて初めて利用可能になることを示している。

21.11 ネットワーク・ブロック・デバイス

NBDの紹介

- 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）へブロックデバイスを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を使うことができる。**
- サーバーによってエクスポートされるユニットは、1つまたは複数のファイル、イメージ、ブロックデバイスになります。
- 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できる：
`/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ないが、暗号化は可能であり、NBD仕様の一部である。

追加情報や参考文献は以下の場所にある：**NBD**プロトコル情報:

<https://github.com/NetworkBlockDevice/nbd/blob/master/doc/proto.md>

NBD用ユーザーランド・クライアントおよびサーバー・アプリケーション

: <https://github.com/NetworkBlockDevice/nbd>

イス

ネットワーク・ブロック・デバイス

nbdクライアント/サーバーのペアを設定する一般的な手順は以下の通り：

- サーバーに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分割とフォーマットが可能です。
- パーティション分割された**nbd**デバイスでは、ほとんどのファイルシステムタイプを使用できる。

これから使うステップは以下の通りだ：

1. 空のファイルを作成するには**dd**を使う。
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アクティブにする。
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から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**ブロックドライバをサーバーに関連付ける。
6. **fdisk**を使って**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、マウントする。

NBDユーザーユーティリティ

nbdクライアントとサーバーのパッケージは、以下のようなものがある：

- **nbdkit**: CentOS、Fedora、Debian、Ubuntu
- **nbd-client**と**nbd-server**: Ubuntu、Debian
- **nbd** (GitHubより): CentOS、Fedora、Debian、Ubuntu
- **xNBD-client** と **xNBD-server** です: Debian
- **qemu-img**: CentOS



一般的に、クライアントとサーバーは混ぜて使うことができるので、ユースケースを慎重にテストすることをお勧めする。

今回のラボ環境では、GitHubの**nbd**パッケージを使用する。パッケージの中では最も複雑なものではないが、ほとんどのLinuxバージョンで動作する。

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：

- **nbd-server.conf**は、サーバーの設定ファイルの例である：
 - 接続をリッスンするIPアドレスとポート
 - ディスクとしてエクスポートするストレージ・デバイス
 - 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に使われる。
- **nbd-clientのマニュアル**ページには、サーバーに接続するためのクライアント関連の情報が含まれています。

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、異なる機能を含んでいるかもしれない。

ネットワーク・ブロック・デバイスの例

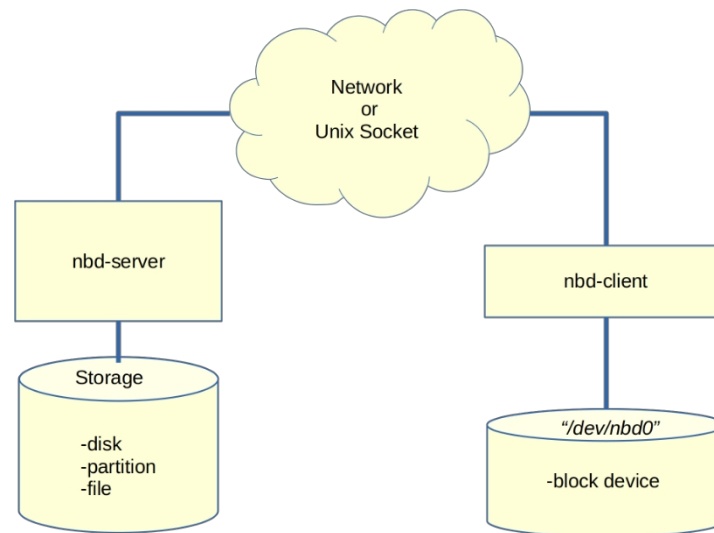


図 21.9: ネットワーク・ブロック・デバイス

クライアントとサーバーのコマンド例。

サーバーは**CentOS-8-Stream**で、**GitHub**の**nbd**パッケージを使用した。

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をローカルデバイス/dev/nbd10に接続する。

```
sudo nbd-client -N foo 192.168.242.160 /dev/nbd10
```

Ubuntuインストールのコマンド例：

- nbdサーバープロセスの開始

```
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
```

イス 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
```

21.12 ラボ



ビデオ・デモンストレーション・リソース

ファイルシステムを使ったデモ.mp4

🔪練習問題 21.1: ファイル属性の操作

1. 通常のユーザーアカウントで、touchを使って/tmp/appenditという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に追加する。
3. tmp/appenditと/etc/hostsの内容を比較し、違いがないことを確認する。
4. chattrを使って、/tmp/appenditにappend-only属性を追加してみてください。ここでエラーが出るはずです。なぜですか？
5. 今度はうまくいくはずだ。今度はうまくいくはずだ。
lsattr.
6. 一般ユーザーとして、catを使って/etc/passwdの内容を/tmp/appenditにコピーしてみる。エラーが出るはずですよ。なぜですか？
7. もう一度同じことをrootでやってみてください。エラーも出るはずです。なぜですか？
8. 通常ユーザーとして、もう一度appendリダイレクト演算子(>>)を使い、/etc/passwdファイルを/tmp/appenditに追加してみる。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、拡張属性をもう一度見てください。
10. tmp/appenditに出力を追加し、ファイル名を変更し、ファイルへのハードリンクを作成し、通常ユーザーとrootの両方でファイルを削除してみてください。
11. 拡張属性を削除すれば、このファイルを削除できる。そうしてください。

✅解答 21.1

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
```

5. `sudo chattr +a appendit`
`lsattr appendit`

-----a-----e-----付記

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
```

終了

8. `cat /etc/passwd >> /tmp/appendit`
`cat appendit`
9. `sudo chattr +i appendit`
`lsattr appendit`

```
----ia-----e- 付記
```

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' に移動できない: 操作は許可されていません。
```

`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: 操作は許可されていません
```

`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' に移動できない: 操作は許可されていません。
```

`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: 操作は許可されていません

終了

11. `sudo su`
`lsattr appendit`

-----ia-----e- 付記

`chattr -ia appendit`
`rm appendit`

rm: 通常のファイル `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がありません

練習21.2: 取り付けオプション

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

この練習では、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ので、両方の方法の詳細を説明します。

1. **fdisk**を使い、システム上に250MBのパーティションを新規作成する（おそらく`/dev/sda`）。あるいは、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、ゼロでいっぱいファイルを作成する。
2. **mkfs**を使って、作成したばかりの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する。これを3回行い、その都度ブロックサイズを変更する。それぞれの場合について、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メモしておくこと。
3. 新しいサブディレクトリを作成し（`/mnt/tempdir`とする）、新しいファイルシステムをこの場所にマウントする。マウントされたことを確認する。
4. 新しいファイルシステムをアンマウントし、読み取り専用として再マウントする。
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。
6. 再度ファイルシステムをアンマウントする。

7. ファイルシステムがブート時にマウントされるように、`/etc/fstab`ファイルに行を追加する。
8. ファイルシステムをマウントする。
9. 新しいファイルシステムの設定を変更し、ファイルシステムからバイナリファイルを実行できないようにする（`/mnt/tempdir`エントリーのデフォルトを`noexec`に変更する）。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。エラーが出るはずだ。

作業が終わったら、`/etc/fstab`からエントリーを削除して、クリーンアップしたいだろう。

✔ 解答 21.2



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なので省略する。作成されたパーティションを `/dev/sda11` を表示するためだ。

```
sudo fdisk /dev/sda
```

```
.....
w
partprobe -s
```

時々、**パートプローブ**が機能しないことがあり、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には、再起動する必要がある。

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) を付けると、要求された情報が表示される。このような小さなパーティションの場合、デフォルトは1024バイトのブロックであることがわかるだろう。

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、現在その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. `etc/fstab`にこの行を入れる:

```
/dev/sda11 /mnt/tempdir ext4 defaults 1 2
```

8.

```
sudo mount /mnt/tempdir
sudo mount | grep tempdir
```

9. `etc/fstab`の行を次のように変更する:

```
/dev/sda11 /mnt/tempdir ext4 noexec 1 2
```

それなら、そうしよう:

```
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 /imagefile`
`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いう警告が出ますが、そのまま進んでください。

vフラグ (verbose) を付けると、要求された情報が表示される。このような小さなパーティションの場合、デフォルトは1024バイトのブロックであることがわかるだろう。

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、現在その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. `etc/fstab`にこの行を入れる:



を `/etc/fstab` に追加する。

`/imagefile /mnt/tempdir ext4 loop 1 2`

8. `sudo mount /mnt/tempdir`
`sudo mount | grep tempdir`
9. `etc/fstab`の行を次のように変更する:



を `/etc/fstab` に追加する。

`/imagefile /mnt/tempdir ext4 loop,noexec 1 2`

それなら、そうしよう:

```
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 21.3: 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**の設定とテストを行います。クライアントとサーバーは同じシステム上で動作します。

NBDエクスポートでは、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かもしれません。



CentOS-stream-8の問題

作成時のCentOS-8-Streamは正常に動作しません。CentOS-

Stream-9またはFedoraをご利用ください。解決待ちです。

1. 最初のステップだ:



Red Hat、Centos、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。クローンし、NBDプログラムをビルドしてインストールする。



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. /optディレクトリにnbd-server.confファイルを作成する。最小限の、コメントなしのファイルで十分である。(より興味深い設定ファイルは SOLUTIONS ディレクトリにあります。
3. エクスポート用に2つのファイルを作成し、所有者を学生に設定する。
4. nbd-serverを起動する。
5. nbdサーバーがクライアントからの問い合わせ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. IPアドレスとポートを使用して、サーバー提供のイメージをローカルブロックデバイスに接続する。
8. NBDの既存の情報をすべて消去する。
9. NBDを使ってGPTを作成する。
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する。

✓ 解答 21.3

1. 最初のステップだ:

**Red Hat、Centos、Fedoraの場合**

```
mkdir ~/src
```

```
cd ~/src
```

```
$ git clone https://github.com/NetworkBlockDevice/nbd.git  
cd nbd/
```

お使いのシステムにはdocbook-utilsとautoconf-archiveパッケージがインストールされていないかもしれません:

```
sudo dnf install docbook-utils  
sudo dnf install autoconf-archive
```

プログラムを構築する:

```
./autogen.sh
```

```
./configure
```

作る

```
sudo make install
```



Debian、Ubuntu、Linux Mintの場合

その Debian または Linux Minthaveのdebパッケージが利用可能な場合は、NBDユーティリティをインストールしてください。

```
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件、新規インストール2件、削除0件、アップグレードなし43件。

\ldots出力切り捨て

2. `opt`ディレクトリに`nbd-server`設定ファイルを作成する。`/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

```
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/*
```

- 4.

```
sudo nbd-server -C /opt/nbd-server.conf
```

- 5.

```
sudo nbd-client -l 127.0.0.1 -p 10042
```

```
交渉: ...
バー
```

6.

```
sudo modprobe -i nbd
```



```
ls /dev/nbd*
```

```
/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
```

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ラベルを作成する:

```
sudo su -c "echo 'label: gpt' | sfdisk /dev/nbd0" "
```

誰もこのディスクを使っていないことを確認。 OK

ディスク /dev/nbd0: 4ギガバイト、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・ヘッダが受理された。

>>> 完了。

新しいGPTディスクラベル (GUID: 000C20C1-6929-424F-93E1-28319DE1FF) を作成しました。

新しい状況だ:

ディスクラベル・タイプ: gpt

ディスク識別子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

スクの同期。

10. NBDにパーティションを追加する:

```
sudo su -c "echo ';' | sfdisk /dev/nbd0"
```

誰もこのディスクを使っていないことを確認。 OK

ディスク /dev/nbd0: 4ギガバイト、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

ディスク識別子000C20C1-6929-424F-93E1-28319DE1FF1F

古い状況だ:

>>> 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 を作成しました。
/dev/nbd0p2: 完了。

新しい状況だ:

ディスクラベル・タイプ: gpt

ディスク識別子 EFFC087B-A15C-7D42-809C-4BDED58EE238

デバイス 開始 終了セクタ サイズ タイプ

/dev/nbd0p1 2048 8388574 8386527 4G 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
クの同期。

11. NBDにext3ファイルシステムを追加してテストする:

```
sudo mkfs.ext3 -L nbd-foo /dev/nbd0
```

```
mke2fs 1.46.5 (2021年12月30日)
```

```
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 ファイルシステムUUID: 6a01ff5a-9ad5-4f43-8050-
```

```
9f2b8df26c14 ブロックに保存されたスーパーブロックバックア
```

```
ップ:
```

```
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
```

```
ジャーナルの作成 (4096ブロック) : 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 の 書き込み: 完了
```

```
sudo mount LABEL=nbd-foo /mnt
```

```
sudo touch /mnt/file1
```

```
ls -l /mnt
```

```
合計16
```

```
0 Jun 9 10:58 file1
```

```
drwxr-xr-x 1 root root 16384 Jun 9 10:56 lost+found
```

第22章

Ext4ファイルシステム



22.1	ext4 機能.....	350
22.2	ext4 のレイアウトとスーパーブロックとブロックグループ	351
22.3	dumpe2fs	354
22.4	tune2fs	355
22.5	研究室.....	356

学習目標

このセッションを終えるまでに、あなたは次のことができるようになるはずだ：

- **ext4** ファイルシステムの主な機能と、それがディスク上にどのように配置されているかを説明する。
- ブロックグループ、スーパーブロック、データブロック、inodeの概念を説明する。
- **dumpe2fs**と**tune2fs**ユーティリティを使う。

349

22.1 ext4 特長

ext4 ファイルシステムの機能

- **ext3**および**ext2**の祖先との下位互換性
- **ブロックリスト**の代わりに**エクステント**を使用
- 持続的な事前割当
- 遅延配分
- 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
- 高速ファイルシステム・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ある。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、断片化を減らすことができる。

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。**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**や**ext2**と下位互換性がある。ファイル用にあらかじめディスク領域を確保することができる。割り当てられた領域は通常、保証された連続領域である。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テクニックも使用します。**ext4**は**ext3**の32,000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するため、信頼性が向上します。また、ジャーナリング中のディスクI/O待ちを安全に回避できるため、パフォーマンスが若干向上します。

ext4はナノ秒単位のタイムスタンプを提供する。

22.2 ext4 レイアウトとスーパーブロックとブロックグループ

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はファイルシステム全体の情報を含む
- 続いて、連続するブロックの集合で構成される**ブロック・グループ**が続く。
 - 管理情報を含む
 - 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される
 - 512バイト、1Kバイト、2Kバイト、4Kバイト、8Kバイトなどがあるが、メモリの1ページ（x86では4KB）より大きくない。

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る。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保持しようとする。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ることになる。

ext4のすべてのフィールドは、ジャーナルを除き、**リトルエンディアン**でディスクに書き込まれる。

Super Block	Group Descriptors	Data Block Bitmap	Inode Bitmap	Inode Table (n blocks)	Data Blocks (n blocks)

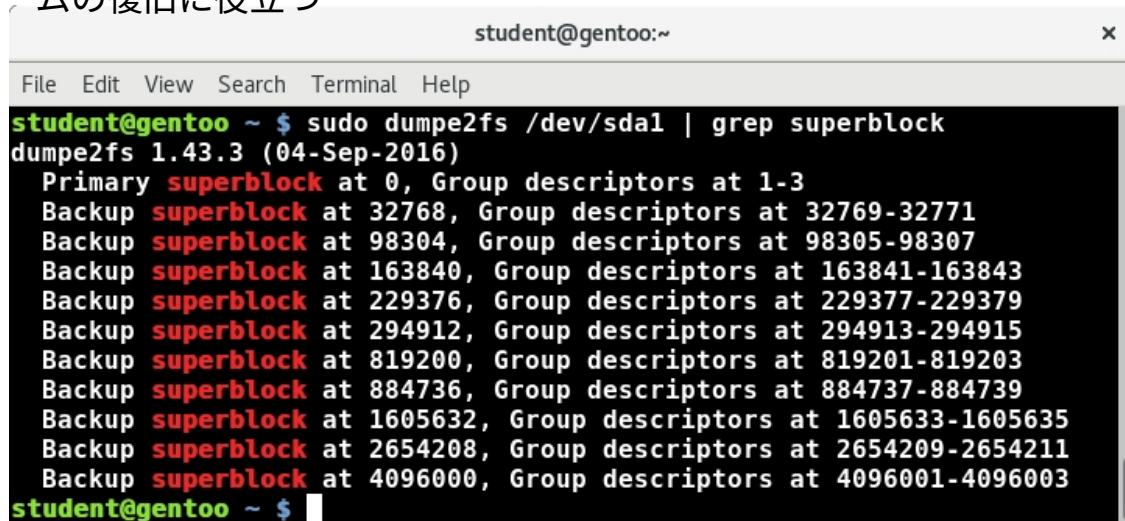
図22.1: ext[3,4] ファイルシステムのレイアウト

標準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の場合、最初の1024バイトは未使用です（ブート・

セクタなどを使用するため)。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、これにグループ記述子といくつかの**GDT**ブロックが続きます。その後にデータブロック・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く。

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がファイルシステムの復旧に役立つ



```
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22.2 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、**スーパーブロック**と**グループ記述子**を構成する。

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、重複コピーはファイルシステムのチェック時に参照されるだけである。問題がなければ、カーネルは最初のブロックグループからコピーするだけである。

マスターコピーに問題がある場合、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、次のコピーに進みます。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る。

extファイルシステム・ファミリーの初期では、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記述子が含まれていました。しかし現在では、最適化のため、すべてのブロックグループがスーパーブロックとグループディスクリプタのコピーを持っている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）調べることができます。これは、ファイルシステムをデフォルトのsparse_superオプション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る。
 - マウント数と最大マウント数
 - このファイルシステムのブロックサイズ
 - ブロック数
 - フリーブロック数
 - 空き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。ファイルシステムのチェックは、最大マウント・カウント毎、または180日毎のどちらか早い方に行われる。

ブロックサイズは**mkfs**コマンドで設定できる。

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に格納される。このスーパーブロックにはファイルシステム自体の情報が含まれている。

22.3 団塊ジュニア

ext4のブロックとInode情報: 団塊ジュ

- 最大数を設定するためのブロックサイズ:
 - 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**を参照のこと:
 - 限界
 - 能力とフラグ
 - その他の属性

dumpe2fsプログラムを使って、特定のパーティションに関する情報を得ることができる。

```

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:
ファイルシステムのUUID: 1ce521c7-e2ce-414a-8a7e-e2311640802f
ファイルシステムマジック番号: 0xEF53
ファイルシステムリビジョ
ン番号: 1 (動的)
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_size
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:
user_xattr acl ファ
イルシステムの状態: クリーン
エラー動作: 続行
ファイルシステム OS タイプ: Linuxxf
ノード数 14352384
ブロック数 57388288
予約ブロック数 2869413
空きブロック数 22270800
空きinode 14352217
最初のブロック 0
ブロックサイズ 4096

ビットマップを1056でブロック (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、8192未使用inode
フリーブロック33822-33985, 34550-34691, 38803-38911
フリー・インノード: 8193-16384
グループ2: (ブロック65536~98303) csum 0xdde9 [INODE_UNINIT, ITABLE_ZEROED]
ビットマップを1057でブロック (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)
0空きブロック、8192空きinode、0ディレクトリ、8192未使用
inode 空きブロック:
空きinode: 16385-24576
....

```


22.4 tune2fs

tune2fs: ファイルシステムのパラメータを変更する

tune2fs は、ファイルシステムのパラメーターを変更するのに**使える**。

- 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：
 へ **変更** **sudo tune2fs -c 25 /dev/sda1** **時間** **間隔** **間** **べんけ**
 (チェック間隔) :
sudo tune2fs -i 10 /dev/sda1 **の** **いじま**
- 変更可能なパラメータの現在値を含むスーパーブロックの内容をリストアップする：
sudo tune2fs -l /dev/sda1

基本的に**dumpe2fs**からのグローバルな情報を示す

```

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:
ファイルシステム: fce521c7-e2ce-414a-8a7e-e2311640802f
ファイルシステムのマジックナン
ズ: 0x1f33 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
リビジョン番号: 1 (ダイナ
ミク)
ファイルシステムの機
能: user_xattr acl ク
能: リーン
ファイルシステ
ムのフラグ: デフォ
ルトのマウントオプ
ション: ファイルシ
ステムの状態エラー
の挙動: ファイルシ
ステム OS タイプ:
Inode count:
ブロック数:
予約ブロック数空き
ブロック数
空きinode: 最
初のブロック
ブロックサイ
ズフラグメント
サイズ
.....
ファイルシステムが
作成されました: 最
終マウント時間最終
書き込み時間マウン
ト回数
最大マウント数最終チ
ェック
チェック間隔生涯書
き込み:
....

```


22.5 ラボ

🔗練習問題 22.1: 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言及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・ファイルシステムは、ファイルシステムの断片化による深刻な問題に悩まされない傾向がある。

これは主に、アクセス時間の速いディスクの一番内側の領域にファイルを詰め込もうとしないためである。その代わり、空き領域をディスク全体に分散させることで、ファイルを作成しなければならないときに、ファイル全体を1つまたは少数の断片に含め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。

最近のハードウェアでは、ディスクの最内周領域という概念は、いずれにせよハードウェアによって曖昧にされている。



こんなことはするな

SSDの場合、デフラグは、読み取り／消去／書き込みサイクルが有限であるため、実際にはストレージメディアの寿命を縮める可能性がある。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になっています。

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**エクステンツ**（連続した大きな領域）で動作します。しかし、**ext4**ファイルシステムの断片化を解除する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、すべての最新 **Linux** ディストリビュー

ションに含まれているはずです。選択肢は2つだけだ：

- `-v`：冗長にする。
- `-c`：実際には何もせず、ただ分析して報告する。

と主張することができる：

- ファイル

- ディレクトリ

- 装置全体

例を挙げよう：

`sudo e4defrag -c /var/log`

<断片化されたファイル	今・最高	サイズ/テキスト
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
エクステンツあたり	120/112	
の平均サイズ	220 KB	

フラグメンテーション
[0-30 問題なし: 31-55 少し断片化: 56- デフラグが必要] このデ
ジョン・スコア
ィレクトリ(/var/log)はデフラグの必要がない。
完了した。

`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]
[4/152]/var/log/messages-20141019.gz: 100%   [OK]
[5/152]/var/log/boot.log:      100%   [OK]
[7/152]/var/log/cups/page_log-20140924.gz: 100%   [OK]
[8/152]/var/log/cups/access_log-20141019.gz: 100%   [OK]
[9/152]/var/log/cups/access_log: 100%   [OK]
[10/152]/var/log/cups/error_log-20141018.gz: 100%   [OK]
[11/152]/var/log/cups/error_log-20141019.gz: 100%   [OK]
[12/152]/var/log/cups/access_log-20141018.gz: 100%   [OK]
[14/152]/var/log/cups/page_log-20141018.gz: 100%   [OK]
...
[152/152]/var/log/Xorg.1.log.old:      100%   [OK]

      成功      [ 112/152 ]
      敗        [ 40/152 ]
```

様々なファイル、ディレクトリ、デバイス全体で**e4defrag**を実行してみてください。

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のは、90%以上の容量がいっぱいになったときか、ブートパーティションを使用しているときのように、容量が小さく、比較的大きなファイルがあるときだけです。

練習問題 22.2: 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プロパティをいじってみる。最初にファイルシステムをアンマウントする必要はない。

以下のように、作成した画像ファイルを扱うことができる:

```
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に置き換えることもできる。

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に関する情報を取得する

。表示する:

- ・ マウント回数の最大値 (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る)。
- ・ 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)
- ・ 予約されたブロック数、および総ブロック数

2. 変化する:

- 最大マウント数を30に。
- チェック間隔は3週間に。
- ブロックの予約率を10%に。

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、元の出力と比較する。

 解答 22.2

1. `dumpe2fs imagefile > dumpe2fs-output-initial`

```
dumpe2fs 1.42.9 (2013-12-28)
```

```
grep -i -e "マウント数" -e "チェック間隔" -e "ブロック数" dumpe2fs-output-initial
```

```
ブロック数:          262144
予約ブロック数:      13107
マウント数:          0
最大マウント数:      -1
チェックの間隔      0 (<なし>)
```

2. `tune2fs -c 30 imagefile`

```
tune2fs 1.42.9 (2013-12-28) 最大マ
ウント数を 30 に設定
```

```
tune2fs -i 3w imagefile
```

```
tune2fs 1.42.9 (2013年12月28日)
チェック間隔を1814400秒に設定
```

```
tune2fs -m 10 imagefile
```

```
tune2fs 1.42.9 (2013年12月28日)
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 (2013-12-28)
```

```
grep -i -e "マウント数" -e "チェック間隔" -e "ブロック数" dumpe2fs-output-final
```

```
ブロック数:          262144
予約ブロック数:      26214
マウント数:          0
最大マウント数:      30
チェックの間隔      1814400 (3週間)
```

```
diff dumpe2fs-output-initial dumpe2fs-output-final
```

```
14c14
< 予約ブロック数          13107
---
> 予約ブロック数29c29      26214
< 最終書き込み時間        2016年10月26日 (水)
---
> 最終書き込み時間        2016年10月26日 (水)
31c31
< 最大マウント数          14:26:20
---
```



```
> 最大マウント数          30
33c33,34
< チェック間隔            0 (<なし>)
---
> チェックの間隔          1814400 (3週間)
> 次のチェックはその      2016年11月16日 (水)
    後だ:                  13:26:16
```


第23章

論理ボリューム管理（LVM）



23.1	論理ボリューム管理（LVM）	362
23.2	ボリュームとボリュームグループ	363
23.3	論理ボリュームの操作	364
23.4	論理ボリュームのサイズ変更	367
23.5	LVM スナップショット	368
23.6	研究室	369

学習目標

このセッションを終えるまでに、あなたは次のことができるようになるはずだ：

- **LVM**の背後にある概念を説明する。
- 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する。
- 論理ボリュームを表示します。

- 論理ボリュームのサイズを変更する。

ご注意

** これらのセクションは、一部または全部をオプションとすることができる。**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。講師は、授業の経験や時間的制約に応じて、これらの項目を取り上げるか取り上げないかを選択することができます。

361

23.1 論理ボリューム管理 (LVM)

LVM

- 複数のディスクをグループ化してボリュームを作成
- RAIDデバイスと同様の機能
 - RAIDデバイスの上に構築可能
- RAIDよりも優れたスケーラビリティ
 - 簡単にサイズ変更可能（上でも下でも）
 - デバイスを追加して成長させることができる

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。

LVMを使うことには多くの利点がある。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たり、ストレージ容量を増やしたり、配置を変えたりするのがとても簡単になる。

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。そして、ボリュームグループは、名目上の物理ディスクパーティションを模倣する**論理ボリューム**に細分化され、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る。

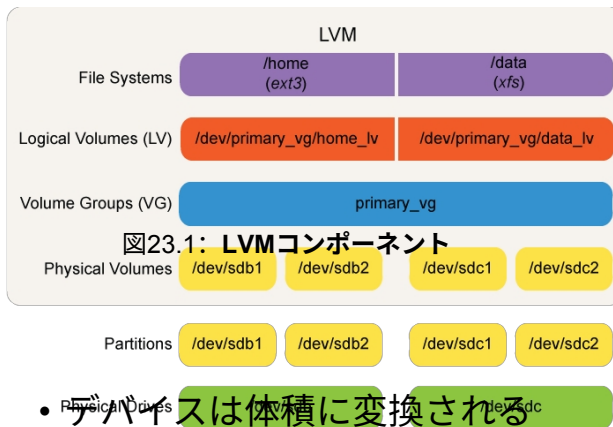
物理ボリュームや論理ボリュームの作成、削除、サイズ変更などを行うコマンドライン・ユーティリティはいろいろある。グラフィカルなツールとしては、ほとんどのLinuxディストリビューションが**system-config-lvm**を使用している。しかし、**RHEL**はもはやこのツールをサポートしておらず、ファイルシステム・タイプなどの最新のバリエーションで確実に動作するグラフィカル・ツールは今のところ存在しない。幸いなことに、コマンドラインユーティリティは使いにくくなく、とにかく柔軟だ。

LVMはパフォーマンスに影響を与えます。**LVM**レイヤーのオーバーヘッドから生じる追加コストが確実に存在する。しかし、**非RAID**システムであっても、**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、並列化をある程度改善することができます。

論理ボリュームは**RAID**デバイスに似た機能を持つ。実際に**RAID**デバイスの上に構築することもできる。これにより論理ボリュームは、**LVM**の拡張性と**RAID**デバイスの冗長性を併せ持つことになる。

LVMは**RAID**よりもスケーラビリティに優れており、論理ボリュームは必要に応じて簡単にサイズを変更することができる。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。

23.2 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

• マルチプル 物理ボリューム

ボリュームグループにグループ化

される

– 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論理ボリューム上に構築される

– 名前は何でもいい

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る。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。これら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**で始まる。

を含む：

- **vgcreate**：ボリュームグループを作成します。
- **vgextend**：ボリュームグループに追加します。
- **vgreduce**：ボリュームグループを縮小します。

どの物理パーティションがボリューム・グループに入るか、またはボリューム・グループから出るかを操作するユーティリティは、**pv**で始まり、以下を含む：

- **pvcreate**：パーティションを物理ボリュームに変換します。
- **pvdisk**：使用中の物理ボリュームを表示します。
- **pvmove**：何らかの理由でディスクやパーティションを削除する場合に必要となる。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かもしれない。
- **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る。

23.3 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23.2 論理ボリュームユーティリティ 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ある。当然のことながら、それらはすべて **lv** で始まる。ここでは最もよく使われるものについて説明するが、簡単なリストは以下の方法で入手できる:

```
ls -lF /sbin/lv*
```

これらのユーティリティは **/usr/sbin** ではなく **/sbin** にある。ブート時や修復・復旧時に必要になる可能性があるからだ。

これらのほとんどは **lvm** にシンボリックリンクされている。lvm はスイスアーミーナイフのようなプログラムであり、すべての作業をこなすが、呼び出された名前から何をするように要求されているかがわかる。これは **pv*** と **vg*** ユーティリティのほとんどにも当てはまる。

```
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23.3 物理ボリュームユーティリティ物理ボリューム・ユーティリテ
イ

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とタイプする) .
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する。
6. 論理ボリュームをマウントする (必要に応じて/etc/fstabも更新する) 。

23.3. 論理ボリュームの操作

365

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。サイズはバイト数かエクステント数で指定できる (デフォルトでは4MBであることを覚えておこう)。名前は任意である。

lvdisplay は利用可能な論理ボリュームを表示します。

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る。

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手順は以下のようになる:

```
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 mylv vg
sudo mkfs -t ext4 /dev/vg/mylv
mkdir /mylv
sudo mount /dev/vg/mylv /mylv
```

必ず次の行を追加してください。

```
/dev/vg/mylv /mylv ext4 defaults 1 2
```

を/etc/fstabに追加して、これを永続マウントにする。

論理ボリュームの表示

- **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一覧表示されます。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。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23.4 ボリュームグループユーティリティボリューム・グループ・ユーティリティ

23.4 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

- 物理パーティションのサイズ変更よりはるかに簡単
- エクステントはボリュームグループ内のどこからでも取得できます。
- オンライン状態でもファイルシステムの縮小や拡張が可能
- エクステント数またはバイト単位（M、Gなど）でサイズを変更できる。
- のように**lvresize**を使うのがベストだ：

```
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・システムのサイズ変更も行う。

- **resize2fs**で**lvextend**や**lvreduce**を使うこともできる。
- リサイズ・ユーティリティはファイルシステム固有

ext4ファイルシステムで論理ボリュームを拡張する：

```
sudo lvresize -r -L +100M /dev/vg/mylvm
```

ここで、プラス記号はスペースの追加を示す。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ことに注意。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減らすこともできる：

```
sudo pvmove /dev/sdc1  
sudo vgreduce vg /dev/sdc1
```

23.5 LVMスナップショット

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
- スナップショットの元の状態はブロックマップ
- デルタの保存にのみスペースを使用
 - 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がスナップショットにコピーされる
 - スナップショットにデータが追加された場合、そこにのみ保存される

LVMスナップショットは、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る。

スナップショットはデルタを保存するためのスペースしか使わない：

- 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が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ると、そこだけに保存される。

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、以下の手順に従います：

```
sudo lvcreate -l 128 -s -n mysnap /dev/vg/mylvm
```

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：

```
mkdir /mysnap
```

```
マウント -o ro /dev/vg/mysnap /マイスナップ
```

スナップショットを使用し、スナップショットを削除する：


```
sudo umount /mysnap
```

```
sudo lvremove /dev/vg/mysnap
```

スナップショットを使い終わったら、必ずスナップショットを削除してください。スナップショットを削除せず、変更によっていっぱいになった場合、スナップショットは自動的に無効になります。オリジナルのサイズのスナップショットがオーバーフローすることはありません。

23.6 ラボ



ビデオ・デモンストレーション・リソース

LVMデモ.mp4

練習問題 23.1: 論理ボリューム

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、そこに2つの物理ボリュームを追加します。デフォルトのエクステン・サイズを使用します。
4. ボリュームグループ myvg から mylvm という名前の 300 MB 論理ボリュームを割り当てます。
5. 論理ボリュームmylvmをフォーマットし、`/mylvm`にマウントする。
6. 論理ボリュームに関する情報を表示するには、`lvs`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に拡張します。

✓ 解答 23.1

1. 実行する:

```
sudo fdisk /dev/sda
```

適当なハードディスクを使って、2つのパーティションを作成する。`fdisk`で`t`と入力すると、パーティションタイプを8eに設定できる。タイプを設定しなくても問題はないが、混乱を減らすには良いアイデアだ。`w`でパーティションテーブルを書き換えて終了し、次に

```
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ったことを確認します。

2. 新しいパーティションを`/dev/sdaX`と`/dev/sdaY`とする:

```
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、この行を追加する：



を `/etc/fstab` に追加する。

```
/dev/myvg/mylvm /mylvm ext4 defaults 0 0
```

6. `sudo lvm`

7. `df -h`
`sudo lvresize -r -L 350M /dev/myvg/mylvm`
`df -h`

または

`sudo lvresize -r -L +50M /dev/myvg/mylvm`

あるいは、もっと古い方法でもできる:

`df -h`
`sudo lvextend -L 350M /dev/myvg/mylvm`
`sudo resize2fs /dev/myvg/mylvm`
`df -h`

第24章

カーネルサービスと構成



24.1	カーネルの概要	372
24.2	カーネルブートパラメータ	373
24.3	カーネルコマンドライン	374
24.4	ブートプロセスの失敗	375
24.5	sysctl	376
24.6	研究室	377

学習目標

このセッションを終えるまでに、あなたは次のことができるようになるはずだ：

- 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どのようなパラメータを設定できるのか、また、それらを1回のシステムブート時だけ、あるいは永続的に有効にする方法について説明する。
- ブートプロセスの失敗のトラブルシューティング。
- これらのパラメータに関する詳細な文書がどこにあるか知っている。
- **sysctl**を使用して、システム起動後にカーネル・パラメーターを設定するか、システム・リブートにまたがって永続的に設定する。

24.1 カーネルの概要

カーネルの概要

- カーネルはオペレーティング・システムの中核である。
- カーネルの責任には以下が含まれる：
 - システムの初期化と起動
 - プロセス・スケジューリング
 - メモリー管理
 - ハードウェアへのアクセス制御
 - 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）とネットワーク上の両方のセキュリティ管理
 - 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る。

狭義には、Linuxは**オペレーティング・システムのカーネル**に過ぎず、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る。

カーネルは、ハードウェアをソフトウェアに接続し、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のシステムリソースを管理する重要な中心的コンポーネントである。**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。

カーネルのみを実行するシステムの機能はかなり限られている。

24.2 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`のカーネル行に表示される。
 - ブート時にインタラクティブに供給することもできる
 - パラメータは次のように指定できる：
 - 引数としての単純な値
 - `param=value`の形式で、`value`には文字列、整数、整数の配列などを指定する。
- ```
vmlinuz root=/dev/sda6 rhgb quiet crashkernel=512M
```
- 意図的な秘密、隠しパラメータはない
  - カーネルソースの `kernel-parameters.txt` または <https://kernel.org/doc/html/latest/admin-guide/kernel-parameters.html> に列挙されている。

```
linux /boot/vmlinuz-5.19.0 root=/dev/sda5 ro crashkernel=512M quiet selinux=0
```

カーネル・オプションはカーネル行の最後に置かれ、スペースで区切られる。カーネルブートパラメータの例：

- `root`: ルート・ファイルシステム (`root=UUID=...`、`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読み取り専用マウントする
- `crashkernel=512M`: `kdump`機能によるカーネル・クラッシュダンプ用に確保するメモリ量。
- `quiet`: ほとんどのログメッセージを表示しない
- `selinux=0` **SELinux** を無効にする

オンマシンのカーネルドキュメントの `man bootparam` と `kernel-parameters.txt` を見てください。カーネルドキュメントはカーネルソースの中にあるか、`kernel-doc` のような名前のパッケージの中にあります。さらに簡単には、<https://kernel.org/doc/html/latest/admin-guide/kernel-parameters.html>をオンラインで見てください。

Linuxディストリビューションは、そのディストリビューション特有のパラメーターを追加することが多いことに注意してください。例えば、上の例の `rhgb` は、**Red Hat Graphical Boot** の略です。

## 24.3 カーネルコマンドライン

# カーネルコマンドライン

- ブート時にカーネルの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の行に置く:  
grub.cfgを/boot以下のどこかに置く。
- ブート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る:

```
linux /boot/vmlinuz-4.15.0-58-generic ㉞
 root=UUID=5cec328b-bcd6-46d2-bcfa-8430257cd7a5
 ㉞ ro find_preseed=/preseed.cfg auto noprompt
 ㉞ root=UUID=5cec328b-bcd6-46d2-bcfa-
 8430257cd7a5
 priority=critical locale=ja_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
 | find_preseed=/preseed.cfg auto noprompt
 priority=critical locale=ja_US quiet crashkernel=512M-:192M
```

カーネルのコマンドラインのサンプルはディストリビューションによって異なり、以下のようになる:

```
linux /boot/vmlinuz-5.19.0 root=UUID=7ef4e747-afae-48e3-90b4-9be8be8d0258 ro quiet
linuxefi /boot/vmlinuz-5.2.9 root=77461ee7-c34a-4c5f-b0bc-29f4fecc743 ro crashkernel=auto rhgb
└─静かなクラッシュカーネル=384M-:128M
```

grub.cfgは、/boot/grubや/boot/efi/EFI/centos/grub.cfgのように、ブート下のサブディレクトリにある。

指定されたvmlinuzファイル以降はすべてオプションである。カーネルが理解できないオプションは、init (pid = 1) システム上で最初に実行されたユーザー・プロセス。システムがどのコマンドラインで起動されたかを見るには、次のようにタイプする:

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）を使用しており、カーネルコマンドラインが設定される場所を変更します。その情報は、[/boot/grub2/grubenv](#)か、[/boot/loader/entries](#)のファイルのいずれかを見てください。

## 24.4 ブートプロセスの失敗

# ブートプロセスの失敗

- ブートプロセスに従う
  - バイオス
  - GRUB
  - カーネル
  - 開始

システムが正しく起動しなかったり、完全に起動しなかったりした場合、各段階で何が起こるかをよく理解しておくことが重要です。**BIOS**の段階を通過したと仮定すると、次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がない：**

**GRUB**の設定ミスまたはブートセクタの破損を確認してください。ブートローダーの再インストールが必要ですか？

**カーネルのロードに失敗：**

ブートプロセス中にカーネルが**パニックになる場合は**、カーネルの設定ミスか破損、または**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。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ている可能性があります。どちらかによって、カーネルを再インストールするか、ブート時に対話型**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って修正を試みます。あるいは、レスキューイメージで起動してみることもできます。

**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：**

主な原因は以下の通りである。

1. **GRUB** 設定ファイルの設定ミス。
2. `etc/fstab` の設定ミス。
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ていない。

**init プロセス中の失敗。**

**init**が始まるとうまくいかないことがたくさんあります。停止する前に表示されるメッセージをよく見てください。停止する前に表示されるメッセージをよく見てください。以前は別のカーネルで動いていたのなら、それは大きな手がかりです。ファイルシステムの破損、スタートアップスクリプトのエラーなどに注意してください。3（グラフィックなし）や1（シングルユーザーモード）などの低いランレベルで起動してみてください。

## 24.5 sysctl

# sysctl

- `proc/sys`の設定は**sysctl**で操作できる。
- 変数名は`/proc/sys`のパスに対応する。
- カーネルパラメータの表示、設定、変更が可能

```
[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24.1: sysctl

**sysctl**インターフェースは、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できる。現在の値は

```
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'
sudo sysctl net.ipv4.ip_forward=1
```

(最初の形式では、単純な**sudo**と**echo**を使うことはできない。

設定が`/etc/sysctl.conf`に置かれていれば、ブート時に設定を修正することができる。と入力することに注意：

```
sudo sysctl -p
```

これはブートプロセスの一部でもある。



**/usr/lib/sysctl.d**および**/usr/lib/sysctl.d**

**systemd**の出現で、物事は少し複雑になった。ベンダーは設定を`/usr/lib/sysctl.d`ディレクトリに置く。これらは`/etc/sysctl.d`に置かれたファイルに追加したり、置き換えたりすることができます。しかし、自分で

ドキュメント化したオリジナルのファイル (`/etc/sysctl.conf`) はまだサポートされています。





## 24.6 ラボ



### ビデオ・デモンストレーション・リソース

[sysctl\\_demo.mp4](#)を使用する。

### 練習問題 24.1: sysctlによるシステム調整

1. 自分のシステムにpingが打てるかどうか確認してください。
2. `net.ipv4.icmp_echo_ignore_all`の現在の値をチェックする。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る。値が0の場合、システムはpingに応答する。
3. **sysctl**コマンドラインユーティリティを使用して値を1に設定し、pingが応答するかどうかを確認する。
4. 値を0に戻し、元の動作を復元して表示する。
5. ここで、[/etc/sysctl.conf](#)を修正して値を変更し、システムを強制的に再起動せずにこの設定ファイルを有効にする。
6. これが正しく機能したことを確認する。

作業が終わったら、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だろう。

### 解答 24.1

以下のpingのターゲットには、localhost、127.0.0.1（ループバックアドレス）、または実際のIPアドレスのいずれかを使用できます。

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`に以下の行を追加する：



を `/etc/sysctl.conf` に追加する。

```
net.ipv4.icmp_echo_ignore_all=1
```

そして、そうする：

```
sysctl -p
```

6. 

```
sysctl net.ipv4.icmp_echo_ignore_all
```

```
$ ping localhost
```

`etc/sysctl.conf`へ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だろう。

## ✎練習問題24.2： 最大プロセスIDの変更

Linuxシステムの通常の動作は、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。

しかし、PIDが`/proc/sys/kernel/pid_max`に表示されている値（従来は32768（32K））に達すると、それ以下の数に折り返される。PIDのロットはその数しかないので、何もなければ、これはシステム上に32K以上のプロセスを持ってないことを意味する。

1. 現在のPID最大値を取得する。
2. 現在発行されているPIDを調べる
3. `pid_max`を現在発行されている値よりも低い値にリセットする。
4. 新しいプロセスを開始し、PIDを確認する。

## 🟢 解答 24.2



### 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echoする方法だ。**echo**の方法はrootである必要があることに注意してほしい。`sudo`では動作しない。その理由は、まだわかっていなければ、わかりだろう！

1. `sysctl kernel.pid_max`  
`cat /proc/sys/kernel/pid_max`

2. タイプ

猫

```
[1] 29222
```

```
$ kill -9 29222
```

3. `sudo sysctl kernel.pid_max=24000`  
`$ echo 24000 > /proc/sys/kernel/pid_max # これはroot権限で行う必要があります。`  
`cat /proc/sys/kernel/pid_max`

4. 猫

```
[2] 311
```

```
$ kill -9 311
```

**ご注意**

やり直す場合、カーネルはPID=300から開始し、それ以下の値では開始しないことに注意してください。システムがすでにターンオーバーしている可能性があるため、カーネルは新しいPIDを生成するときに、そのPIDがすでに使用されていないかどうかを常にチェックしなければならない。**Linux**カーネルには、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がある。

## 第25章

# カーネルモジュール



|      |                    |     |
|------|--------------------|-----|
| 25.1 | カーネルモジュール.....     | 380 |
| 25.2 | モジュール・ユーティリティ..... | 382 |
| 25.3 | モデインフォ.....        | 384 |
| 25.4 | モジュール構成.....       | 385 |
| 25.5 | 研究室.....           | 386 |

### 学習目標

このセッションを終えるまでに、あなたは次のことができるようになるはずだ：

- カーネルモジュールを利用する利点を挙げてください。
- カーネルモジュールのロードとアンロードには、**insmod**、**rmmod**、**modprobe**を使う。
- カーネルモジュールに関する情報を調べるには**modinfo**を使う。
- モジュールの設定方法を学ぶ。

379



## 25.1 カーネルモジュール

# カーネルモジュール

- カーネルの拡張
- 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。
- 多くの場合、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。
- `lib/modules/$(uname-r)`にある。
- 特定のカーネル・バージョン用にコンパイル
- カーネルは**モノリシック**なままであり、**マイクロカーネル**ではない

**Linux**カーネルは**モジュール**を多用している。モジュールには重要なソフトウェアが含まれており、システム起動後に必要に応じてロードしたりアンロードしたりできる。

多くのモジュールは、システム内部または周辺に接続されたハードウェアを制御するための**デバイスドライバ**を内蔵している。他のモジュールは、ネットワークプロトコルを制御したり、異なるファイルシステムタイプをサポートしたり、他の多くの目的を果たすことができる。

モジュールのロード時にパラメータを指定して、その動作を制御することができる。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。



ル

## lsmodで使用されているモジュールのリスト

```

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25.1: lsmodの使用

ほとんどのカーネル機能はモジュールとして設定することができる。この柔軟性はまた、開発やデバッグ中にテストするためにシステムを再起動する必要がほとんどないため、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のような方法は使われているが、**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る。

## 25.2 モジュール・ユーティリティ

# モジュール・ユーティリティ

- **エルエスマッド**

ロードされたモジュールのリスト

- **インスマッド**

モジュールを直接ロードする

- **rmmod**

モジュールを直接取り外す

- **モッドプローブ**

依存関係や位置情報を含む事前に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。

- **デブモッド**

モジュール依存データベースを再構築する

- **モデインフォ**

モジュールに関する情報を表示する

モジュールをロードするには**modprobe**を使う：

```
modprobe e1000e
```

モジュールのアンロードや削除には**modprobe -r**を使う：

```
modprobe -r e1000e
```

**modprobe**はモジュール依存性データベースの更新を要求します。  
`/lib/modules/$(uname -r)/modules.dep`.

モジュールを直接ロードするには**insmod**を使う（完全修飾モジュール名が必要）：

```
$ 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**を使う：

```
rmmod e1000e
```

ロードされたモジュールを一覧表示するには**lsmod**を使う：

```
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**を使う：

```
modinfo e1000e
```



リティ

## モジュールに関する考察

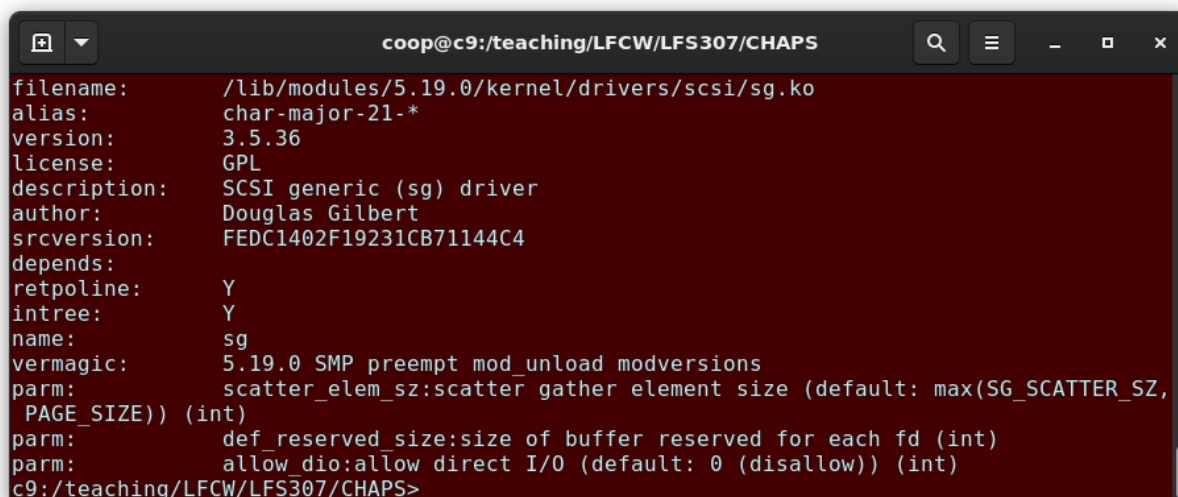
- モジュールをアンロードすることは不可能だ：
  - 別のモジュールで使用中
  - 実行中のプロセスで使用中
- モジュールをロードすることができない：
  - 異なるカーネル、または異なるオプションでコンパイルされたもの
  - 現在実行中のモジュールと競合する
- モジュールは通常、**insmod**ではなく**modprobe**でロードされる。
- 容認できないオープンソースライセンスでロードされたモジュールは、カーネルを**汚染されたもの**としてマークする。

モジュールの積み下ろしには、いくつか注意すべき点がある：

- 他の**モジュールによって**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。  
を掲載した。
- 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ある。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る。ネットワーク・スタック全体をシャットダウンして再起動することなく、モジュールを一時的に置き換えるのはあまりにも難しくなるからだ。
- モジュールが**modprobe**でロードされると、最初にロードする必要がある他のモジュールはシステムが自動的にロードします。
- モジュールが**modprobe -r**でアンロードされると、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ければ、システムは自動的にアンロードします。

## 25.3 モデインフォ

### 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25.2: 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）を調べるのに使うことができ、バージョン、ファイル名、デバイス・ドライバ・モジュールがどのハードウェア・デバイスを扱えるか、ロード時にどのようなパラメータを与えることができるかといった情報を表示する。

モジュールに関する多くの情報は、[/sys](#)擬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せよ、いくつかのパラメータは[/sys/module/sg/パラメータ](#)の下で読み書きできます。次にその設定方法を説明する。

のように、パラメータ値を指定しながらロードできるモジュールも多い：

```
$ sudo /sbin/insmod <pathto>/e1000e.ko debug=2 copybreak=256
```

あるいは、すでに適切なシステム位置にあるモジュールの場合は、もっと簡単だ：

```
sudo /sbin/modprobe e1000e debug=2 copybreak=256
```

## 25.4 モジュール構成

### /etc/modprobe.d/

- `etc/modprobe.d`サブディレクトリ・ツリー内の、`/etc/modprobe.d`の末尾が`.conf`拡張子は、**modprobe**を使用してモジュールがロードおよびアンロードされるときにスキャンされます。
- 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**化できる
- 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またはアンロード時に実行するコマンドを指定する。

`etc/modprobe.d`ディレクトリのファイルは、**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**に入れてロードされないようにすることもできます。

設定は、モジュールがロードまたはアンロードされるときに適用され、ニーズの変化に応じて設定を変更することができます。

`etc/modprobe.d`のファイル形式はシンプルで、1行に1コマンド、空白行と`#`で始まる行は無視される（コメントを追加するのに便利）。行末にバックスラッシュを付けると、次の行に続くので、ファイルが少しすっきりする。

## 25.5 ラボ



### ビデオ・デモンストレーション・リソース

カーネルモジュールを使う.mp4

#### 練習 25.1: カーネルモジュール

1. システムに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する。
2. 現在ロードされていないモジュールをシステムにロードする。  
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)` を見るだけです。  
`/kernel/drivers/net` ディレクトリにあるドライバを1つ取得する。(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あらゆるデバイス、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る)。しかし、カスタムカーネルを実行している場合は、アンロードされたモジュールがあまり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からです。
3. ロードされているすべてのカーネルモジュールを再リストし、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する。
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ているかどうかを確認してください。

#### ✓ 解答 25.1

1. `lsmod`
2. 以下では、使用したモジュール名を `e1000e` に置き換えてください。これらの方法はどちらも使えるが、もちろん2番目の方法の方が簡単だ。

```
sudo insmod /lib/modules/$(uname -r)/kernel/drivers/net/ethernet/intel/e1000e.ko
sudo /sbin/modprobe e1000e
```

3. `lsmod | grep e1000e`
4. もう一度言うが、どちらの方法も有効だ。

```
sudo rmmod e1000e
sudo modprobe -r e1000e
```

5. `lsmod | grep e1000e`



## 第26章

# デバイスと udev



|      |                   |     |
|------|-------------------|-----|
| 26.1 | udev とデバイス管理..... | 388 |
| 26.2 | デバイス・ノード .....    | 389 |
| 26.3 | 規則 .....          | 392 |
| 26.4 | 研究室.....          | 395 |

### 学習目標

このセッションを終えるまでに、あなたは次のことができるようになるはずだ:

- デバイスノードの役割と、メジャー番号とマイナー番号をどのように使用するかを説明する。
- udevメソッドの必要性を理解し、その主要コンポーネントを列挙する。
- udev デバイスマネージャの機能について説明する。
- udev ルールファイルを特定し、カスタムルールを作成する方法を学ぶ。

387



## 26.1 udevとデバイス管理

# ウデヴ

- udevはUser Device Managementの略です。
- **udevは**、内蔵ハードウェアと周辺デバイスを動的に検出する：
  - システム起動中
  - ホットプラグ接続時
- **udevは**、必要に応じて適切な設定を行い、デバイスドライバのロードとアンロードを処理する。
- **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** がどのように動作するかを制御し、特別な**udevのルール**。

udevのアクションには以下が含まれる：

- デバイス名
- デバイスファイルとシンボリックリンクの作成
- ファイル属性の設定
- 必要な行動をとる

デバイスがシステムに追加または削除されると、**udev**はカーネルからメッセージを受け取る。そして、以下のルールを解析する。

ファイルを/etc/udev/rules.dディレクトリに追加し、追加または削除されたデバイスのルールがあるかどうかを確認する。

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、作成されるシンボリックリンクの指定、デバイス・ファイルに設定されるファイル属性の指定（ユーザーとグループの所有権を含む）

、さらに実行されるアクション（実行されるプログラム）の指定まで可能です。

## 26.2 デバイス・ノード

# デバイス・ノード

- デバイス・ノードは/devディレクトリにあります。
- キャラクタ・デバイスとブロック・デバイスにはデバイス・ノードがあるが、ネットワーク・デバイスにはない。
- プログラムは、通常のI/Oメソッドを使用して、これらのノードを介してデバイスと通信します。
- 1つのデバイス・ドライバが複数のデバイス・ノードを使用することがある
- デバイスノードは

```
$ sudo mknod [-m mode] /dev/name <タイプ> <メジャー> <マイナー>
```

例えば

```
sudo mknod -m 666 /dev/mycdrom c 254 1
```

ls -l /dev

```
合計 0
クルー -----1 コーブオー 14, 4 7月9日 01:54 オーディオ
ディオ
crw -----1 root ルー 10, 62 7月9日 01:54 autofs
ト
lrwxrwxrwx 1 root root 4 7月9日01:54 cdrom -> scd0
lrwxrwxrwx 1 root root 4 7月9日 01:54 cdrw -> scd0
crw -----1 coop root 5, 1 7月9日 06:54 コンソール

lrwxrwxrwx 1 root root 4 7月9日01:54 dvd -> scd0
lrwxrwxrwx 1 root root 4 7月9日01:54 dvdwriter -> scd0

lrwxrwxrwx 1 root root 8, 0 7月9日01:53 SDA
lrwxrwxrwx 1 root root 8, 1 7月9日 01:53 sda1

brw-r-----1 root ディ 8, 2 7月9日 06:54 sda2
スク
brw-r-----1 root ディ 8, 16 7月 9 01:53 sdb
スク 8, 17 7月 9 01:53 sdb1
brw-r-----1 root ディ 8, 18 7月 9 01:53 sdb2
スク
crw-rw-rw- 1 root tty 5, 0 7月 9 01:54 tty
brw-rw-----1 root ネット 4, 0 7月 9 14:54 tty0
スク -----1 root ルート 4, 1 7月 9 06:54 tty1
brw-r-----1 root ディ 1, 7 月 9 01:53 ウランダ
スク -----1 root root 日 ム
brw-r-----1 root ディ 1, 5 7月 9 01:54 ゼロ
スク-rw-rw- 1 root root

```

# デバイス・ノードがハードウェア・アクセスを提供

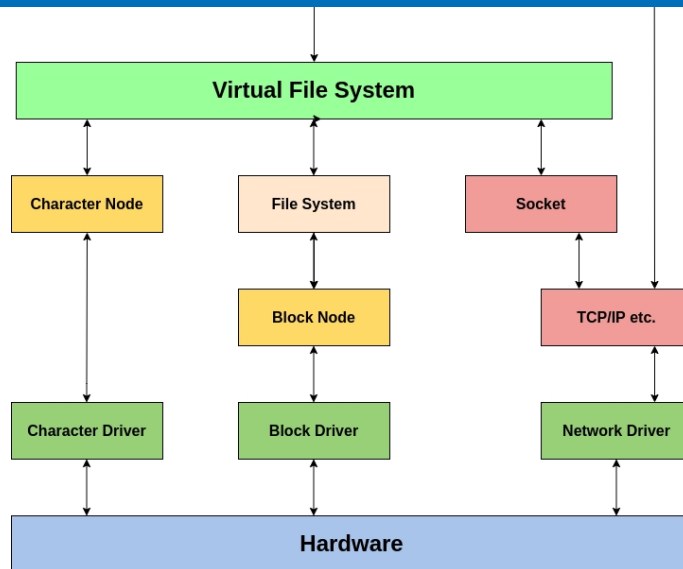


図26.1: デバイス・ノード

`dev/null`や`dev/zero`など、必要な（udev以外で静的に作成された）擬似デバイスを提供することに加え、`/dev`に存在するデバイスノードは、マシン上のハードウェアとの主要な、あるいは最初の相互作用となる。

一般的な装置には次のようなものがある：

- `/dev/sd*`-ハードドライブとして表示されるデバイス
- `/dev/ttyS*`-シリアルポート、コンソールとしてよく使われる
- `/dev/snd/*` - 各種サウンドデバイス

## udev コンポーネント

- **libudev** ライブラリ

アクセスを許可する アクセス  
へのアクセスを許可しま  
す。 デバイスに関する情報

- **udev**または**systemd-udev**  
dae- mon

devディレクトリの管理

- **ウデバドム**

制御・診断用ユーティリティ

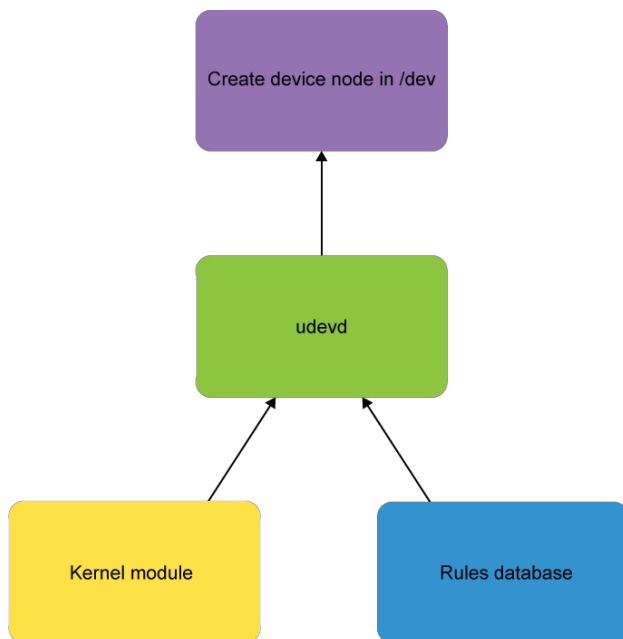


図26.2: udevのルール

**udev**はデーモン（**udev**または**systemd-udev**）として動作し、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** カーネル機能がソケットを通じてメッセージを送信し、**udev** がそれを受信して、ルールに従って適切な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3つのコンポーネントは以下の通り：

1. デバイスに関する情報にアクセスできる**libudev**ライブラリ。
2. devディレクトリを管理する**udev**デーモン
3. 制御と診断のための **udevadm** ユーティリティ。

**udev**を使用する最もクリーンな方法は、純粋なシステムを持つことです。devディレクトリは、最初のカーネルブート時には空で、必要に応じてデバイスノードを追加します。この方法で使用する場合は、**initramfs**イメージを使ってブートする必要があります。**initramfs**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が含まれ





## 26.3 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
```

- 例を挙げよう:

```
30-usb.ルール
90-mycustom.rules
70-mouse.rules
60-persistent-storage.規則
```

1行に2つのパートが定義されている:

- 最初の部分は、`==` で示される1つまたは複数のマッチ・ペアから構成される。これらは、デバイスの属性および/または特性と、ある値とのマッチングを試みる。
- 2番目の部分は、ファイル名、グループ割り当て、ファイル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ています。

一致するルールが見つからなければ、デフォルトのデバイス・ノード名とその他の属性を使用する。

```
$ 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/XBoxCreateUSBNode.sh $major $minor
↳ $attr{bDeviceClass}"
SUBSYSTEM=="usb", ACTION=="add", ENV{DEVTYPE}=="usb_device", RUN+="/usr/lib/virtualbox/VBoxCreateUSBNode.sh $major
↳ $minor $attr{bDeviceClass}"
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"
```

## udevルールの作成

- ルールのフォーマット

```
<match><op>value [, ...] <assignment><op>value [, ...]。
```

- サンプル

```
KERNEL=="sdb", DRIVER=="usb-disk", SYMLINK+="sparedisk"
KERNEL=="sdb", NAME="my-spare-disk"
```

```
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デバイス:**

```
$ cat /etc/udev/rules.d/99-fitbit.rules
```

```
SUBSYSTEM=="usb", ATTR{idVendor}=="2687", ATTR{idProduct}=="fb01",
↳ SYMLINK+="フィットビット", MODE="0666"
```

- **kdump/kexecでクラッシュダンプを作成し、高速にカーネルをロードする**

```
:
```

```
$ 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=="メモリ", ACTION=="オンライン", PROGRAM="/bin/systemctl try-restart
↳ kdump.service"
```

```
SUBSYSTEM=="メモリ", ACTION=="オフライン", PROGRAM="/bin/systemctl try-restart
↳ kdump.service"
```

```
KERNEL=="kvm", GROUP="kvm", MODE="0666"
```

行することで、これらのルールがリアルタイムでどのように適用されるかを監視する:

- **kvm仮想マシン・ハイパーバイザーの場合:**

```
sudo udevadm monitor
```

```
$ cat /usr/lib/udev/rules.d/80-kvm.rules
```



## ビデオ・デモンストレーション・リソース

udev\_demo.mp4を使用しています。



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

/メディア

5. ls -lF /dev | grep myusb



- 27.1 IPアドレス
- 27.2 IPv4アドレスの種類
- 27.3 IPv6アドレスの種類
- 27.4 IPアドレスクラス
- 27.5 ネットマスク
- 27.6 ホスト名
- 27.7 エヌティーピー
- 27.8 ラボ

## 学習目標

このセッションを終えるまでに、あなたは次のことができるようになるはずだ:

- IPv4アドレスとIPv6アドレスの種類を区別する。
- IPアドレスクラスを理解する
- ネットマスクの役割を理解する。
- システムホスト名の取得、設定、変更。
- NTPを使用して、時刻の設定、同期、修正を行う。

397





# 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る。
- IPアドレスはRIRを通じて割り当てられる
- **IPv4** 32ビット・アドレスは4オクテットから構成される。
- 予約済みアドレスがある
- ここでは、現在も最も一般的に使用されている**IPv4**に焦点を当てる。

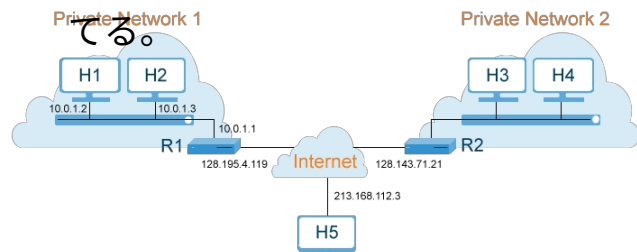


図27.1: IPアドレス

2003:0db5:6123:0000:1f4f:0000:5529:fe23  
148:114:252:10

(オクテットは単なるバイト、8  
ビット)。

- **IPv6** 128ビット・アドレスは、8つの16ビット・オクテット・ペアで構成される。

## 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

-

-

-

-

## 予約アドレス

- ループバックインターフェース:

127.x.x.x

- 住所は未定:

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 - 最上位オクテットレンジ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、ループバック用に127.x.x.xを予約
- 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 - 最上位オクテットレンジ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
  - 16進数でff:00:00:00:00
  - 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
  - 16進数でff:ff:00:00:00
  - 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 ネットマスク

-----

172.16.0.0 ネットワークアドレス



## ホスト名の取得と設定

- ネットワークデバイスを他のノードと区別するためのラベル
- 歴史的には**ノードネーム**とも呼ばれる
- インストール時に設定するが、後で変更可能
- どのユーザーでもホスト名を取得できる：

ホスト名

wally

- ホスト名を変更するにはroot権限が必要です：

`sudo hostname lumpy`

lumpy

- 現在の値は、ほとんどのLinuxでは常に/etc/hostnameに保存されています。
  -
- を分配する。
- リブートしても変更を**持続**させるには、**systemd**インフラストラクチャの一部である**hostnamectl**を使う。

`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 シ

チャーシ: 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

'\_(ウートパ)

dracut-049-70.git20200228.el8 (Initramfs) カ

ーネル: Linux 5.11.6

アーキテクチャ: x86-64

と使用法のメッセージを見る:

hostnamectl --help

hostnamectl [OPTIONS...] COMMAND ...シス

テムのホスト名を照会または変更する。

-h --ヘルプ 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

--transient ホスト名を要求しない、一時的なホスト名

--static ホスト名を要求しない、永続的なホスト名

-H --host=[USER@]HOST 現在のホスト名を表示する

-M --machine=CONTAINER 現在のホスト名を操作する

--status システムのホスト名

--set-hostname NAME を設定する ホストの

set-icon-name NAME アイコン名を設定す

set-chassis NAME set-

deployment NAME る

ホスト ホストのシャーシタイプ

set-locale LANG を設定する 配備環境を設

set-locale LANG を設定する

set-locale LANG を設定する

set-locale LANG を設定する

set-locale LANG を設定する

set-locale LANG を設定する

set-locale LANG を設定する

set-locale LANG を設定する

set-locale LANG を設定する

set-locale LANG を設定する

set-locale LANG を設定する

set-locale LANG を設定する

set-locale LANG を設定する

set-locale LANG を設定する

set-locale LANG を設定する

set-locale LANG を設定する

set-locale LANG を設定する

set-locale LANG を設定する

set-locale LANG を設定する

set-locale LANG を設定する

set-locale LANG を設定する

set-locale LANG を設定する

set-locale LANG を設定する

set-locale LANG を設定する

set-locale LANG を設定する

set-locale LANG を設定する

set-locale LANG を設定する

set-locale LANG を設定する

set-locale LANG を設定する

set-locale LANG を設定する

set-locale LANG を設定する

set-locale LANG を設定する

set-locale LANG を設定する

set-locale LANG を設定する

set-locale LANG を設定する

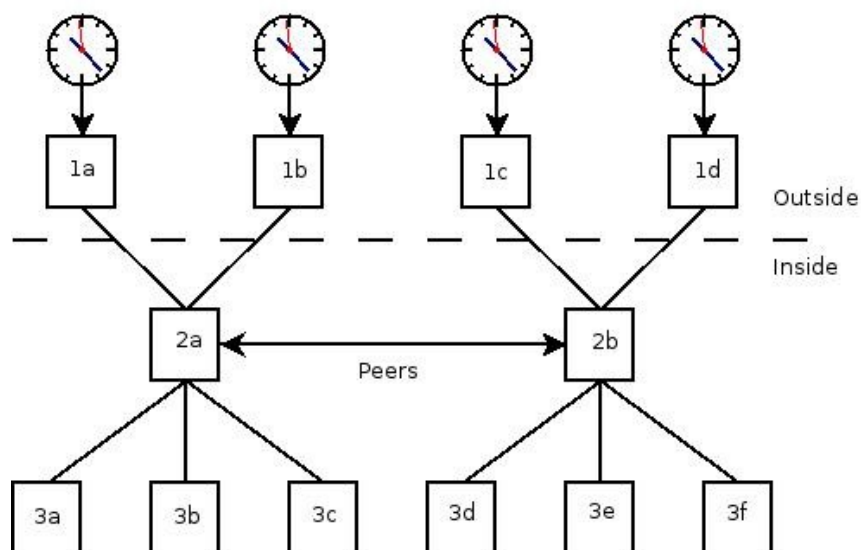
set-locale LANG を設定する

set-locale LANG を設定する

set-locale LANG を設定する

set-locale LANG を設定する

# 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27.2: 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るが、最も一般的なものをいくつか紹介しよう：

- **ntp**、<http://www.ntp.org>、デフォルト。
- **chrony**、<http://chrony.tuxfamily.org>、断続的なネットワーク接続や仮想マシンを含む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クライアントである。

- [/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する。
- NTPデーモンの起動
- 時刻が同期していることを確認する

クアドレス

<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 pool``0.pool.ntp.org``pool 1.pool.ntp.org``pool 2.pool.ntp.org``pool 3.pool.ntp.org``# timedatectl`

現地時間日 2023-03-05 16:13:50 UTC

世界標準時日 2023-03-05 16:13:50 UTC

RTC時間日 2023-03-05 16:13:50

タイムゾーンEtc/UTC (UTC,

+0000) システムクロック同期: 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**のコンフィギュレーション要素は

[/etc/ntp.conf](#)ファイル。以下は、**ntpサーバー**に関連する項目の一部である：

- ローカルマシンを時間参照と宣言する
  - サーバーとごまかしのディレクティブ
- **ntpq** および **ntpd** コマンドで、誰がタイムサーバに問い合わせることが出来るかを制御する (CVE-2013-5211 を参照)
  - 制限指令
- どのシステムが**ntpピア**であるかの宣言
  - ピアは[/etc/ntp.conf](#)ファイルにリストされている。
- NTPデーモンの起動

`/etc/ntp.conf`

### `/etc/ntp.conf`: アクセス制御

```
デフォルトのポリシーはクエリを防ぐ
restrict default nopeer nomodify notrap noquery
特定のサブネットからの問い合わせを許可する。
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 #
```

ローカルホストの制限解除

**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`

```
サーバー 127.127.1.0
ファッジ 127.127.1.0 層数10
```





1.

```
yum install ntp
zypper install ntp
apt install ntp
```

2.

/etc/ntp.conf



/etc/ntp.conf

```
サ ー バ ー
0.pool.ntp.org サ ー
バ ー 1.pool.ntp.org
サ ー バ ー
2.pool.ntp.org サ ー
バ ー 3.pool.ntp.org
```

3.

```
systemctl restart ntpd
systemctl restart ntp
```

4.

```
ntpq -p
```



ご注意

出力を得る前に、タイムサーバーが同期するまで数分待たなければならないかもしれない。



- 28.1 ネットワーク機器
- 28.2 予測可能なネットワーク・インターフェース・デバイス名
- 28.3 ネットワーク設定ファイル
- 28.4 ネットワーク・マネージャー
- 28.5 ルーティング
- 28.6 仮想ネットワーク・インターフェイス
- 28.7 DNSと名前解決
- 28.8 ネットワークのトラブルシューティング
- 28.9 ネットワーク診断
- 28.10 ラボ

**学習目標**

このセッションを終えるまでに、あなたは次のことができるようになるはずだ:

- 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、特定の職務に割り当てるかを理解する。
- **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。
- **PNIDN (Predictable Network Interface Device Names)** スキームを理解する。
- **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。
- 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するには、**Network Manager** (**nmtui**と**nmcli**) を使用します。
- デフォルトルートやその他のスタティックルートを設定する方法を知っている。
- 仮想ネットワークインターフェースを理解する
- DNSと名前解決を理解する
- ネットワークのトラブルシューティングを行い、貴重な診断ユーティリティを実行します。



## ネットワーク機器

- 名前は通常、タイプ識別子の後に数字が続く：
  - イーサネット：eth0、eth1、eno1、eno2など。
  - ワイヤレス：wlan0、wlan1、wlp3s0、wlp3s2など
  - ブリッジ接続：BR0、BR1など
  - 仮想：vmnet1、vmnet8など。
- 1つのデバイスが複数のIPアドレスを持つことができる
  - 古いシステムでは、eth0:4 → eth0 のような関連付けがよくあった。
  - **ifconfig**の代わりに**ip**を使用する新しいシステムでは、1つのインターフェース名に複数の広告ドレスが割り当てられます。

/dev

## 予測可能なネットワーク・インターフェース名

- **PNIDN (Predictable Network Interface Device Names)** は、udevとsystemdとの統合に接続されている。
- 現在、デバイスに付けられる名前は5種類ある：
  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む。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みます。例: ens1
  3. ハードウェア接続の物理的・地理的位置を含む。例: enp2s0
  4. MACアドレスを組み込む。例: enx7837d1ea46da
  5. 古い古典的な方法を使う。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
```

```
lspci | grep Ethernet
```

```
02:00.0 イーサネット・コントローラーイーサネットコントローラ: Marvell Technology Group Ltd.88E8056 PCI-E
ギガビット イーサネット コントローラ (rev 12) 04:02.0 イーサネット コントローラ: イーサネットコントロー
ラ: 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)
```



## ネットワーク設定ファイル

- 各ディストリビューションは、独自のファイルやディレクトリのセットを持っています：
  - 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**の標準化が進む中
  - 多くの場合、**Network Manager**を使用するのが望ましい。
  - 新しいディストリビューションでは、これらのファイルはなくなっているか、空になっている可能性があります。

## ネットワークマネージャーの設定ファイル

- **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**プラグインによって**サポートされています。
- **Key-Value**が望ましいファイル形式です。
- 従来のコンフィギュレーション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ある。
  - `etc/network/interfaces` の **ifupdown**
  - `etc/sysconfig/network-scripts` 用の **ifcfg-rh**
  - **ifcfg-suse**はSUSEとopenSUSEのシンプルな互換性です。
  - **key-file**は、システム固有のコンフィギュレーション・ファイルの一般的な置き換えである。

詳細は <https://wiki.gnome.org/Projects/NetworkManager/SystemSettings/> を参照。

`/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、ハードウェアでもソフトウェアでも、以前はほとんどが静的だった。
- 現代のシステムは、しばしば動的なコンフィギュレーションを持つ：
  - デバイスが場所を移動すると、ネットワークが変更される可能性がある。
  - 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。
  - ワイヤレス・デバイスなどのハードウェアが接続されたり、オン・オフされたりすると、デバイスが変更されることがある。
- これまでに説明した設定ファイルは、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する。
- **ネットワーク・マネージャ**は、異なるシステムでもほとんど同じであるべきです。

/etc

# ネットワーク・マネージャー・インターフェース

- **GUI**

すべてのLinuxデスクトップ（GNOME、KDE、XFCEなど）には、通常、水平パネル内の何かをクリックすることで到達できる。

- **ンツイ**

ネットワーク・マネージャー・テキスト・ユーザー・インターフェースは、**ncurses**ライブラリー・インターフェースを使用した、ナビゲート可能なシンプルなコマンドライン・インターフェースです。

- **エヌエムクリ**

ネットワークマネージャー・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ある。スクリプト

ター

# ンツイ



図28.1: nmtuiメイン画面

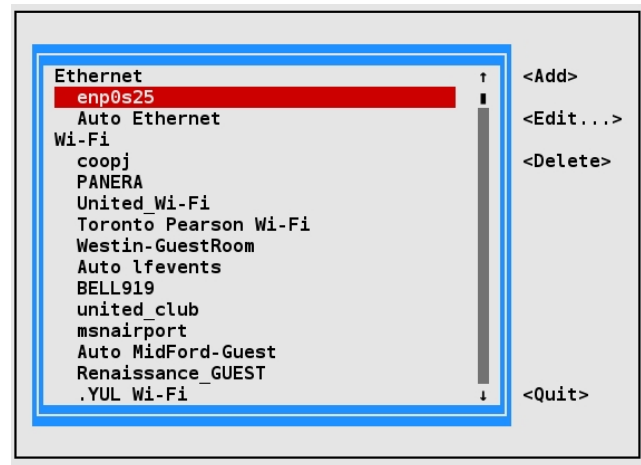


図28.2: nmtui編集画面

# nmtuiワイヤレス設定

**Edit Connection**

Profile name   
Device

**WI-FI** <Hide>

SSID   
Mode

Security   
Password   
[ ] Show password

BSSID   
Cloned MAC address   
MTU

= IPv4 CONFIGURATION  <Show>  
= IPv6 CONFIGURATION  <Show>

[X] Automatically connect  
[X] Available to all users

<Cancel> **<OK>**

図28.3: nmtuiワイヤレス設定

マナー

# エヌエムクリ

- **Network Manager**へ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
- nmcli-examples も参照してください。



- 練習でいくつかの例を試してみる

<https://fedoraproject.org/wiki/Networking/CLI> を参照。

## nmcli --help

使用法: nmcli [OPTIONS] OBJECT { COMMAND | help

.OPTIONS

|                                  |                              |
|----------------------------------|------------------------------|
| -a, --ask                        | 足りないパラメータを尋ねる                |
| -c, --colors auto yes no         | 出力に色を使うかどうか                  |
| -e, --escape yes no              | 値の列区切り文字をエスケープする。            |
| -f, --fields <フィールド,...> すべて 共通  | 出力するフィールドを指定する               |
| -g, --get-values <field,...> all | 一般的なショートカット -m tabular -t -f |
| -h, --help                       | ヘルプを表示する                     |
| -m, --mode tabular multiline     | 出力モード                        |
| -o, --オーバービュー                    | 概要モード                        |
| -p, --pretty                     | きれいな出力                       |
| -s, --show-secrets               | パスワードの表示を許可する                |
| -t, --terse                      | 簡潔な出力                        |
| -v, --バージョン                      | プログラムのバージョンを表示               |
| -w, --wait <秒>                   | 操作の終了を待つタイムアウトを設定する。         |

オブジェクト

|                |                             |
|----------------|-----------------------------|
| 一般             | NetworkManager の一般的なステータスと操 |
| 作 n[etworking] | ネットワーク全体の制御                 |
| r[adio]        | NetworkManagerの無線スイッチ       |

c[onnection] NetworkManagerの接続 d[evice]  
NetworkManagerが管理するデバイ  
ス  
エージェント NetworkManagerのシークレットエージェ  
ントまたはpolkitエージェントm[onitor] NetworkManagerの  
変更を監視する



ヤー

## ルーティング

- パケットのネットワーク間移動
- 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
  - 直接送信されるローカル・パケット
  - 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28.4: routeとip routeの使用



グ

## デフォルトルート

- ルーティングテーブルエントリーにマッチしない場合に使用される
- DHCPを使用して動的に取得可能
- 手動で設定可能（静的）
  - **Red Hat** 由来のシステム: GATEWAY=x.x.x.xのいずれかを指定してください。  
`/etc/sysconfig/network` または  
`/etc/sysconfig/network-scripts/ifcfg-ethX`
  - **Debian** 由来のシステム: gateway=x.x.x.x in  
`/etc/network/interfaces`
- **nmcli**を使えば、`/etc`のファイルを手で修正する必要がなくなる:
 

```
$ sudo nmcli con mod virbr0 ipv4.routes 192.168.10.0/24 \
+IPv4.ゲートウェイ 192.168.122.0
sudo nmcli con up virbr0
```

```
sudo ip route add default via 192.168.1.10 dev enp2s0
ip route
```

```
デフォルト 192.168.1.10 dev enp2s0 経由
デフォルト via 192.168.1.1 dev enp2s0 metric 1024
192.168.1.0/24 dev enp2s0 proto kernel scope link src 192.168.1.100
```

```
sudo ip route add default via 192.168.1.1 dev enp2s0
```

## 静的ルート

- 複数のルーターやルートが存在する場合に、パケットの流れを制御するために使用される
- インターフェースごとに定義
- 持続性も非持続性もある
- ipの使用（非永続的）
- あるいはしつこく：
  - Red Hatベースのシステム：  
`/etc/sysconfig/network-scripts/route-ethX`
  - Debianベースのシステム：  
`/etc/network/interfaces`
  - SUSEベースのシステム：  
`/etc/sysconfig/network/ifroute-eth0`
- 手で/etcのファイルを変更するのを避けるためにnmcliを使う。

```
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 ip route add 10.1.2.51 dev eth1
post-up ip route add 10.1.2.52 dev 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
```

グ  
10.1.1.0/24 192.168.1.1 - eth0



## ネットワーク・インターフェースのチーム化とボンディング

- ・ チーミング・インターフェースとボンディング・インターフェースは、動的なネットワーク・インターフェース構成を可能にし、より高いスループットとさらなる堅牢性をネットワークに提供します。

### – 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をアグリゲーションし、フェイルオーバー、冗長性、パフォーマンスの向上を実現します。

### – チーム編成

ボンディングと似ているが、チーミングはteamedというユーザー空間のサービスを使う。



#### チーム編成は非推奨とされている！

- ・ **Teaming**コンフィギュレーションメソッドは**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**へのスムーズな変換のために、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る。

# ボンディング・ネットワーク・インターフェイス

ボンディング・ネットワーク・インターフェイスの設定にはいくつかの方法がある。

- シスフ

`sys` 疑似ファイルシステムに直接変更を加える。変更は保存されない。

- アイブルート2

`ip` コマンドによるボンディングをサポートする。変更は保存されない。

- ネットワークマネージャー

**NetworkManager** のすべてのユーザーインターフェイスはボンディングをサポートしています。変更は保存されます。

**NetworkManager** の設定ファイルは、お好みのテキストエディタで編集することもできます。



## もっと知りたい？

ボンディングの `sys` インターフェイスの詳細については、

<https://www.kernel.org/doc/Documentation/networking/bonding.txt> を参照のこと。



## nmcliによるボンディングネットワークインターフェースの作成

nmcliでボンディングインターフェースのコンフィグレーションを作成するには、いくつかのステップが必要です。最小限の手順は次のとおりです：

- **アダプタを識別します：** nmcli device status を使用する。
- **ボンディングデバイスを作成する：** nmcli connection addを使用する。
- **インタフェースをボンドにアタッチします：** 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・アダプタをオンラインに設定する：** ボンド・アダプタに対して nmcli connection up コマンドを発行する。
- **再起動すること：** 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、コンフィギュレーションの断片が一掃される。

\$ \_

## コマンドライン

```
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 e
ifname enp0s20u1u1 e
master bond0
nmcli connection add type ethernet slave-type bond
\ con-name bond0-port2 \
ifname enp0s20u1u2 \
master bond0
nmcli connection up bond0
nmcli device status reboot
nmcli デバイス ステータス
```



## もっと知りたい?

nmcliについては<https://wiki.gnome.org/Projects/NetworkManager/SystemSettings/>、ボンディングとボンディング・オプションについては<https://www.kernel.org/doc/Documentation/networking/bonding.txt>。

## /etc/hosts

- ローカルまたは静的ホスト名解決に使用
- 孤立した小規模ネットワークに有効
- 通常、DNSの前にチェックされる

/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

/etc /etc/hosts.deny /etc/hosts.allow

/etc/host.conf

## DNSと名前解決

- 名前解決：ホスト名をIPアドレスに変換する
  - 静的名前解決
  - 動的名前解決（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
```

[/etc/hosts](#)

-  
-  
-

## ディグの使用

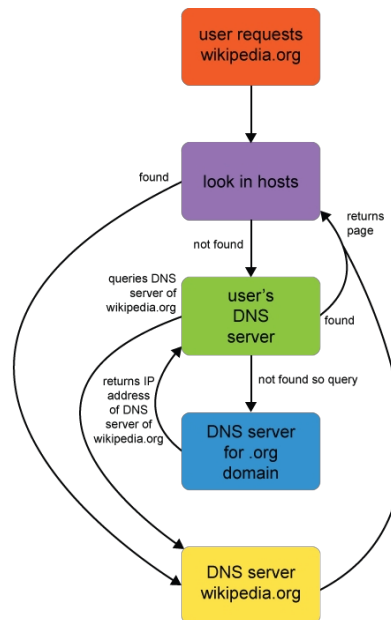
- 出荷されたツールの中で最も正しい答えを出す
- デフォルトの出力は、単純なタスクやスクリプティングには冗長すぎるが多い。
- 解像度チェーン全体を見ることができる
- DNSSECの動作を表示および検証できる
- 便利なオプションもある：
  - +ルートサーバー（デフォルト）からパス全体をチェックする。
  - t[レコードタイプ] レコードを指定します (A、TXT、CNAME、MX を含む)。

\$ dig linuxfoundation.org

```
<<>> DiG 9.16.27-Debian <<>> linuxfoundation.org
グローバルオプション: +cmd
答えが出た:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, ID: 54765
フラグ: qr rd ra; QUERY: 1, ANSWER: 1, 権限: 0, 追加: 1

linuxfoundation.org. イン A
EDNS: version: 0, flags:;, udp: 4096
回答欄
質問コーナー: 524 イン A 3.13.31.214
linuxfoundation.org
ウェリー時間3ミリ秒
サーバ: 10.139.1.1#53(10.139.1.1)
いつ: 2023年5月7日 (日) 13:40:00 EDT
MSG SIZE rcvd: 64
```

\$ dig -x 1.1.1.1 +short



ワン・ワン・ワン・ワン

# DNS

- 動的ネームサーバーを使用
- `/etc/resolv.conf`
  - 特定のDo-mainを指定して検索することができる。
  - クエリーす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、

図28.5: DNS

**DHCP** (Dynamic Host Configuration Protocol) などのサービスから更新することもできる。

/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

これは、**NetworkManager**がプライマリ・ネットワーク・インターフェイス上で**DHCP**を起動することによって生成されたものである。



## 28.8 ネットワークのトラブルシューティング

# ネットワーク問題のトラブルシューティング

- IPコンフィギュレーション
- 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

ネットワークの問題は、ソフトウェアまたはハードウェアのどちらかが原因で発生することがあり、デバイスドライバがロードされているか、ネットワークケーブルが接続されているかといった単純なものから発生することがあります。ネットワークが稼動しているにもかかわらずパフォーマンスが悪い場合は、トラブルシューティングではなく、パフォーマンスチューニングの対象となります。問題はマシンの外部にあるかもしれないし、バッファサイズなどさまざまなネットワーキング・パラメーターの調整が必要かもしれない。

ネットワークに問題がある場合、以下の項目をチェックする必要がある：

### • IPコンフィギュレーション：

ifconfigがipを使って、インターフェイスがアップしているかどうか、またアップしている場合はコンフィグレーションされているかどうかを確認する。

### • ネットワークドライバ：

インターフェイスが立ち上がらない場合、ネットワークカード用の正しいデバイスドライバがロードされていない可能性がある。ネットワーク・ドライバがカーネル・モジュールとしてロードされているかどうかをlsmodで確認するか、[/proc/interrupts](#)や[/sys/class/net](#)など、[/proc](#)や[/sys](#)にある関連する擬似ファイルを調べてください。

### • 接続性：

tracerouteはネットワークを介してパケットを追跡し、mtrはこれを連続的に行うことができます。これらのユーティリティを使うことで、問題がローカルにあるのかインターネット上にあるのかを知ることができる。

### • デフォルトゲートウェイとルーティングの設定：

route -nを実行し、ルーティングテーブルが意味を持つかどうかを確認する。

- ホスト名の解決:

URLに対して**dig**または**host**を実行し、**DNS**が正しく機能しているかどうかを確認する。

## 28.9 ネットワーク診断

# ネットワーク診断

- ping
- traceroute
- mtr

基本的なネットワーク・ユーティリティの数々は、すべてのシステム管理者のツールボックスに入っている：

**ping:**

64バイトのテストパケットを指定されたネットワークホストに送り、到達に必要な時間（ミリ秒単位）、失われたパケット、その他のパラメータを報告しようとする。正確な出力は、対象となるホストによって異なるが、少なくともネットワークが動作しており、ホストに到達可能であることがわかる。

**traceroute:**

宛先までのネットワーク経路を表示するために使われる。パケットがホストに到達するまでに通過するルーターと、**各ホップにかかる時間を表示する。**

**mtr:**

pingとtracerouteの機能を組み合わせ、**top**のような継続的に更新される表示を作成する。

RHELなど一部のLinuxディストリビューションでは、これらのユーティリティを実行するために（**sudo**のような）root権限が必要な場合があることに注意。

## 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28.6: ping

### ピング

往復の所要時間を調べる。..

**-c 10**

合計**10**パケット。

**linuxfoundation.org**

に、設定されたリゾルバから返されたIPアドレス（複数ある場合はランダム）を渡す。

**linuxfoundation.org** ...

最後に、結果をまとめる（デフォルトの動作）

## tracerouteの例

```
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28.7: トレースルート

### トレースルート

ネットワークパスを検索する（デフォルトではUDPポートを使用）。..

### グーグル

を、設定されたリゾルバが**google.com**に返したIPアドレス（複数ある場合はランダム）に変換する。



#### 隠された複雑性

**google.com**のホスト名検索は、さまざまなDNSとインターネットルーティング技術を使用して、ネットワークの場所に依存するIPアドレスの異なるセットを返します。

## 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
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28.8: mtr

### mtr

ネットワーク経路を検索する（UDPポートまたはICMPを使用）。...

### t

GUI ではなくターミナルを使って。...

### カーネル

を、設定されたリゾルバが**kernel.org**に返したIPアドレス(複数ある場合はランダム)に変換する。

## 28.10 ラボ

### 練習28.1: 静的ホスト名の追加

この練習では、ローカル・ホスト・データベースにエントリーを追加する。

1. `etc/hosts`を開き、ネットワークカードに関連付けられたIPアドレスを指す`mssystem.mydomain`のエントリーを追加する。
2. `ad.doubleclick.net`へのすべての参照を`127.0.0.1`を指すようにする2番目のエントリを追加します。
3. オプションの練習として、<http://winhelp2002.mvps.org/hosts2.htm> または <http://winhelp2002.mvps.org/hosts.txt> から直接ホストファイルをダウンロードし、システムにインストールしてください。新しいホストファイルがある場合とない場合で、ブラウザの使い方に違いはありますか？

### 解答 28.1

1. 

```
$ sudo sh -c "echo 192.168.1.180 mssystem.mydomain >> /etc/hosts"
$ ping mssystem.mydomain
```
2. 

```
$ sudo sh -c "echo 127.0.0.1 ad.doubleclick.net >> /etc/hosts"
$ ping ad.doubleclick.net
```
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... 接続されました。
HTTP リクエストが送信されました、応答待ちです... 200 OK
長さ: 514744 (503K) [text/plain] 保存先:
100%[=====>>] 514,744 977KB/s 0.5秒
hosts.txt
2014-11-01 08:57:13 (977 KB/s) - hosts.txtが保存された
[514744/514744]。
```

```
sudo sh -c "cat hosts.txt >> /etc/hosts"
```

### 練習問題 28.2: digでDNS情報を検証する

この演習では、IPv4 Aレコードのフォワードが、**dig**を持つメールサーバーのリバースPTRレコードと一致することを検証する。

1. バインドがインストールされていることを確認してください。なければ



Red Hat、Centos、Fedoraの場合

V.1.4

bind-utilsをインストールする

2. **protonmail.com**のメールサーバーIPアドレスを見つける。



3. IPアドレスの逆引きが一致することを確認する。

## ✓ 解答 28.2

1. `$ dig +short -t mx protonmail.com`

```
10 mailsec.protonmail.ch.
5 mail.protonmail.ch.
```

`$ dig +short -t A mail.protonmail.ch`

```
185.70.42.128
185.205.70.128
176.119.200.128
```

2. `$ dig +short -x 185.205.70.128`

```
mail.protonmail.ch.
```

## ✍ 演習28.3: nmcliを使用したネットワークインターフェイスエイリアス/アドレスの追加

システムにIPv4アドレスを追加し、それを永続化する。これは、`/dev`配下のファイルを直接編集することなく、`nmcli`を使って行う。

1. まず、現在のインターネット・アドレスとインターフェース名を取得する：

`sudo nmcli con`

```
名前 ユー タイ デバイ
自動イーサネット 1c46bf37-2e4c-460d-8b20-421540f7d0e2 802-3-ethernet ens33
virbr0 a84a332f-38e3-445a-a377-4363a8eb963f bridge virbr0
ディ
ー
```

は接続名がオートイーサネットであることを示す。

`sudo nmcli con show "Auto Ethernet" | grep IP4.ADDRESS`

```
ip4.address[1]: 172.16.2.135/24
```

は、アドレスが172.16.2.135であることを示す。このコマンドは接続に関するすべての情報を表示することに注意：

`nmcli con show 1c46bf37-2e4c-460d-8b20-421540f7d0e2`

2. あなたのマシンが見える新しいアドレスを追加する：

---

```
sudo nmcli con modify "Auto Ethernet" +ipv4.addresses 172.16.2.140/24
```

3. アクティベートして、そこにあるかどうかテストする：

```
sudo nmcli con up "自動イーサネット"
```

接続は正常にアクティブ化された (Dバス・アクティブ・パス：  
↳ /org/freedesktop/NetworkManager/ActiveConnection/6)

```
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。
172.16.2.140からの64バイト: icmp_seq=1 ttl=64 time=0.038 ms
172.16.2.140からの64バイト: icmp_seq=2 ttl=64 time=0.034 ms
172.16.2.140からの64バイト: icmp_seq=3 ttl=64 time=0.032 ms

--- 172.16.2.140のping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 = 0.032/0.034/0.038/0.007 ms
```

4. エイリアスを削除してクリーンアップする:

```
sudo nmcli con modify "Auto Ethernet" -ipv4.addresses 172.16.2.140/24
...
sudo nmcli con up "自動イーサネット"
...
```

## ✎ 演習28.4: nmcliを使用した静的ルートの追加

静的IPv4ルートアドレスをシステムに追加し、それを永続化する。これは、`/dev`配下のファイルを直接編集せずに、`nmcli`を使って行う。

1. `ip route`を使って現在のルーティングテーブルを調べることから始める:

```
ip route
```

```
デフォルト 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. `nmcli`を使用して新しいルートを追加します:

```
sudo nmcli conn mod "自動イーサネット" +ipv4.routes "192.168.100.0/24 172.16.2.1"
```

3. まだ発効していない:

```
ip route
```

```
デフォルト 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
```

4. インターフェイスをリロードすると、それが有効になり、表示されます:

```
sudo nmcli conn up "自動イーサネット"
```

ip route

```
デフォルト 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
192.168.100.0/24 via 172.16.2.1 dev ens33
```

5. 再起動し、ルートが有効になったことを確認する：その場合は削除してください：

`ip route`

```
デフォルト 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
192.168.100.0/24 via 172.16.2.1 dev ens33
```

```
sudo nmcli conn mod "自動イーサネット" -ipv4.routes "192.168.100.0/24 172.16.2.1"
```

```
sudo nmcli conn up "自動イーサネット"
```

接続は正常にアクティブ化された (Dバス・アクティブ・パス：

↳ /org/freedesktop/NetworkManager/ActiveConnection/3)

`ip route`

```
デフォルト 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
```

6. コマンドラインからipでルートを設定することはできるが、再起動には耐えられない：

```
sudo ip route add 192.168.100.0/24 via 172.16.2.1
```

...

この方法で確立されたルートが永続的でないことを確認できる。

## 第29章

# LDAP



|      |              |     |
|------|--------------|-----|
| 29.1 | LDAP認証 ..... | 446 |
| 29.2 | ラボ .....     | 450 |



### 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、クライアント設定のテストに使用する小型のLDAPサーバーです。LDAP アプライアンスは約 512MB のメモリと 2GB のディスク領域を使用します。この演習は授業外で行ってください。

### 学習目標

このセッションを終えるまでに、あなたは次のことができるようになるはずだ:

- LDAP認証の目的を理解する。
- LDAPに関係する/etcの重要なファイルを知る
- これらのファイルを扱うためにどのユーティリティが利用できるかを知っておく。



### ご注意

\*\* これらのセクションは、一部または全部をオプションとすることができる。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。講師は、授業の経験や時間的制約に応じて、これらの項目を取り上げるか取り上げないかを選択することができます。



## 29.1 LDAP認証

# ライトウェイトディレクトリアクセスプロトコル

- ライトウェイトディレクトリアクセスプロトコル
- 認証およびユーザー情報サービスの提供
- ユーザー情報サービスのために**Kerberos**と共に使用されることが多い。
- 多くのシングルサインオンおよびアイデンティティ管理ソリューションで使用されている。

LDAPは、IPプロトコルを介してディレクトリサービス（電話帳を想像してください）を提供するためのx.500仕様から派生したものです。**LDAPは**、特に高速で計算が軽く、堅牢であるように設計されたデータベーススタイルの機能を提供する。**LDAPアプリケーション・プロトコル**は、トランスポート、ディレクトリ・レイアウト（構成）、データへのアクセス方法（読み取り、書き込み、削除など）を詳細に規定している。

しかし、**LDAPは認証と認可**の両方のデータを提供できる。LDAPは主に認可だけに使われ、認証には**Kerberos**（信頼されたサードパーティ認証）がよく使われる。興味深いことに、Microsoftの**Active Directory**サービスは（少なくとも認証/認可に関しては）、LDAPと**Kerberos**を結合させたものに過ぎない。



# LDAP認証

- 集中認証
- 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（Lightweight Directory Access Protocol）は、ネットワーク上で分散型ディレクトリサービスを利用・管理するための業界標準プロトコルであり、オープンでベンダーニュートラルであることを意図している。

集中型認証に **LDAP** を使用する場合、各システム（またはクライアント）は、ユーザ認証のために集中型 **LDAP** サーバに接続する。

LDAPは**TLS**（Transport Layer Security）と**SSL** Secure Sockets **Layer**に対応しています。**TLS/SSL**は安全な暗号化接続を提供します。

クライアント・コンフィギュレーションは、**LDAP**サーバーに接続するための最小限の情報を指定するだけでよい。LDAPサーバーのIPアドレス、検索ベース**DN**（ドメイン名）、オプションで**TLS**などの要素が認証に必要である。

追加のユーザ空間プログラムは**Openldap-clients**パッケージによって提供される。

# LDAP認証プロセス

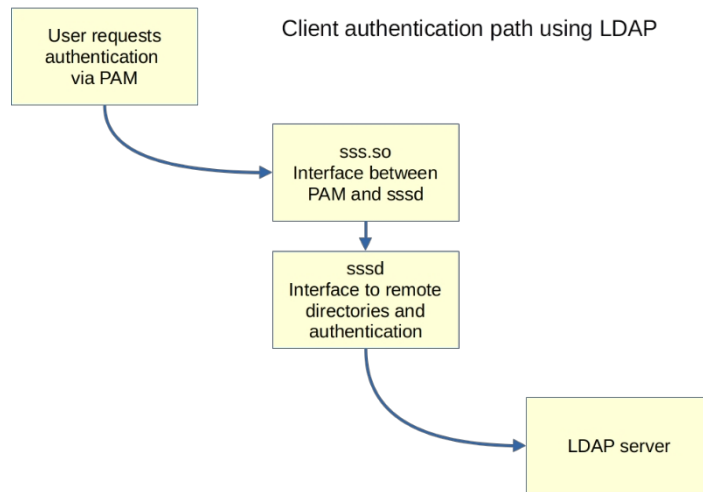


図 29.1: LDAPクライアント認証プロセス

上の図は、**LDAP/SSSD**が使用する情報経路を示している。**LDAP**サーバーへの接続は、プラグ可能な認証モジュールである**PAM**から始まる。PAMはLinuxのデフォルトの認証メカニズムであり、LDAPとSSSDはデフォルトの認証コンフィギュレーションに追加される拡張モジュールである。

- PAMは**pam.sss.so**モジュールを使用するように設定されている。
- **sss.so**と**sssd**は、sssdとLDAPを組み合わせた設定ファイルを使用する。

## /etc/sss/conf.d/00-sssd.conf

```

[sssd]
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 および sssd パッケージ

- **sssd** システム・セキュリティ・サービス・デーモン
- LDAP と併用する場合の sssd-ldap 設定メカニズム
- **sssd-tools** sssd で使用する各種ツール
- **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・削除するためのツールで、 **Ubuntu** 固有のもの。
- **openldap-clients** ldap-utils と同じです。 **CentOS**

LDAP認証用にクライアント・システムを設定すると、以下のファイルが変更されます:

`/etc/sss/conf.d/00-sss.conf`

`/etc/pam.d/common-session.conf`

`etc/sss/conf.d/00-sss.conf`ファイルには2桁の接頭辞があり、複数のコンフィギュレーション・ファイルが使用されている場合に順番に並べられるようになっている。

`etc/pam.d/`設定ファイルは、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。

-  CentOS-8 `/etc/pam.d/system-auth`
-  Ubuntu-22.04 `/etc/pam.d/common-session.conf`

これらのファイルは手動で編集できる。

以前のディストリビューションリリースでは、設定に特定のユーティリティが使用されていました。sssdとその仲間たちの採用により、クライアント設定の複雑さが大幅に軽減され、特別な設定ツールが不要になりました。テキストファイルの設定情報は、お好みのテキストエディタで設定できます。

## 29.2 ラボ

### 実習29.1: LDAPとTLSを使用した集中認証 - LDAPサーバーアプライアンスのインストール



#### 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、クライアント設定のテストに使用する小型の**LDAPサーバー**です。**LDAP** アプライアンスは約 512MB のメモリと 2GB のディスク領域を使用します。この演習は授業外で行ってください。

この演習では、集中型ネットワーク認証用の**LDAP**サーバー・アプライアンスをインストールする。**ターンキーの openldap**アプライアンス・サーバーは、さまざまなディストリビューションで利用できることから選択されました。このアプライアンスをインストールすると、新しい**VM**が作成される。**アプライアンスVM**は非常に小さく、わずか512MBのメモリを割り当て、1GB未満のディスク領域を使用します。ディスク容量は20GBまで動的に拡張可能ですが、使用量はかなり少ないままです。アプライアンスを入手するには多くのオプションがあり、VMWareや**Virtual-Box**と互換性のあるOVAイメージは以下のディレクトリで入手できる。また、**VMWare Player**と**VMWare Workstation**で動作する**vmdk**イメージファイルも以下のディレクトリにあります。

<https://training.linuxfoundation.org/cm/LFS207>

利用可能なオプションについては、<https://www.turnkeylinux.org/openldap>。

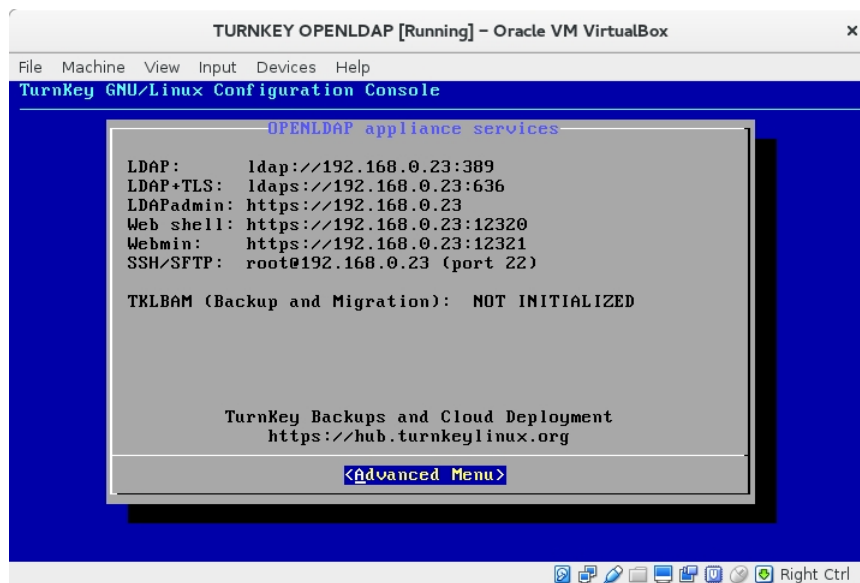


図 29.2: ターンキー OpenLDAP アプライアンス・サービス

#### 1. アプライアンスのインポート

- [turnkey-openldap-14.1-jessie-amd64.ova](#) ファイルが利用可能であることを確認してください（バージョンが異なる可能性があります）。
- **Virtual-Box Manager**を開く。
- File -> Import Appliance を選択します。
- **OpenLDAP アプライアンス**の場所 (またはブラウズ) を入力します。
- **Next (次へ)** を選択し、プロンプトに従って続行します。

2. 作成した **TURNKEY OPENLDAP** 仮想マシンを起動します。

3. 初回起動時に、以下のパラメーターで LDAP アプライアンスを構成する：

- ルートのパスワードは"pa\$\$w0rd"



- 管理者のパスワードID "pa\$\$w0rd"
- ドメインは "example.com"
- HUB"、"システム通知"、"更新 "サービスはない

Turnkey OpenLDAP アプライアンスサマリーが表示されます。

## 🔪演習29.2: LDAPとTLSを使用した集中認証 - LDAPサーバーのテストとユーザーの追加。

この演習では、LDAPサーバーの基本的な動作を確認する。具体的には、サーバーがクエリに応答し、POSIXグループとユーザーを追加できることを確認する。ユーザーを追加したら、wiresharkで接続を監視し、LDAPサーバーとの間でプレーンテキストの情報が転送されるのを観察します。平文のユーザー情報を持つことはセキュリティ・ポリシーに反するので、データ・ストリームを暗号化する。



### 非常に重要

このラボでは、Idap 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、どのような本番環境においても**推奨されません**。ラボの環境に合わせて、証明書の検証を無効にします。鍵の検証を行わない機能は、ラボでは便利ですが、本番システムでは**絶対に**使用しないでください。

この演習では、前回の演習で使用したマシンを使ってLDAPサーバーと通信します。

ready-for.shスクリプトが実行され、ソリューションとリソース・ファイルがロードされ、ディレクトリに展開されたと仮定します。

1. openldap-clientsパッケージを確認またはインストールするには、適切なコマンドを使用します。

#### 🐧Ubuntu

```
apt install sssd sssd-ldap sssd-tools oddjob-mkhomedir ldap-utils
```

#### 🐘CentOS

```
yum install sssd sssd-ldap sssd-tools oddjob-mkhomedir openldap-clients
```

2. 前の演習でTurnkey OpenLDAP Appliance Summary画面に表示されたLDAPサーバーのIPアドレスを使用して、example.comドメインの簡単なLDAP検索を実行します。example.com ドメインのベースレコードに関する情報が表示されるはずです。

```
ldapsearch -x -H ldap://192.168.0.23 -b "dc=example,dc=com" -s sub "objectclass=*
```

```
extended LDIF
#
LDAPv3
スコープのサブツリーを持つ # base
<dc=example,dc=com> # filter: (objectclass=*)
requesting: 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: 管理者

説明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=roups,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

# Idmaps, example.com

dn: ou=Idmaps,dc=example,dc=com  
objectClass: organizationalUnit  
objectClass: top  
ou: 地図

# samba, example.com

dn: cn=samba,dc=example,dc=com  
cn: サンバ

objectClass: simpleSecurityObject

objectClass: organizationalRole 説明:

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: 別名

# nssspam, example.com

dn: cn=nssspam,dc=example,dc=com  
cn: nssspam

objectClass: simpleSecurityObject

objectClass: organizationalRole 説明

: NSS/PAM アクセスアカウント

# 検索結果の検索

2

結果: 0 成功

# numResponses: 11

# numEntries: 10

3. **solutions**ディレクトリに含まれる**groups.ldif**ファイルを使用して、新しいグループレコードを追加する。

```
ldapadd -x -D "cn=admin,dc=example,dc=com" -W -H ldap://192.168.0.23 -f groups.ldif
```

LDAPパスワードを入力します：

新しいエントリ「cn=luser1,ou=Groups,dc=example,dc=com」を追加する。

4. **ldapsearch**コマンドで新しいグループが表示されるはずだ。

```
ldapsearch -x -H ldap://192.168.0.23 -b "dc=example,dc=com" -s sub "objectclass=*" 拡張
```

```
LDIF
#
LDAPv3
スコープのサブツリーを持つ # base
<dc=example,dc=com> # filter: (objectclass=*)
requesting: 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
説明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
```

```
Idmaps, example.com
dn: ou=Idmaps,dc=example,dc=com
objectClass: organizationalUnit
```

```

objectClass: top
ou: 地図

samba, example.com
dn: cn=samba,dc=example,dc=com
cn: サンバ
objectClass: simpleSecurityObject
objectClass: organizationalRole 説明:
明: 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: 別名

nssspam, example.com
dn: cn=nssspam,dc=example,dc=com
cn: nssspam
objectClass: simpleSecurityObject
objectClass: organizationalRole 説明:
NSS/PAM アクセスアカウント

luser1、グループ、example.com
dn: cn=luser1,ou=Groups,dc=example,dc=com
cn: luser1
objectClass: posixGroup
objectClass: top
gidNumber: 999001
memberUid: luser1

検索結果の検索
2
結果: 0 成功

numResponses: 12
numEntries: 11

```

上の出力で、新しいグループレコードを追加したときに **numResponses** と **numEntries** が増えていることに注目してください。

5. **solutions**ディレクトリに含まれる**users.ldif**ファイルを使用して、新しいユーザー・レコードを追加する。

```
ldapadd -x -D "cn=admin,dc=example,dc=com" -W -H ldap://192.168.0.23 -f users.ldif
```

LDAPパスワードを入力します:  
新しいエントリ "cn=luser1,dc=example,dc=com" の追加

6. **ldapsearch**コマンドは新しいユーザーを表示するはずだ。

```
ldapsearch -x -H ldap://192.168.0.23 -b "dc=example,dc=com" -s sub "objectclass=*"
```

```
extended LDIF
#
LDAPv3
スコープのサブツリーを持つ # base
<dc=example,dc=com> # filter: (objectclass=*)
requesting: 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
説明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=roups,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

Idmaps, example.com
dn: ou=Idmaps,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou: 地図

samba, example.com
dn: cn=samba,dc=example,dc=com
cn: サンバ
objectClass: simpleSecurityObject
objectClass: organizationalRole 説明
: 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: 別名

nsspam, example.com
dn: cn=nsspam,dc=example,dc=com

```

cn: nssspam  
objectClass: simpleSecurityObject  
objectClass: organizationalRole 説明  
: NSS/PAM アクセスアカウント



```
luser1、グループ、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

検索結果の検索
2
結果: 0 成功

numResponses: 13

numEntries: 12
```

7. wiresharkをインストールするか、インストールされていることを確認する。

```
yum install wireshark-gnome
wireshark
```

8. wiresharkでldapsearchコマンドをキャプチャすると、データがクリアテキストであることに気づく。

Wiresharkには優れたフィルターがあり、"tcp and (tcp.port==389) or (tcp.port==636)"というフィルターを使って、**wiresharkに次のようなことをさせることができる。**

**LDAP通信のみを表示する。**

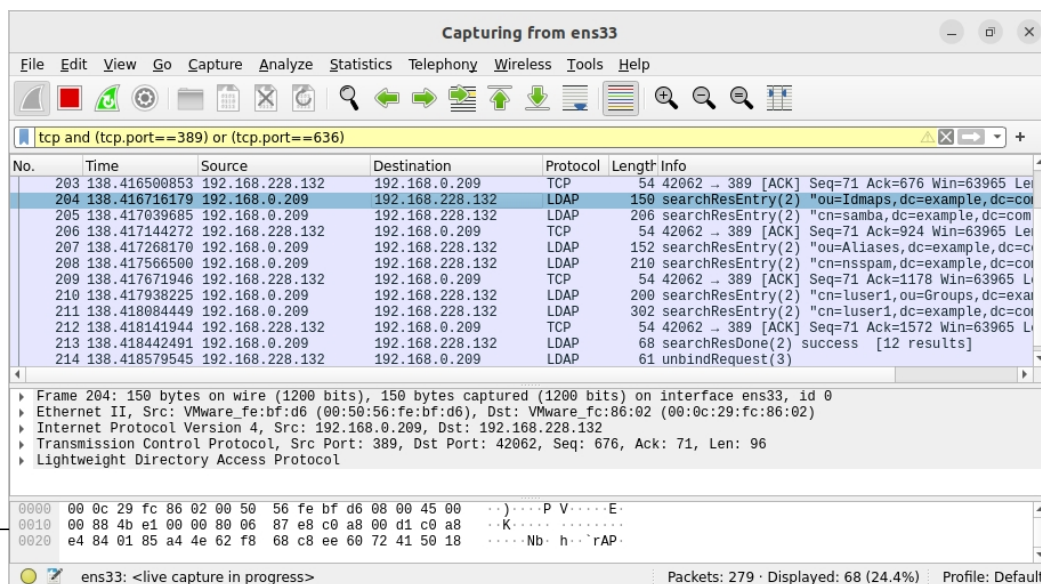


図29.3: Wireshark LDAPパケットトレース

## 🔗練習問題 29.3: ユーザー認証にLDAPを使用した集中型認証

この演習では、集中認証にLDAPを使用するようにクライアントを設定します。



### 非常に重要

CentOS-8とUbuntu 20.04から、リモート・ディレクトリへのアクセスと認証の管理にSystem Security Services Daemon (sssd) が使用されるようになった。sssdの使用と設定は、参加ディストリビューション間で同じです。

1. 必要なパッケージをインストールします。

```
apt install sssd sssd-ldap ldap-utils oddjob-mkhomedir
```

2. sssdコンフィギュレーションを作成または更新し、以下の要素を含める。

ldap uri フィールド。



### /etc/sss/conf.d/00-sss.conf

```
[sssd]
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. sssd.confファイルのパーミッションを確認し、設定する。

```
chmod 600 /etc/sss/conf.d/00-sss.conf
chown root.root /etc/sss/conf.d/00-sss.conf
```

4. oddjobが自動的にホームディレクトリを作成するように設定します。以下の例のように、`/etc/pam.d/common-session`にoddjobを追加してください:



### /etc/pam.d/common-session.conf

```
セッション必須
 pam_unix.so pam_oddjob
 pam_sss.so
```

任意 セッション

任意

セッション任意

5. サービスを再起動して変更を反映させる。

```
systemctl restart sssd oddjobd
systemctl enable sssd oddjobd
```

6. **ldap**サーバーをテストする。
7. ユーザー**luser1**のユーザー情報が利用可能であることを確認する。

```
getent passwd luser1
```

と答えるべきだ:

```
luser1:*.999001:999001:luser1:/home/users/luser1:
```

8. ユーザー**luser1**のユーザー認証が機能していることをテストする。パスワードはpasswordでなければならない。利用可能なホーム・ディレクトリがないかもしれない。

```
ssh luser1@localhost
```

9. データストリームがクリアテキストであることを確認する。
  - ポート389と636をフィルタリングする**wireshark**セッションを開く。
  - **wireshark** がパケットをキャプチャしている間、ユーザー **luser1** として **localhost** に **ssh** でログオンしたりログオフしたりする。
  - **wireshark**の出力を見る

10. 証明書の検証を無効にする。



### 非常に重要

**本番システムでは、証明書の検証を無効にすることは推奨されない (NOT RECOMMENDED)。**

お気に入りのエディターで/etc/sss/sss.confファイルを編集する。domain/default]セクションに次の行を追加する:

```
ldap_tls_reqcert = never
```

**sss**サービスを再起動する。

```
systemctl restart sssd
```

11. **luser1**のユーザー情報が**LDAP**から提供されていることをテストする。

```
getent passwd luser1
```

12. **luser1**のユーザー認証が機能していることをテストする。

```
ssh luser1@localhost
```

13. **LDAP**サーバーとの間のデータストリームが暗号化されていることを確認する。

- ポート389と636をフィルタリングする**wireshark**セッションを開く。

- **wireshark** がパケットをキャプチャしている間、ユーザー `luser1` として `localhost` に `ssh` でログオンしたりログオフしたりする。
- **wireshark** の出力を観察する。

14. 完了したら元の設定に戻す。

## 第30章

# ファイアウォール



|      |                 |     |
|------|-----------------|-----|
| 30.1 | ファイアウォール.....   | 460 |
| 30.2 | インターフェイス.....   | 463 |
| 30.3 | ファイアウォール.....   | 465 |
| 30.4 | ゾーン .....       | 468 |
| 30.5 | ソース管理 .....     | 472 |
| 30.6 | サービスと港湾管理 ..... | 473 |
| 30.8 | 研究所.....        | 481 |

### 学習目標

このセッションを終えるまでに、あなたは次のことができるようになるはずだ:

- 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を知る。
- **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る。

459





## 30.1 ファイアウォール

# ファイアウォールとは？

- 最新のネットワークに不可欠なセキュリティ設備
- システムおよびネットワークへの入出アクセスを制御する。
- 侵入やその他の攻撃ベクトルからの保護
- 特定のインターフェイスやアドレスの信頼レベルを制御する
- 信頼レベルとネットワーク・トポグラフィに応じた柔軟なバリア制御

## ルール

**ファイアウォール**は、すべてのネットワーク・トラフィックを監視・制御するネットワーク・セキュリティ・システムである。ネットワーク接続とパケットの送受信の両方に**ルール**を適用し、特定の接続の信頼度に応じて柔軟なバリア（すなわちファイアウォール）を構築します。

ファイアウォールには、ハードウェアベースのものとソフトウェアベースのものがある。ファイアウォールは、ネットワーク・ルーターにも、個々のコンピューター（ネットワーク・ノード）にもあります。多くのファイアウォールはルーティング機能も持っています。

# パケットフィルタリング

- ファイアウォール構築の基本
- ネットワーク・ストリームの各パケットは検査される
- システムで定義されたルールに従う
- アクションには、アクセプト、ドロップ、マングル、リダイレクトなどがある。

ほとんどのファイアウォールはパケットフィルタリングに基づいている。

ネットワーク上では、情報はパケットという形で伝送される：

- ヘッダー
- ペイロード
- フッター

ヘッダーとフッターには、宛先アドレスと送信元アドレス、どの種類のパケットか、どのプロトコルに従うか、さまざまなフラグ、ストリーム中のどのパケット番号か、その他送信に関するあらゆる種類のメタデータなどの情報が含まれている。

実際のデータはペイロードにある。

パケットフィルタリングは、アプリケーション、トランスポート、ネットワーク、データリンクなど、ネットワーク伝送の1つ以上の段階でパケットを遮断する。

ファイアウォールは、各パケットが：

- 内容、住所などに基づいて受理または拒否される。

- 別のアドレスにリダイレクト
- 安全上の理由から検査
- その他

パケットフィルタリングの結果として取られるべきルールやアクションを確立するための様々なユーティリティが存在する。



## ファイアウォール世代

- **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、または送信する。
- **ステートフル・フィルター**: コネクションの状態を調べ、新しいコネクションの一部か、既存のコネクションか、あるいは何もないかを判断する。
- **アプリケーション層ファイアウォール**: アプリケーションとプロトコルのパケットを認識する。

初期のファイアウォール（1980年代後半までさかのぼる）は、**パケットフィルタリングに基づいていた**。各ネットワークパケットの内容が検査され、ドロップされるか、拒否されるか、あるいはそのまま送信された。各ネットワーク・パケットの内容は検査され、ドロップされるか、拒否されるか、送信されるかのいずれかであった。パケットがどのようなトラフィックの流れの一部であるかという**接続状態については考慮されなかった**。

次世代のファイアウォールは**ステートフル・フィルターに基づいており**、パケットの**接続状態**（新しい接続なのか、既存の接続の一部なのか、あるいは接続の一部ではないのか）も調べる。サービス拒否攻撃は、この種のファイアウォールを圧倒しようと砲撃することができる。

第3世代のファイアウォールは**アプリケーション・レイヤー・ファイアウォール**と呼ばれ、接続が使用しているアプリケーションやプロトコルの種類を認識します。通常のフローに含まれるべきでないものをブロックすることができます。

# ファイアウォールのインターフェイスとツール

- IPテーブル
- ファイアウォールコマンド
- ユーエフ
- グラフィカル・インターフェース
  - システム設定ファイアウォール
  - ファイアウォール設定
  - グフ
  - ヤスト
- ショアウォールのような他のユーティリティもある。
- コマンドラインツールは、ディストリビューションによってあまり差がない

システムのファイアウォールの設定は、以下の方法で行うことができる：

- コマンドラインから比較的低レベルのツールを使い、[/etc/](#)にあるさまざまな設定ファイルを編集する。

サブディレクトリ・ツリー：**iptables**、**firewall-cmd**、**ufw**など。

- 堅牢なグラフィカル・インターフェースの使用：**system-config-firewall**、**firewall-config**、**gufw**、**yast**など。

私たちは以下の理由から、下位レベルのツールに取り組む：

- グラフィカルなものほど頻繁に変わるものではない。
- 彼らはより大きな能力を持つ傾向がある。

- **GUI**はかなり異なる傾向があり、それぞれ1つのディストリビューション・ファミリーに限定されている。

デメリットは、最初のうちは習得が難しく感じられることだ。

以下では、**firewall-cmd**と**firewall-config**の両方を含む最新の**firewalld**パッケージの使用に集中します。firewall-cmdとfirewall-configの両方を含む最新のfirewalldパッケージを使用することに集中します。デフォルトでfirewalldがないディストリビューションでは、演習で必要に応じて行うように、ソースから割と簡単にインストールすることができます。



イス

## iptablesが使えない理由

- 現在、ほとんどのファイアウォールはユーザー側で**iptables**パッケージを使っている。
- 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る。
- **iptables**は有用な機能に到達するまでに時間がかかるため、教えないことにした。
- しかし、**iptables**については、**Linux Foundation** システム管理者コース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**

見てください:

<https://training.linuxfoundation.org/linux-courses/system-administration-training/linux-for-system-engineers>



## 30.3 ファイアウォールド

### firewalld と firewall-cmd

- ダイナミックファイアウォールマネージャ (firewalld)
- ゾーンと連動
- IPv4とIPv6をサポート
- ランタイムとパーマネントの設定を分ける
- メインツールは**firewall-cmd**
- **iptables**ユーティリティを置き換える
- **systemd**ユニバースの一部
- 徐々に主要なディストリビューション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。

firewalldは**ダイナミックファイアウォールマネージャー**です。ネットワークインターフェースや接続の信頼レベルを定義したネットワーク/ファイアウォール・**ゾーン**を利用する。**IPv4とIPv6の両方**のプロトコルをサポートしています。

さらに、**ランタイム**と**永続的な**（永続的な）コンフィギュレーションの変更を分離し、ファイアウォールルールを追加するためのサービスやアプリケーションのためのインターフェースも含んでいる。

設定ファイルは**/etc/firewalld**と**/usr/lib/firewalld**に保存されます。**/etc/firewalld**にあるファイルは他のディレクトリにあるファイルより優先され、システム管理者が作業すべきものです。

コマンドラインツールは、実はこれから説明する**firewall-cmd**である

。先に進む前に、次のコマンドを実行することをお勧めする：

**firewall-cmd --help**

使用法: firewall-cmd [OPTIONS...].

....

**ステータスオプション** firewalldの状態を返し、表示する

**--州**  
**-完全再読み込み** 3秒の遅延を伴って状態を再読み込みする  
**--リロード** 情報を保持する  
**をリロードする**

**--runtime-to-permanent** ランタイム設定からパーマネントを作成する。

....

これは約200行に及ぶので、ここで紹介するには長すぎる。しかし、ほとんどすべてのオプションに名前がつけられているので、一目瞭然であることがわかるだろう。

サービスとしては、**firewalld**が古い**iptables**に**取って代わる**。両方のサービスを同時に実行するのは誤りである。

## 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/停止する必要があります:

```
sudo systemctl [enable/disable] firewalld
sudo systemctl [start/stop] firewalld
```

- ステータスを表示する:

```
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フォワーディング**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
```

firewalldは、ファイアウォールを使用し設定するために実行する必要があるサービスで、有効/無効、または上記のような通常の方法で起動/停止します。現在の状態は、以下のいずれかの方法で表示できます：

`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 (実行中) 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
ング
```

**IPv4**を使用しているときに複数のネットワーク・インターフェイスがある場合、**ipフォワーディング**をオンにしなければならないことに注意。これは実行時に以下のどちらかの方法で行うことができる：

```
sudo sysctl net.ipv4.ip_forward=1
echo 1 > /proc/sys/net/ipv4/ip_forward
```

(ここで、2番目のコマンドはrootとして実行しないと**echo**が正しく動作しない)。しかし、これは永続的ではない。そのためには、`/etc/sysctl.conf`に`net.ipv4.ip_forward=1`という行を追加し、再起動するか、再起動せずに`sudo sysctl -p`と入力して新しい設定を読み込む必要がある。

## 30.4 ゾーン

# ゾーン

- 各ゾーンには、信頼レベルとパケット管理動作が定義されている：
  - **drop** すべての受信パケットをドロップ
  - **ブロック** すべての着信ネットワーク接続を拒否
  - **public** デフォルトでは信頼なし
  - マスカレードによる**外部公開**
  - **dmz** (非武装地帯) 一部の接続に一部のサービスを許可する
  - **仕事** 信頼できる（ある程度）特定のネットワーク接続。
  - **ホーム** 特定のネットワーク接続を（ほとんど）信頼する。
  - **内部 作業区域**に似ている。
  - **信頼できる** すべての接続を許可する

**firewalld**はゾーンで動作し、それぞれのゾーンは定義された信頼レベルと、送受信パケットに対する特定の既知の動作を持っている。各インターフェイスは特定のゾーンに属しますが（通常、どのゾーンが適用されるかを**firewalld**に通知するのは**NetworkManager**です）、これは**firewall-cmd**または**firewall-config** GUIで変更できます。ゾーンは以下の通りです：

- **ドロップする**：すべての着信パケットは応答なしでドロップされる。発信コネクションのみが許可される。
- **ブロックする**：すべての着信ネットワーク接続が拒否される。許可されるのはシステム内からの接続のみです。
- **公開する**：意識的に選ばれた特定の着信接続のみが許可される。
- **external**：ルーターなどで**マスカレード**を使用する場合に使用する。信頼レベルは**public**と同じ。
- **dmz**：(非武装地帯) 一部の(すべてではないが)サービスへのアクセスを一般に許可する場合に使用する。特定の着信接続のみが許可される。

- **働く**：完全にではないが）接続されたノードが有害でないことを信頼する。特定の着信接続のみが許可される。
- **ホーム**：他のネットワークノードをほとんど信頼しているが、どの着信接続を許可するかを選択している。
- **内部**：ワークゾーンと同様。
- **信頼できる**：すべてのネットワーク接続を許可する。

システムのインストール時、すべてのLinuxディストリビューションではないにせよ、ほとんどの場合、すべてのインターフェースのデフォルトとして**パブリックゾーン**が選択される。上記のいくつかのゾーンの違いは明白ではないので、ここでそれほど詳しく説明する必要はないが、必要以上にオープンゾーンを使用すべきではないことに注意されたい。



## ゾーン・マネジメントの例 I

- デフォルトゾーンを取得する:

```
sudo firewall-cmd --get-default-zone
```

public

- 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リストアップします:

```
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 仕事

`firewall-cmd --help`

....

### ゾーン・オプション

--get-default-zone 接続とインターフェイスのデフォルトゾーンを表示する

--デフォルトゾーン=<ゾーン>

デフォルトゾーンの設定

--get-active-zones 現在アクティブなゾーンを印刷する

--get-zones 定義済みのゾーンを印刷する [P]

--get-services 定義済みサービスを印刷する [P]

--get-icmptypes 定義済みの icmptypes を表示する。

--get-zone-of-interface=<interface>

インターフェイスがバインドされているゾーンの名前を印刷する [P]。

--get-zone-of-source=<source>[/<mask>].

ソース[/mask]がバインドされているゾーンの名前を[P]に印刷する。

--list-all-zones すべてのゾーンで追加されたもの、または有効になっている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。      新しいゾーンを追加する。

--delete-zone=<zone> 既存のゾーンを削除する。

--zone=<zone> オプションの設定や問い合わせにこのゾーンを使う。

--get-target      ゾーンターゲットを取得 [P] [Z]

-セットターゲット=<ターゲット>

ゾーンターゲットの設定 [P] [Z]

## ゾーン・マネジメントの例 II

- デフォルトのゾーンを信頼済みに変更し、その後元に戻す:

```
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一時的に特定のゾーンに割り当てる:

```
sudo firewall-cmd --zone=internal --change-interface=enol 成功
```

- 永久にそうするために:

```
sudo firewall-cmd --permanent --zone=internal --change-interface=enol 成功
```

これは/etc/firewalld/zones/internal.xmlファイルを作成する。

**firewalld**を制御するすべての作業は、**firewall-cmd**プログラムを通して行われます。より詳細な情報は

`man firewalld-cmd`

## ゾーン・マネジメントの例 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=enol 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)

ターゲット: デフォルト

icmp-block-inversion: no

interface: enol sources:

サービス: chromecast libvirt libvirt-tls nfs nfs3 rsyncd ssh

ports:

protocols:

masquerade: no

forward-ports:

source-ports:

icmp-blocks:

rich rules:



### もっと知りたい?

firewall-cmdに関するまとまったリファレンスは、

<https://firewalld.org/documentation/man-pages/firewall-cmd.html>

。

## 30.5 ソース管理

### ソース管理

- インターフェイスだけでなく、特定のネットワークアドレスにゾーンをバインドする
- 関連アドレス以外のパケットはデフォルトゾーンへ
- **firewall-cmd**で割り当てる:

```
sudo firewall-cmd --permanent --zone=trusted --add-source=192.168.1.0/24 成功
```

どのゾーンも、ネットワーク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もバインドすることができる。以下の場合、パケットはゾーンに関連付けられる:

- その送信元アドレスは既にゾーンにバインドされている:
- それはゾーンにバインドされたインターフェースから来る。

上記の条件に当てはまらないパケットは、デフォルトゾーン(つまり通常は**パブリック**)に割り当てられる。ソースをゾーンに割り当てる(恒久的に)

```
sudo firewall-cmd --permanent --zone=trusted --add-source=192.168.1.0/24
```

成功

これは、192.168.1.xのIPアドレスを持つ人は誰でも**信頼された**ゾーンに追加されるというものだ。

remove-sourceオプションを使用すると、ゾーンから以前に割り当てられたソースを 削除でき、--change-sourceを使用するとゾーンを変更することに注意。

でゾーンにバインドされたソースをリストすることができる:

```
sudo firewall-cmd --permanent --zone=trusted --list-sources 192.168.1.0/24
```

上記のどちらのコマンドでも、--permanentオプションを省くと、現在の実行時の動作だけが得られる。

## 30.6 サービスと港湾管理

### サービス管理

- 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
```

```
success
```

```
sudo firewall-cmd --reload
```

これまでのところ、特定のインターフェースやアドレスをゾーンに割り当ててきたが、ゾーン内でアクセス可能なサービスやポートについては定義していなかった。利用可能なすべてのサービスを見るには

`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
```

または、特定のゾーンで現在アクセス可能なものを確認する：

`sudo firewall-cmd --list-services --zone=public`

```
dhcpv6-client ssh
```

ゾーンにサービスを追加するには

`sudo firewall-cmd --permanent --zone=home --add-service=dhcp`

```
成功
```

`sudo firewall-cmd --reload`

変更を有効にするには、`--reload`を指定した2番目のコマンドが必要である。のファイルを編集して、新しいサービスを追加することも可能である。

[/etc/firewalld/services.](#)

## 港湾管理

- 港湾管理：

```
sudo firewall-cmd --zone=home --list-ports
sudo firewall-cmd --zone=home --add-port=21/tcp
```



港湾管理はサービス管理とよく似ている：

```
sudo firewall-cmd --zone=home --add-port=21/tcp success
```

```
sudo firewall-cmd --zone=home --list-ports
```

```
21/tcp
```

`etc/services`を見れば、ポート21が**ftp**に対応していることがわかる：

```
$ 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す。**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 (送信元ネットワークアドレス変換)
- MASQUERADE、SNATのバリエーション

DNAT、SNAT、MASQUERADEについては次のページで説明する。

# DNAT

**DNAT (Destination Network Address Translation: 宛先ネットワークアドレス変換)** は、パケットがインターネットからシステムに入る際に、終端アドレス（「宛先」IPアドレス）と、場合によっては宛先ポートを変更する。

- パケットが送信されると、そのIPアドレスは **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は、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、目的のサービスに到達するために必要である。

**DNAT**ファイアウォールルールは通常、**NAT**テーブルのプレルーティングチェーンに配置される。これは、**インターネット**に面したアダプターと、ウェブサーバーなどのサービスで使用するパブリックアドレスとなる。

ファイアウォールは通常パススルーデバイス（ローカルサービスはほとんどない）であるため、パケットは検査され、宛先ポートが検査され、適用可能なルールがあるかどうかを確認される。私たちのルールは、パケットを以下のように検査する：

- 宛先IPアドレス：ファイアウォールのものでなければならない。
- 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**ルールは、宛先IPアドレスをファイアウォールの背後にあるウェブサーバーのアドレスに変更する。
- 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**される。
- ウェブサーバーはパケットを受け入れ処理し、応答を送信する。

## DNATの例

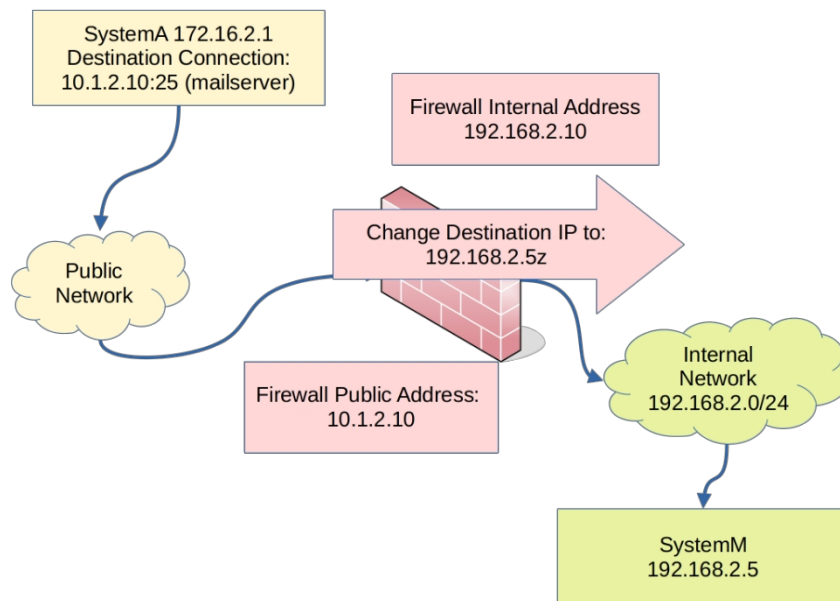


図30.1: 宛先NAT

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示している。

- SystemA(IP172.16.2.1)は、メール配信のためにsystemM(IP10.1.2.25)のアドレスを検索する。
- 宛先ポート番号は、サービスのポート番号 (smtp=25) になる。
- 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) に変更する。
- 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**される。
- メールサーバーはパケットを受け入れ処理し、返信を送る。

## SNATとMASQUERADE

**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、システムから離れる際に、パケットのソース（「送信元」）IPアドレスを変更する。

- 変換は、内部ネットワークで使われるアドレスをインターネットへの露出から保護するために必要です。
- ほとんどの内部ネットワークはプライベート・ネットワークであり、ルーティング可能ではない。

- アウトバウンドインターフェースのIPアドレスがDHCPによって提供される場合、**MASQUERADE**オプションは**SNAT**を実装するために使用される。
- マスカレード・オプションは、**SNATよりも**多くの計算資源を必要とする。
- アウトバウンド・インターフェースがスタティックの場合、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る。
- SNATとマスカレードは**NAT**テーブルのポストルーティングチェーンにある。

## SNATの例

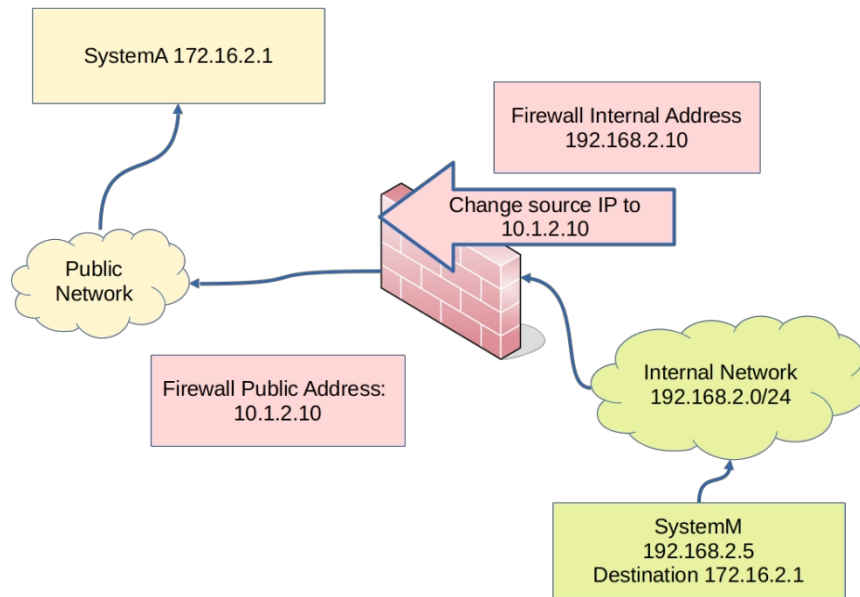


図30.2: ソースNATとMASQUERADE

**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る。通常、ポストルーティングとは、パケットがインターネットに向かう途中、システムから離れる際の最後の検査または修正ポイントである。

- 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ものである。
- の場合、パケットのソース・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る。

## 30.8 研究室

### 🔗演習30.1: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には**firewalld**パッケージ（**firewall-cmd**多目的ユーティリティを含む）が用意されているが、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ない。

まず、それがすでにインストールされているかどうかを確認する必要があります。

どの**firewalld** **firewall-cmd**か。

```
/usr/sbin/firewalld
/usr/bin/firewall-cmd
```

プログラムを見つけられなかった場合は、ディストリビューションに応じて以下のいずれかを通常の方法でインストールする必要がある：

```
sudo dnf install firewalld
sudo zypper install firewalld
sudo apt install firewalld
```

これに失敗した場合、あなたのディストリビューションでは**firewalld**パッケージが利用できません。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・リポジトリを入手するか、最新のtarボールを簡単にダウンロードすることができる。

それから、（現在のバージョンが何であれ）ソースからインストールする一般的な手順に従わなければならない：

```
tar xvf firewalld-0.3.13.tar.bz2
cd firewalld-0.3.13
./configure
作る
sudo make install
```

このソースにはアンインストールターゲットもある：

```
sudo make uninstall
```

万が一、後悔したときのために。

`./configure` ステップで、ライブラリの欠落などの不備があれば、それに対処しなければなりません。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ような問題を処理してくれますが、ソースからインストールする場合は問題があります。**Linux Foundation**の**ready-for.sh**スクリプトをあなたのシステムで実行していれば、問題が発生する可能性は低いでしょう。

### 🔗演習30.2:firewall-cmdを調べる

ここまでで、**firewalld**パッケージの使い方のほんの一部を紹介したに過ぎない。ほとんどのことは**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る。この感覚をつかむには、実際にやってみるしかない:

#### firewall-cmd --help

```

 使用法: firewall-cmd [OPTIONS...].

 サービスオプション
 --new-service=<service>
 新しいサービスを追加する[Pのみ]
 --delete-service=<service>
 既存サービスの削除[Pのみ]

```



これは**RHEL 8**システムで409行あるので、ここでは再現しない。

また、`/etc`にある様々なコンフィギュレーションファイルの説明や、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30.3: 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、それらが現在表示されていることを確認する。

### ✅解答 30.3

```
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

を実行した場合に注意してください。

```
sudo firewall-cmd --reload
sudo firewall-cmd --list-services --zone=public
```

dhcpv6-client ssh

新しいサービスを追加すると、リストから消えてしまうのだ！この不思議な動作はサービスを追加するときに `--permanent` フラグを付け、`--reload` オプションは、既知の永続的サービスのみをリロードする。

### 🔪練習30.4: ファイアウォール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**、**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、単に配布に柔軟に対応したいので、コマンドラインアプローチに集中しました。しかし、ほとんどの比較的単純なファイアウォール設定タスクは、GUIからより少ない暗記で効率的に行うことができます。

ファイアウォール設定 GUI を起動したら、**パブリックゾーン**に **http** と **https** を追加する前回の演習を実行し、それが有効であることを確認する。

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てください。

### 練習30.5: 本当のIPアドレスを特定する

この演習では、**NAT**の最も一般的な使い方である、**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封する方法を見ていきます。

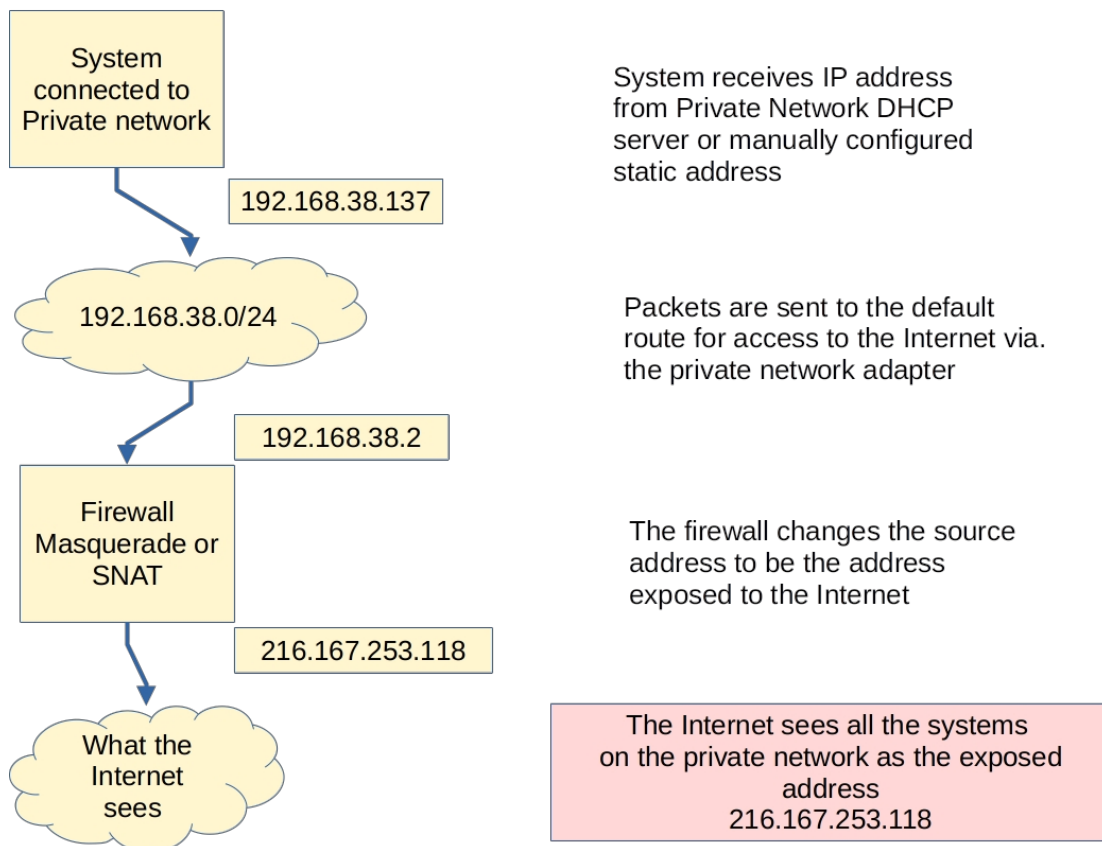


図30.3: SNATまたはMASQUERADEの例

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・アドレスを取得する。**ip**コマンドはうまく機能し、必要でない余分な情報は以下のよう  
に削除できる。

```
sudo ip a | grep " inet "
```

```
inet 127.0.0.1/8 scope host lo
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、**インターネット上ではルーティング不可能なアドレス**です。  
**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詳細は**、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ネクストIPアドレスを見つけるには、**デフォルトルート**を調べる。

```
sudo ip r
```

```
デフォルト via 192.168.38.2 dev ens33 proto dhcp metric 100
```

ネクストホップが**192.168.38.2**のIPアドレスを持つシステムであることがわかる。

3. システム**192.168.38.2**は、この環境の本当のファイアウォールなので、ログインできない。世界が私たちのシステムをどのように見ているのか抽出してみよう。

コマンドラインから:

```
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つを選択する：

#### **私のIPアドレスは何ですか**

外部との通信に使われるアドレスはプライベート・アドレスではなく、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、**ソース・ネットワーク変換**の一例である。

または**マスカレード**

## 第31章

# System Init: systemdの歴史とカスタマイズ



|      |                   |     |
|------|-------------------|-----|
| 31.1 | イニシャルプロセス .....   | 486 |
| 31.2 | スタートアップの選択肢 ..... | 487 |
| 31.3 | システムド .....       | 488 |
| 31.4 | systemctl .....   | 490 |
| 31.5 | 研究室 .....         | 491 |

### 学習目標

このセッションを終えるまでに、あなたは次のことができるようになるはずだ:

- SysVinitとUpstartを置き換えるためにsystemdがどのように生まれたかを理解する。
- systemd の主な機能を説明してください。
- systemctl を使って systemd でサービスを設定・制御する。
- ユーザーのsystemdユニットとドロップインファイルがどのように機能するかを見る。



## 31.1 イニシャルプロセス

### 開始

- `/sbin/init`
- すべてのプロセスの親
- 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:
  - システムデーモン
  - アップスタート
  - シスビット
- 主要なディストリビューションはすべて **systemd** に移行した

`/sbin/init` (通常は単に **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。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が、伝統的にすべてのユーザープロセスの親とみなされてきました。

**init** は、ブートプロセスの後段を調整し、環境のあらゆる面を設定し、システムへのログインに必要なプロセスを開始する。**init** はまた、プロセスが終了したときの後始末においてカーネルと密接に連携する。

伝統的に、ほとんど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 **SysVinit** を **init** プロセスのベースとしていました。しかし、このスキームは数十年前、かなり異なる状況下で開発されたものだ:

- 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・コンピューターやラップトップ、その他のデバイスではない) だった。
- ターゲットはシングルプロセッサのシステムだった。
- スタートアップ (およびシャットダウン) 時間は重要な問題ではなく、物事を正しく進めることよりもはるかに重要度が低かった。



スタートアップは、一連の連続したステージ（ラン・レベルと呼ばれる）に分割されたシリアル・プロセスとみなされていた。したがって、スタートアップは、複数のプロセッサやコアで実行できる並列処理を利用することは容易ではなかった。

第二に、シャットダウン／再起動は比較的まれな出来事とみなされ、それにかかる正確な時間は重要視されなかった。今日、Linuxシステムは通常、数秒で起動する。

現代のシステムは、より強化された能力を持つ新しい方法を必要としている。

## 31.2 スタートアップの選択肢

# スタートアップの選択肢

### ・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）と**RHEL 6**とそのクローンに採用された。
- 様々な組み込み機器やモバイル機器にも使用された

### ・システムデー

- Fedoraは2011年にこれを採用した最初の主要ディストリビューションである。
- RHELと**SUSE**に続く
- **Ubuntu 16.04**は**Upstart**を**systemd**に置き換えた。
- 重要な**Linux**ディストリビューションはすべて**systemd**をベースにしている。

**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法が開発されました。他にもありますが、Enterprise ディストリビューターは、**Upstart** と **systemd** という 2 つの主要な方式を採用しました。

他の主要な**Linux**ディストリビューションはすべて**systemd**を採用し、デフォルトとして設定している。Ubuntuでさえ**アップスタート**が有利

**systemd**への移行は自明なことではなく、バグや欠落した機能は非常に使いづらくなる可能性があったため、レガシーソフトウェアのために必要不可欠な互換性レイヤーが採用され、今でも存在しています。したがって、**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、色とりどりの個性によって、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ない。しかし、この聖戦は、私たちがここにいる限りでは終わったことだ。

次の章では、**systemd** だけに集中します。ほとんどの場合、**SysVinit** も **Upstart** も無視します。  
(もはや重要な意味では使われていない)。

## 31.3 システムデー

### systemd の特徴

- 従来のinitシステムより高速に起動
- 積極的な並列化機能を提供
- サービス起動にソケットとDバス起動を使用
- 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が可能
- **cgroups**を使用してプロセスを追跡
- マウントポイントおよびオートマウントポイントの維持
- 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の実装
- **cron**と同様の機能を含む、詳細な開始と停止の機能を提供します。

Linuxのsystemdシステムとセッション・マネージャーは、現在ほとんどの主要ディストリビューショ

ンで主流となっている。**bash**スクリプトの代わりに、**systemd**は.serviceファイルを使います。

きめ細かなロギングは、デフォルトではユニット名を用いて提供される。

root権限なしでユーザープロセスを追跡・作成することも、**systemd**の強力な機能だ。

## systemd 設定ファイルの場所

**systemd** はユニットファイルを複数の場所に置くことができ、root 以外のユーザーで実行したり、システムユニットファイルから継承したりすることができます。以下は典型的な場所です（ディストリビューションによって異なるかもしれません）：

システム・ユニット・ファイル：

- `/etc/systemd/system`
- `/lib/systemd/system`



ユーザー・ユニット・ファイル：

- `/etc/systemd/user`
- `~/.config/systemd/user`

ほとんどのシステムでは、`/lib/systemd/`は`/usr/lib/systemd/`へのシンボリックリンクである。

📌 Debianの標準的なシステム・ユニット・ファイルの例である：

**/lib/systemd/system/dbus.service****[ユニット]**

説明=D バスシステムメッセージバス

ドキュメンテーション=man:dbus-

daemon(1)

Requires=dbus.socket

**[サービス]**

ExecStart=/usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation

↳ --syslog-only

ExecReload=/usr/bin/dbus-send --print-reply --system --type=method\_call --dest=org.freedesktop.DBus / とします

。

↳ org.freedesktop.DBus.ReloadConfig

OOMScoreAdjust=-900

ユニット・ファイルは非常に複雑になることがある。この例では、ドキュメントへのリンクを含める方法や、サービスを開始したり再起動したりするためのさまざまな方法を見ることができます。



## 31.4 システムctl

# システムctl

systemctl [オプション] コマンド [名前]

- **systemd** がコントロールする全てのもののステータスを表示する:

```
systemctl
```

- 利用可能なすべてのサービスを表示します:

```
$ systemctl list-units -t service --all
```

- アクティブなサービスのみを表示します:

```
systemctl list-units -t service
```

- 1台以上の**ユニット**をスタート（起動）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。

同様の方法で、**ユーザー・ユニット**も見ることができる:

```
systemctl list-units --user 'xdg*'
```

| ユニ                             | ロード  | アクティブサ     | 説明                                 |
|--------------------------------|------|------------|------------------------------------|
| xdg-desktop-portal-gtk.service | ロード中 | アクティブに動作する | Portal サービス (GTK+/GNOME 実装)        |
| xdg-desktop-portal.service     |      | アクティブに動作する | Portal サービスをロード                    |
| xdg-document-portal.service    |      |            | フラットパック ドキュメントポータルサービス             |
| xdg-permission-store.service   |      |            | サンドボックス化されたアプリのパーミッションストアをアクティブに実行 |

LOAD= ユニット定義が適切にロードされたかどうかを反映する。

ACTIVE= 上位ユニットの活性化状態、すなわちSUBの一般化。SUB

### ご注意

ユーザー・ユニットも多数存在するため、上記のコマンドでは、デスクトップXorgセッションでxdgで始まるユニットのみに出力を限定しているすべてのユニットファイルを表示するには

V.1.4 systemctl list-unit-files' を使う。

© Copyright the Linux Foundation 2023. 無断複写・転載を禁

じます。





## 31.5 ラボ



### ビデオ・デモンストレーション・リソース

[systemctl\\_demo.mp4](#)を使用しています。

### 練習 31.1: systemdを使って既存のサービスを変更する

**systemd** はビルトインサービスを完全に置き換えたり無効にしたりするだけでなく、**.override** ファイルや **.config** ディレクトリを経由して**ドロップイン**と呼ばれる既存のサービスの特定のパラメータだけを変更することもできます。この方法を使う利点として、パッケージのアップグレードによって変更が戻されることはありません。

最初のコンソール (tty1) でログイン (getty) を担当する**ユニット**のDescriptionを変更し、それを元に戻す。

ここでは、グラフィカル・インターフェースまたはssh経由でワークステーションにログインしていると仮定している。安全のため、このターミナルでログインしていないことを確認してください:

tty



### 非常に重要

このコマンドの結果が`/dev/tty1`と表示されたら、このラボはスキップした方がいいだろう。

1. まず、`getty@tty1.service`:

`systemctl status getty@tty1.service`

```
systemctl status getty@tty1.service
* getty@tty1.service - Getty on tty1
 ロード: ロード (/lib/systemd/system/getty@.service; enabled; vendor preset: enabled) アクテ
 イブ: アクティブ (実行中) 以降 Thu 2023-03-09 12:58:42 EST; 2ヶ月7日前
 ドキュメント: man:agetty(8)
 man:systemd-getty-generator(8)
 http://0pointer.de/blog/projects/serial-console.html
 メインPID: 621 (agetty) タス
 ク: 1 (リミット: 4656) メ
 モリ: 200.0K
 CPU: 2ミリ秒
 CGグループ /system.slice/system-getty.slice/getty@tty1.service
 621 /sbin/agetty -o -p -- \u --noclear tty1 linux
```

警告: パーミッションが不十分のため、一部のジャーナル・ファイルが開けませんでした。

2. 次に、サービスの説明を変更してみましょう。**Description=**の行をコピーして、違うことを書いてください。この例では、動的出力を**最初のターミナルに変更**しています。

`sudo systemctl edit getty@tty1.service`

/etc/systemd/system/getty@tty1.service.d/override.conf の編集

### ここから下のコメントまでの間が、ファイルの新しい内容になります。

[ユニット]

解説=第1ターミナルのゲッティ

### このコメント以下の行は破棄されます。

```
/lib/systemd/system/getty@.service
SPDX-License-Identifier: LGPL-2.1 もしくは
それ以降 #
このファイルは systemd の一
部です。##
systemd はフリーソフトウェアです。
フリーソフトウェア財団によって発行された ## GNU劣等一般公衆利用許
諾契約書(GNU Lesser General Public License)の条項の下 ## にあるものとす
る。
(任意で)それ以降のバージョンでも構いません。
```

3. 次に、`/etc/systemd/system/getty@tty1.service.d/override.conf` の内容を確認する。  
これは、今行った変更を表しているはずである。2行だけで、最後に改行はない。

```
$ cat /etc/systemd/system/getty@tty1.service.d/override.conf
```

```
[ユニット]
Description=最初の端末にゲッティを置く。
```

4. デーモンをリロードして変更を取り込み、結果をチェックする:

```
sudo systemctl daemon-reload
systemctl status getty@tty1.service
```

```
* getty@tty1.service - 最初の端末のゲッティ
 ロード: ロード (/lib/systemd/system/getty@.service; enabled; vendor preset: enabled)
 ドロップイン: /etc/systemd/system/getty@tty1.service.d
 オーバーライド.conf
 アクティブ: アクティブ (実行中) since Thu 2023-03-09 12:58:42 EST; 2 ヶ月 7 日前
 Docs: man:agetty(8)
 man:systemd-getty-generator(8)
 http://0pointer.de/blog/projects/serial-console.html
 メインPID: 621 (agetty) タス
 ク: 1 (リミット: 4656) メ
 モリ: 200.0K
 CPU: 2ミリ秒
 CGグループ /system.slice/system-getty.slice/getty@tty1.service
 621 /sbin/agetty -o -p -- \u --noclear tty1 linux
```

警告: パーミッションが不十分なため、一部のジャーナル・ファイルが開けませんでした。

5. 最後に、作成したファイルを削除して正常な状態に戻す。

```
sudo rm /etc/systemd/system/getty@tty1.service.d/override.conf
systemctl status getty@tty1.service
```

\* `getty@tty1.service` - Getty on tty1

ロード: ロード (`/lib/systemd/system/getty@.service`; enabled; vendor preset: enabled) アクテ

ィブ: アクティブ (実行中) 以降 Thu 2023-03-09 12:58:42 EST; 2ヶ月7日前

ドキュメント: `man:agetty(8)`

`man:systemd-getty-generator(8)`

<http://0pointer.de/blog/projects/serial-console.html>

メインPID: 621 (`agetty`) タス

ク: 1 (リミット: 4656) メ

モリ: 200.0K

CPU: 2ミリ秒

CGグループ `/system.slice/system-getty.slice/getty@tty1.service`

`621 /sbin/agetty -o -p -- \u --noclear tty1 linux`

警告: パーMISSIONが不十分なため、一部のジャーナル・ファイルが開けませんでした。



## 第32章

# バックアップとリカバリーの方法



|      |                                |     |
|------|--------------------------------|-----|
| 32.1 | バックアップの基本 .....                | 496 |
| 32.2 | バックアップとアーカイブ .....             | 498 |
| 32.3 | バックアップの方法と戦略 .....             | 500 |
| 32.4 | タール .....                      | 503 |
| 32.5 | 圧縮: gzip、bzip2、xzとバックアップ ..... | 506 |
| 32.6 | dd .....                       | 507 |
| 32.7 | rsync .....                    | 508 |
| 32.8 | バックアップ・プログラム .....             | 509 |
| 32.9 | 研究室 .....                      | 510 |

### 学習目標

このセッションを終えるまでに、あなたは次のことができるようになるはずだ:

- 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使用する。
- xzやgzipといった様々な圧縮方法を使い、tarアーカイブプログラムと効率的に組み合わせる。



### ご注意

\*\* これらのセクションは、一部または全部をオプションとすることができる。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。講師は、授業の経験や時間的制約に応じて、これらの項目を取り上げるか取り上げないかを選択することができます。





## 32.1 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
- 人は間違いを犯す
- 悪意ある人間が故意に損害を与える
- 不可解な出来事が起こる
- 巻き戻しは役に立つ

個人で管理するシステムが1台であろうと、多数のマシンで構成されるネットワークであろうと、システムのバックアップは非常に重要です。

オンディスク・データは重要な作業成果物であるため、保護したい商品である。失われたデータを再作成するには、時間とコストがかかる。データによっては一意であり、再作成する方法がない場合もあります。

ストレージメディアの信頼性が向上する一方で、ドライブ容量も増加している。バイトあたりの故障率が低下しても、予測不可能な故障は依然として発生している。故障したドライブとこれから故障するドライブの2種類しかないと言うのは悲観的かもしれないが、本質的には真実である。**RAID**を使うことは助けになるが、それでもバックアップは必要だ。

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もある。長く使われている安定したプログラムにも問題はあります。

誰もが一度や二度は隣の部屋から（あるいは自分の口から）**OOPS!**（あるいはもっとひどい言葉）を聞いたことがあるだろう。単なるタイプミスが、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る。

それは典型的な不満を抱えた従業員かもしれないし、主張したいことを持った外部のハッカーかもしれない。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る。

ファイルは、いつ、誰が、どのようにして消失したのか、まったく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へのリ

ストアが必要になることもあります。

## バックアップの必要性

- 間違いない：
  - ビジネス関連のデータ。
  - システム設定ファイル。
  - ユーザーファイル（通常は/home配下）。
- たぶんね：
  - 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などにある）。
- おそらく無理だろう：
  - 簡単に再インストールできるソフトウェア。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だ。
  - tmpディレクトリ
- 絶対に違う：
  - proc、/dev、/sys などの擬似ファイルシステム、スワップ

組織にとって不可欠なファイルのバックアップが必要なのは明らかです。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。

ロギング・ファイルは、システムの履歴を調査する必要がある場合に重要な意味を持つことがあり、侵入やその他のセキュリティ違反を検出するために特に重要です。

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、基本的に一時的なデータ（tmpディレクトリと同じ）なので、バックアップする必要はありません。

本

## 32.2 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、読めなくなるまでに有限の寿命があります。
- 従来の見積もり：
  - 磁気テープ：10～30年
  - CDとDVD：3～10年
  - ハードディスク：2～5年
- 生涯、非常に敏感である：
  - 環境条件（温度、湿度など）
  - メディアの質
  - 現在のオペレーティング・システムとハードウェア上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っていること
- バックアップには十分だが、永久的なデジタル・アーカイブには適さない。

通常のバックアップ・タイムスケールよりも長い寿命の場合、複数のコピーを使用してデータを保存し、時々新しいメディアにコピーすることができる。

非常に長い時間（つまり、何十年、何世紀など）、標準的な方法は、すべてが陳腐化する可能性があるため、簡単には機能しない：

- 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

どの安価なデジタルフォーマットも、（ワインのように）適切に保管され、手入れされ続ければ、紙やフィルムに長期間対抗することはできない。

これは深刻な問題であり、すべてが失われる前に良い解決策があるはずだ。

ブ

## テープドライブ

- 比較的遅い
- 順次アクセスのみを許可する
- かつてほど一般的ではない
- 現在、プライマリー・バックアップに使用されることはほとんどない
- オフサイトでの保管やアーカイブに役立つこともある。

テープドライブは以前ほど一般的ではありません。比較的低速で、シーケンシャル・アクセスしかできない。最新のセットアップでは、プライマリー・バックアップに使われることはほとんどない。オフサイト・ストレージとして、長期参照用のアーカイブ目的で使用されることはある。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たない。

現代のテープドライブは通常、**LTO**（リニア・テープ・オープン）タイプで、その最初のバージョンはオープンスタンダードの代替品として1990年代後半に登場した。初期のバージョンは100GBまで、新しいバージョンでは同じサイズのカートリッジに2.5TB以上。

日々のバックアップは通常、**NAS**（ネットワーク・アタッチド・ストレージ）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導入は魅力的ではなくなってきている。しかし、テープ・ベースのバックアップはまだ存在し、システム管理者はそれらに対処する必要があるかもしれない。

以下では、バックアップメディアの特定の物理的な形態に焦点を当てないようにし、より抽象的に話を進めていく。

### 32.3 バックアップの方法と戦略

## バックアップ方法

- **フルだ:**

システム上のすべてのファイルをバックアップする。

- **インクリメンタルだ:**

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されたすべてのファイルをバックアップする。

- **ディファレンシャル:**

前回のフルバックアップ以降に変更されたすべてのファイルをバックアップする。

- **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、同じレベルまたは以前のレベルでバックアップする。

- **ユーザー**

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併用する。

すべてのバックアップを、保護対象のシステムと同じ物理的な場所に置いてはならない。そうでなければ、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る。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、インターネット経由でバックアップ・ファイルを別の物理的な場所に転送することが一般的になっている。もちろん、これは暗号化やその他のセキュリティ対策を適切に用いて安全な方法で行われなければならない。



## バックアップ戦略

テープを使った有用な戦略のひとつである（説明の中で、他のメディアで代用することも簡単にできる）：

1. テープ1を金曜日のフルバックアップに使用する。
2. 月曜日から木曜日の増分バックアップには、テープ2～5を使用します。
3. 第2金曜日のフルバックアップにはテープ6を使用する。
4. テープ2-5を第2月曜日から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へのフルバックアップ後、ディザスタリカバリ用にテープ1を外部に移動する。
7. 次回のフルバックアップ（来週の金曜日）では、テープ1とテープ6を交換する。

バックアップ方法は、関連するリストア方法がなければ意味がないことに注意しなければならない。ストラテジーを選択する際には、双方の堅牢性、明確性、容易性を考慮しなければならない。

目安としては、少なくとも2週間分のバックアップを用意しておくことだ。

最も単純なバックアップ方式は、すべてのバックアップを一度フルバックアップし、その後変更されたすべてのバックアップを増分バックアップすることです。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、両方をミックスして使用することができる。

## バックアップ関連ユーティリティ

- **cpio**
- **タール**
- **gzip、bzip2、xz**
- **dd**
- **同期**
- **ダンプ/リストア**
- **鈹**

**cpio**と**tar**は、ファイルのアーカイブを作成・抽出する。

アーカイブはしばしば**gzip**、**bzip2**、または**xz**で圧縮される。アーカイブファイルは、ディスク、磁気テープ、またはファイルを格納できるその他のデバイスに書き込むことができる。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システムへ ファイルを転送するのに非常に便利である。

**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ある。パーティション全体やディスク全体をコピーするのに使うことができる。

**rsync** は強力なユーティリティで、ディレクトリのサブツリーやファイルシステム全体をネットワーク越しに、あるいはローカルマシン上の異なるファイルシステムのロケーション間で同期することができる。

**dump**と**restore**は、バックアップ用に特別に設計された古いユーティリティである。ファイルシステムから直接読み込む（その方が効率的）。ただし、リストア元と同じファイルシステム・タイプにのみリストアする必要がある。より新しい代替手段がある。

**mt**は、バックアップとリストアを実行する前に、テープを照会し、位置決めするために使用される。

## 32.4 タール

# バックアップにtarを使う

- アーカイブの作成: `-c`を使うか、単に`c`を使う。

```
tar cvf /dev/st0 /ルート
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、短縮形（1文字に`-`を付けたもの）と長形式（`--`を付けたもの）があることに注意。
- **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とき、引数として与えられた各ディレクトリの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る。リストアップするときには、必要に応じてディレクトリを再構成する。

さらに`--newer`オプションで増分バックアップもできる。

**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らないバックアップも扱うことができる。`f`または`-file`オプションでデバイスまたはファイルを指定できる。

バックアップを作成した後、検証オプションを使って、バックアップが完全で正確であることを確認することができる。デフォルトでは、**tar**はアーカイブ内の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とき、**tar**は絶対パス名から先頭のスラッシュを取り除くというメッセージを表示する。これによって

ファイルをどこにでもリストアできるようになるが、デフォルトの動作は変更することができる。

ほとんどの**tar**オプションは、ダッシュ1つの短い形式か、ダッシュ2つの長い形式で指定できる。cは-createと完全に等価である。また、ダッシュをいちいち入力する必要があるように、（短い表記法を使う場合）オプションを組み合わせることができることに注意すること。

例えば、`tar cvf file.tar dir1`は`tar -cvf file.tar dir1`と同じ結果になる。

## ファイルの復元にtarを使う

- アーカイブから取り出す: `-x` または `--extract` を使う。

```
tar --extract --same-permissions --verbose --file /dev/st0
tar -xpvf /dev/st0
tar xpvf /dev/st0
```

- リストアする特定のファイルに名前を付けることができます。

```
tar -xvf /dev/st0 somefile
```

- tarバックアップの内容をリストアップする

```
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する。特定のファイルだけを指定して、ファイル抽出リストを絞り込むこともできる。ディレクトリを指定した場合は、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る。

`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 `--list` 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、抽出は行いません。

## tarによる増分バックアップ

- newerまたは--after-dateオプションを使用する。
- 日付ベース
- 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、2011年12月1日以降に変更された/var/tmp内の全ファイルのバックアップ・アーカイブが作成される。

N（または同等の--newer）または--after-dateオプションを使用すると、**tar**でインクリメンタルバックアップを行うことができる。どちらのオプションでも、日付か修飾（参照）ファイル名のどちらかを指定する必要がある。

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は考慮されない。これらの変更を含む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する。

**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わなければならない。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ユーティリティで起こることがある。

## 32.5 圧縮: gzip、bzip2、xzとバックアップ

### アーカイブ圧縮方法

- 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転送時間を節約できます。
- 圧縮効率を上げるためには、圧縮時間が長くなる:
  - **gzip**: 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、レガシーな.lzma形式もサポートする。
- 最近のマシンは、**圧縮→送信→解凍**のサイクルが、単に非圧縮ファイルを送信（またはコピー）するよりも速いことが多い。

- <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われていない。

カーネル

圧縮ユーティリティは、**tarと組み合わせ**て非常に簡単に（そして頻繁に）使用できる:

```
tar zcvf source.tar.gz source
tar jcvf source.tar.bz2 source
tar Jcvf source.tar.xz source
```

圧縮アーカイブを作成する。最初のコマンドは、次のコマンドとまったく同じ効果を持つ:

```
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、より効率的である。

減圧用:

```
tar xzvf source.tar.gz
tar xjvf source.tar.bz2
tar xJvf source.tar.xz
```

あるいはもっと単純なものだ:

```
tar xvf source.tar.gz
```

最近の**tar**のバージョンは、圧縮方法を感じて自動的に処理してくれるからだ。

もちろん、画像や.pdfファイルなど、コンポーネント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う価値はない。



## 32.6 dd

# dd

- 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 オプション`  
`dd if=/dev/sda of=/dev/sdb`
- ファイル作成に使用可能  
`dd if=/dev/zero of=file1 bs=1M count=10`
- ハードディスク全体またはパーティションをバックアップ可能
- CDやDVDをコピーできる

ddは一般的なUNIXベースのプログラムで、生データの低レベルコピーと変換を主な目的としている。指定したバイト数またはブロックをコピーし、その場でバイトオーダーを変換したり、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われる。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`のような特別なファイルから固定量のデータを読み込んだりするのも使われる。

ハードドライブ全体を別のハードドライブ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する。

```
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
```

## 32.7 同期

# バックアップにrsyncを使う

- 使い方:

`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間:

`$ rsync file.tar someone@backup.mydomain:/usr/local`

- dry-runで実行する前にテストすること:

`rsync -r --dry-run /usr/local /BACKUP/usr`

**rsync**（リモート同期）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る。転送元と転送先は`target:path`の形式で指定し、`target`は`[user@]host`の形式で指定する。`user@]`ホストは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る。したがって、これらはすべて可能な**rsync**コマンドである:

```
$ 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（特に`--delete`オプションを使う場合）には細心の注意を払わなければならないので、最初に`--dry-run`オプションを使い、予測される動作が正しそうなコマンドを繰り返すことを強く推奨する。

**rsync**は非常に賢い。ローカルファイルとリモートファイルを小分けにしてチェックし、あるディレクトリを似たようなディレクトリにコピーする際、差分だけが最初のディレクトリと一緒にネットワーク上にコピーされるという点で非常に効率的である。`r`オプションを使うと、**rsync**はディレクトリツリーを再帰的にたどり、ソースファイルとしてリストされているディレクトリ以下のすべてのファイルとディレクトリをコピーする。したがって、プロジェクト・ディレクトリをバックアップする非常に便利な方法は、次のようなものである:

```
rsync -r project-X archive-machine:archives/project-X
```

単純な（そして非常に効果的で高速な）バックアップ戦略は、ネットワーク上でディレクトリやパーティションを**rsync**コマンドを頻繁に実行すること。

## 32.8 バックアップ・プログラム

# バックアップ・プログラム

- アマندا
- バクラ
- クロネジラ

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、**Linux**で利用可能なバックアップ・プログラム・スイートには事欠きません。特によく知られているのは、以下のようなものである：

- **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) を使用するが、はるかに堅牢でコントロールしやすい。

**Amanda** は一般的に Enterprise **Linux** システム上で、通常のリポジトリから入手可能です。完全な情報は <http://www.amanda.org> にあります。

- **Bacula**は異種ネットワークの自動バックアップ用に設計されている。使い方はかなり複雑で、(作者によって) 経験豊富な管理者だけに推奨されている。

**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](https://www.bacula.org/7.0.x-manuals/en/main/Main%20Reference.html) にある。

- **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、**ゴースト**に使用したり、多くのマシンをインストールするためのイメージを提供したりすることができます。完全な情報は<https://clonezilla.org>。

このプログラムには2つのバージョンがある：**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。

**Clonezilla SE**はサーバー版で、同時に多くのコンピューターにクローンを作成できる。

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、ブートローダーをサポートしている。

## 32.9 ラボ

### 🔪練習問題 32.1: 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をincludeを最上位ディレクトリとして圧縮したtarアーカイブを置きます。圧縮方法は何でもよい（gzip、bzip2、xz）。
2. アーカイブ内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进行比较する。

### ✅解答 32.1

1. 

```
mkdir /tmp/backup
cd /usr ; tar zcvf /tmp/backup/include.tar.gz include
cd /usr ; tar jcvf /tmp/backup/include.tar.bz2 include
$ cd /usr ; tar Jcvf /tmp/backup/include.tar.xz include
```

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つの方法の圧縮効果に注目してほしい:

```
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 include/
-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
.....
```

解凍時にj、J、zオプションを指定する必要はないことに注意。tarは賢いので、何が必要かを判断してくれる。

4. 

```
cd ... ; mkdir restore ; cd restore
tar xvf ../backup/include.tar.bz2
```

```
include/unistd.h
include/re_comp.h
include/regex.h
include/link
.....
diff -qr include /usr/include
```

## 練習 32.2: バックアップにrsyncを使う

1. **rsync**を使って、バックアップ・ディレクトリに`/usr/include`の完全なコピーを再び作成します:

```
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. 回目のコマンドを実行し、何かが起こるかどうかに見てみよう:

```
rsync -av /usr/include .
```

```
インクリメンタルファイルリストの送信

送信 127398 バイト 受信 188 バイト 255172.00 バイト/秒
合計サイズ 41239979 スピードアップ 323.23
```

3. **rsync**で混乱することのひとつは、正しいコマンドを期待してしまうことだ:

```
rsync -av /usr/include include
```

```
インクリメンタルファイルリストの送信
...
```

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に気づくだろう!

4. 余分なファイルを削除するには、`--delete`オプションを使う:

```
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 の削除
include/include/xen/evchn.h の削除
....
include/include/FlexLexer.h の削除
include/include/ の削除
```

```
送信 127401 バイト 受信 191 バイト 85061.33 バイト/秒 合
計サイズ 41239979 スピードアップ 323.22
```

5. もう1つ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・ツリーを削除し、`-dry-run`オプションの有無にかかわらず**rsync**を再度実行してみよう:

```
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/
インクルード/xen/イ
ンクルード
/xen/evtchn.h
```





```
include/xen/privcmd.h
```

```
送信 127412 バイト 受信 202 バイト 255228.00 バイト/秒 合
```

```
計サイズ 41239979 高速化 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・セットを備えたシンプルなスクリプト:

SH

### rsyncを使用したスクリプト

```
#!/bin/sh
```

```
set -x
```

```
rsync --progress -avrxH --delete $*.
```

これは、ネットワーク上だけでなく、ローカルマシン上でも動作する。**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

もし複数のコンピューターにアクセスできるのであれば、ソースとデスティネーションをそれぞれ別のコンピューターに置いて、この手順を試してみしてほしい。

## 第33章

# Linuxセキュリティ・モジュール



|      |                        |     |
|------|------------------------|-----|
| 33.1 | Linuxセキュリティ・モジュール..... | 514 |
| 33.2 | SELinux.....           | 516 |
| 33.3 | AppArmor.....          | 531 |
| 33.4 | 研究室.....               | 537 |

### 学習目標

このセッションを終えるまでに、あなたは次のことができるようになるはずだ：

- Linux セキュリティモジュールのフレームワークがどのように機能し、どのように展開されるかを理解する。
- 利用可能なさまざまなLSM実装をリストアップします。
- SELinuxの主な機能を説明する。
- 利用可能なさまざまなモードと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。
- AppArmorに慣れる。

513



## 33.1 Linuxセキュリティ・モジュール

# Linuxセキュリティ・モジュールとは？

- **LSM**（Linuxセキュリティ・モジュール）は、カーネルへのリクエストに対する**強制的なアクセス制御**を実装している。
- **LSMの使用**：
  - カーネルへの変更を最小限に抑える。
  - カーネルのオーバーヘッドを最小限に抑える。
  - さまざまな実装を柔軟に選択でき、それぞれが自己完結型の**LSM**（Linux Security Module）として提供される。

現代のコンピューター・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ー数、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・保守者の両方にある。もちろん、ユーザも適切な手続きに従わなければならないが、適切に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侵害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セキュリティ・モジュール・フレームワーク、特に**SELinux**のデプロイメントを使用することで、どのようにセキュリティを強化するかについて説明します。

基本的な考え方は、**システムコールをフックすること**である。強化された能力を必要とする作業を達成するために、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する。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の主な選択肢

- 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LSMの使用

2019年以降、特定の順序で**LSM**を組み合わせる（スタックする）ことが可能になった。

- 参照 <https://www.starlab.io/blog/a-brief-tour-of-linux-security-modules>  
メカニズムと選択肢についての素晴らしいレビューがある。

長い間、実装されていたセキュリティ強化モデルは**SELinux**だけだった。このプロジェクトが2001年に初めてカーネルに直接含まれるよう上流に持ち上がったとき、セキュリティ強化に1つのアプローチしか使わないことに異論があった。

その結果、**SELinux**の代替モジュールが開発されるたびに使用できる**LSM**アプローチが採用され、2003年に**Linux**カーネルに組み込まれた。

もともとは、**Linux**カーネルの同じ部分を変更する可能性があるため、一度に使用できる**LSM**は1つだけだった。

しかし現在では、多少の注意を払えば、それらを積み重ねることが可能になっている。LSMの組み合わせを設定する際、メジャーかマイナーかを考慮するようになった。

使用量の多い順に、主に**SELinux**、次に**AppArmor**を取り上げる。

## 33.2 セリナックス

### SELinuxの概要

- NSAが開発
- 追加の保護方法
- 用途
  - 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ます。
- その他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\\_users\\_and\\_administrators\\_guide/index](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/selinux_users_and_administrators_guide/index)

SELinuxは元々、米国NSA（国家安全保障局）によって開発されたもので、RHELに不可欠なものである。

は非常に長い間、多くの利用者を獲得してきた。

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つの概念的な量を扱う：

1. **コンテキスト**：ファイル、プロセス、ポートに対するラベル。コンテキストの例としては、SELinuxのユーザー、ロール、タイプがある。
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**：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するルールのセットである。



**SELinux**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ルールで使用されるラベルです。デフォルトのポリシーはいかなるアクセスも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るアクションを記述するために使用される。



ス

## SELinuxの強制モード I

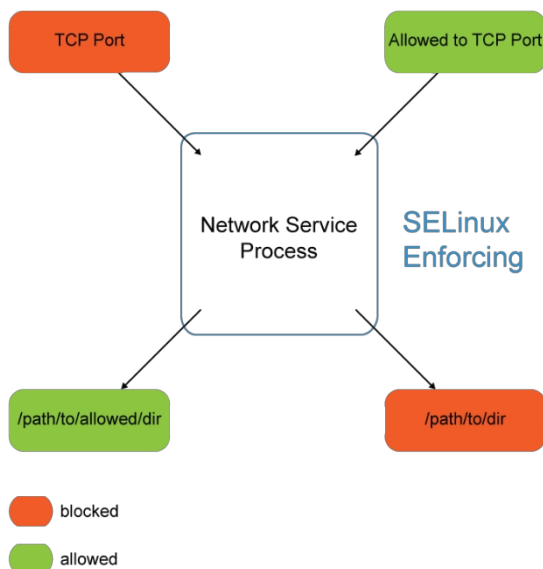


図33.1: SELinux強制モード

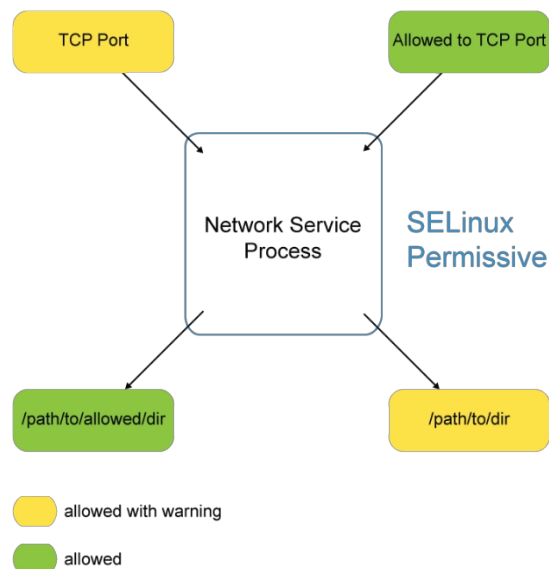


図33.2: SELinux許可モード

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、その場所はディストリビューションによって異なる（多くの場合、`/etc/sysconfig/selinux` にあるか、そこからリンクされている）。このファイルはよく自己文書化されている。

## SELinuxの強制モード II

- **SELinuxの3つのモード**

- 施行
- 許可制
- 無効

- `etc/sysconfig/selinux` (**CentOS**および**openSUSE**) で設定するか、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。

- **強制する**: すべての**SELinux**コードが作動し、ポリシーに従ってアクセスが拒否される。すべての違反は監査され、ログに記録される。
- **許可する**: **SELinux**コードを有効にするが、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う。
- **無効**: **SELinux**カーネルとアプリケーションコードを完全に無効にし、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。

ス

## getenforceおよびsetenforce

- 現在のモードを確認または設定する：

- ゲーテンフォース

```
ゲットエンフォース
```

施行

- セテンフォース

```
sudo setenforce Permissive
```

```
ゲットエンフォース
```

許可制

**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る。しかし、この方法では、無効モードへの切り替えや無効モードからの切り替えはできない。

setenforceでは、許可モードと強制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。を完全に無効にする。**SELinux**を無効にするには、少なくとも2種類の方法がある：

- **設定ファイル**：**SELinux**設定ファイル（通常は`/etc/selinux/config`）を編集し、`SELINUX=disabled`に設定する。これはデフォルトの方法であり、**SELinux**を恒久的に無効にするために使うべきである。
- **カーネル・パラメーター**：再起動時のカーネル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**モードを使うことが望ましい。

## 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制限的にする**デフォルト**のポリシー。ユーザープロセスと**initプロセス**はターゲットにされないが、ネットワークサービスプロセスはターゲットにされる。**SELinuxは全てのプロセス**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減らします。

- **最低限だ:**

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:**

マルチレベル・セキュリティ・ポリシーは、より制限的である。すべてのプロセスは、特定のポリシーを持つきめ細かいセキュリティ・ドメインに配置される。

モードを設定する同じ設定ファイル（通常は`/etc/sysconfig/selinux`）は、**SELinuxポリシー**も設定する。複数のポリシーを設定できますが、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、システムのリブートと、時間のかかるファイルシステムの内容の再ラベル化が必要になる場合があります。各ポリシーには、`/etc/selinux/[SELINUXTYPE]`の下にインストールしなければならないファイルがあります。

ス

## 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
- コンテキストを検索または変更する：
  - コンテキストを表示するには **-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**を使う  
`$ chcon -t etc_t somefile`  
`chcon --reference somefile someotherfile`

前述したように、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。これらのラベルは、アクセスルールを記述するために使用される。**SELinux**のコンテキストには、**User**、**Role**、**Type**、**Level**の4つがある。

ここでは、最もよく使われるコンテキストである**type**に注目する。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で終わることになっている。

ls -Z

```
-ドッグドッグ アンコンファインド_u:object_r:user_home_t:s0 サムファイル
```

\$ chcon -t etc\_t somefile

ls -Z

```
-ドッグドッグ アンコンファインド_u:オブジェクト_r:etc_t:s0 ファイル
```

ls -Z

```
-ドッグドッグ アンコンファインド_u:オブジェクト_r:etc_t:s0 ファイル
-ドッグドッグ アンコンファインド_u:object_r:user_home_t:s0 somefile1
```

\$ chcon --reference somefile somefile1

ls -Z

```
-ドッグドッグ アンコンファインド_u:オブジェクト_r:etc_t:s0 ファイル
-ドッグドッグ アンコンファインド_u:オブジェクト_r:etc_t:s0 ファイル1
```



ス

## SELinuxと標準コマンド

- 多くの基本ユーティリティ（ls、ps、cpなど）が**SELinux**で動作するように拡張されている。
- 通常、Zオプションを追加する：  
`$ ls -Z ...`  
`ps axZ`
- **SELinux**が無効の場合、拡張オプションは何もしない。

lsやpsのような多くの標準コマンドラインコマンドは、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た。標準的なコマンドラインツールには、次のようにパラメータZが渡されることが多い:

ps axZ

```
label 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] ?
...
sshd: jimih@pts/0:未定義_u:未定義_r:未定義_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 ?S 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/

```
/ホーム/:
jimih jimih unfinid_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 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がある。

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有用な情報は表示されないことに注意。

## コンテキストの継承

- 新規ファイルは親ディレクトリのコンテキストを継承
- ファイル移動の際、移動元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ていない場合、問題が発生する可能性がある

ス

新規に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継承しますが、ファイルの移動やコピーの際には、移動元ディレクトリのコンテキストが保存されるため、問題が発生する可能性があります。

先ほどの例に引き続き、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なかったことがわかる：

```
cd /tmp/
touch tmpfile
ls -Z tmpfile
```

```
-jimih jimih unfinid u:object r:user tmp t:s0 tmpfile
```

```
cd
touch homefile
ls -Z homefile
```

```
-jimih jimih unconfined u:object r:user home t:s0 homefile
```

```
mv /tmp/tmpfile .
ls -Z
```

```
-jimih jimih unconfined_u:object_r:user_home_t:s0 homefile
-jimih jimih unfinid u:object r:user tmp t:s0 tmpfile
```

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ある。**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、それを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`サーバーはそのファイルにアクセスできなくなります。

# リストコン

- 親ディレクトリに合わせて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いて、ファイルコンテキストをリセットします。

以下の例では、**restorecon**はホーム・ディレクトリにあるすべてのファイルのデフォルト・ラベルを再帰的にリセットしている：

```
ls -Z
```

```
-jimih jimih unconfined_u:object_r:user_home_t:s0 homefile
-jimih jimih unfined_u:object_r:user_tmp_t:s0 tmpfile
```

```
$ restorecon -Rv /home/jimih
```

```
restorecon reset /home/jimih/tmpfile context | コンテキストを元に戻す
アンコンファインド_u:object_r:user_tmp_t:s0->アンコンファインド_u:object_r:user_home_t:s0
```

```
ls -Z
```

```
-jimih jimih unconfined_u:object_r:user_home_t:s0 homefile
-jimih jimih unfined_u:object_r:user_home_t:s0 tmpfile
```

**tmpfile**のコンテキストが、ホーム・ディレクトリー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。型が **user\_tmp\_t** から **user\_home\_t** に変更された。

ス

## セマネージ

- 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る。

**semanage fcontext** (**policycoreutils-python** パッケージによって提供される)は、ファイルとディレクトリのデフォルトコンテキストを変更し、表示することができます。**semanage fcontext** はデフォルト設定を変更するだけで、既存のオブジェクトには適用しないことに注意してください。そのため、**restorecon** を後で呼び出す必要があります。例えば

```
[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 \
 アンファインド_u:object_r:default_t:s0->システム_u:object_r:httpd_sys_content_t:s0
```

```
[root@rhel7 /]# ls -Z
```

```
drwxr-xr-x. root system_u:object_r:httpd_sys_content_t:s0 virtualHosts
```

したがって、**default\_t**から**httpd\_sys\_content\_t**へのコンテキストの変更は、**restorecon**の呼び出し後にのみ適用される。

ス

## SELinuxブール値の使用

- SELinuxポリシーの動作を変更する
- 有効にも無効にもできる
  - **getsebool**: ブーリアンを見る
  - **setsebool**: ブーリアンを設定する
  - **semanage boolean -l**: 永続的なブーリアン設定を見る

```
$ 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
```

しつこくない注。

```
setsebool -P allow_ftpd_anon_write on
$ semanage boolean -l | grep allow_ftpd_anon_write
 allow_ftpd_anon_write -> オン
```

今はしつこい。

```
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
mindluna_read_generic_user_content (off , off) Allow mindluna to read generic user content
spanassassin_can_network (off , off) Allow spanassassin to can network
gluster_anon_write (off , off) Allow gluster to anon write
deny_ptrace (off , off) Allow deny to ptrace
selinuxuser_execmod (on , on) Allow selinuxuser to execmod
httpd_can_network_relay (off , off) Allow httpd to can network relay
openvpn_enable_homedirs (on , on) Allow openvpn to enable homedirs
glance_use_execmod (off , off) Allow glance to use execmod
telepathy_tcp_connect_generic_network_ports (on , on) Allow telepathy to tcp connect generic network ports
httpd_can_connect_mythtv (off , off) Allow httpd to can connect mythtv
....
student@CentOS7:tmp$
```

図 33.3: semanage

## SELinuxアクセス監視

- SELinuxエラーを追跡するには**setroubleshoot-server**を使用します。
  - インストール後、**auditd**サービスを再起動する。
- 生のメッセージを/var/log/audit/audit.logに保存する。
- メッセージを /var/log/messages に移動する。
- 詳細なメッセージを見るには**sealert**を使用

SELinuxには、実行時に問題を収集し、その問題をログに記録し、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**パッケージによって提供される。以下に使用例を示す：



```
[root@rhel7 ~]# echo 'File created at /root' > 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) を解決しています... ::1,
127.0.0.1 localhost (localhost)::1:80 に接続していま
す... 接続されました。HTTPリクエストが送信され
ました。403 Forbidden 2014-11-21 13:42:04 ERROR 403
: 禁止されています。
```

```
[root@rhel7 ~]# tail /var/log/messages
11/21 13:42:04 rhel7 setroubleshoot: プラグイン例外restorecon
11/21 13:42:04 rhel7 setroubleshoot: SELinux が /usr/sbin/httpd の getattr アクセスを妨げています。 完全な
SELinux 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 が /usr/sbin/httpd が .

とりあえず、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

```
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 [file]。
ソース httpd
ソースパス /usr/sbin/httpd

```



### 33.3 AppArmor

## AppArmor

- 強制アクセス制御（**MAC**）を提供する。
- 管理者は、プログラムの機能を制限するセキュリティ・プロファイル  
をプログラムに関連付けることができる。
- **SELinux**よりも使いやすいと考えられている。
- ファイルシステム・ニュートラル（セキュリティ・ラベルは不要）と考えられている。

AppArmorは、**SELinux**に代わる**LSM**である。2006年から**Linux**カーネルに組み込まれている。**SUSE**、**Ubuntu**、その他のディストリビューションで採用されている。

**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**）モデルを補完する。

手動でプロファイルを指定するだけでなく、AppArmorには学習モードがあり、プロファイル違反は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いて、プロファイルに変換することができます。

## ステータス確認

- **AppArmor**は通常、デフォルトで有効化され、ロードされる。
- **Linux**カーネルがコンパイルされ、オンになっていること。
- 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かロードされないかの原因:  
`sudo systemctl [enable|disable] apparmor`
- 現状の詳細  
`sudo apparmor_status`
- プロファイルとプロセスを強制モードまたは苦情モードのいずれかにする  
- **SELinux**の強制モードと許容モードに直接似ている。

**AppArmorが付属している**ディストリビューションは、デフォルトでこれを有効にし、ロードする傾向がある。**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、一度に実行できる**LSM**は1つだけであることに注意。

**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では、以下のことができる：

```
sudo systemctl [start|stop|restart|status] apparmor
```

現在の動作状態を変更したり、問い合わせたりする：

```
sudo systemctl [enable|disable] apparmor
```

ブート時にロードするかどうかを設定す

る。現在のステータスを見るには

```
sudo apparmor_status
```

```
apparmorモジュールがロードされる。
25のプロファイルがロードされる。
25のプロファイルがエンフォースメント・モードになっている。
/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**complaint**モードになっている

。プロセスリストではPIDが与えられていることに注意：

```
ps aux | grep libvirtd
```

```
ルート 787 0.0 0.9 527200 35936 ? Ssl 10:54 0:00 /usr/sbin/libvirtd
学生 3346 0.0 0.0 13696 2204 ポイント/16 S+ 11:42 0:00 grep --color=auto libvirtd
```

## モードとプロファイル

- モード：
  - **Enforce Mode** 制限されたアクションが阻止され、報告される。  
**aa-enforce**で設定
  - **苦情** ポリシーは実施されないが、ポリシー違反は報告される。  
**aa-complain**とセット
- **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る:

・ 強制モード

アプリケーションは、制限された動作をすることができない。違反が試みられると、システムのログファイルに報告される。これはデフォルトのモードである。**aa-enforce**でプロファイルをこのモードに設定できる。

・ 苦情

ポリシーは強制されないが、ポリシー違反の試みは報告される。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**

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化されています。

これらのプロファイルは、`/etc/apparmor.d`に**保存されます**。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場合、パッケージ内の実

行可能ファイルに固有の新しいプロファイルを作成できます。

どの**AppArmor**プロファイルがシステムにインストールされるかは、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**システムでは

`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何を書けるかについての完全なドキュメントは、`man apparmor.d`で入手できる。

# AppArmor ユーティリティ

表33.1: AppArmorユーティリティ

| プログラム      | 用途                                  |
|------------|-------------------------------------|
| アーマー・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 |
| アラーム通知     | AppArmorログメッセージのサマリーを表示する           |

|         |                                                                                       |
|---------|---------------------------------------------------------------------------------------|
| 文句を言う   | 指定したプロファイルを苦情モードに設定する。                                                                |
| 強制する    | 指定したプロファイルを強制モードに設定する。                                                                |
| 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、システムにロードされないようにする。<br>を立ち上げた。                             |
| 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は、既存のプロファイルを考慮する方法を提案し、承認されれば修正して再ロードする。 |
| イージープロフ | プログラムの基本的な <b>AppArmor</b> プロファイルの設定を支援します。                                           |



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用意されている。例えば、**openSUSE** システムの場合：

```
rpm -qil apparmor-utils | grep bin
```

```
/usr/bin/aa-easyprof
/usr/sbin/aa-audit
/usr/sbin/aa-autodep
/usr/sbin/aa-cleanprof
/usr/sbin/aa-complain
/usr/sbin/aa-デコード
/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せることに注意：

```
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 root 11 Nov 11 13:02 /usr/sbin/complain -> aa-complain*
linux-llgn:/etc/apparmor.d #.
```



## 33.4 ラボ

### 🔗練習問題 33.1: SELinux: コンテキスト



#### ご注意

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）でのみ実行できる。**Ubuntu**のような**Debian**ベースのディストリビューションにインストールすることは可能ですが、最も簡単な作業ではありませんし、あまり行われていません。

1. **getenforce**と**sestatus**を実行して、**SELinux**が有効になっていて、強制モードになっていることを確認する。そうではない場合は、`/etc/selinux/ config`を編集して再起動し、再度確認してください。
2. **Apache**ウェブ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、それが動作していることを確認する:

```
sudo dnf install httpd
sudo systemctl start httpd
elinks http://localhost
```

(これ以降の手順では、ブラウザとしてlynxやelinksなどを使うこともできるし、Firefoxや**chrome**などのグラフィカル・ブラウザを使うこともできる)

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する:

```
sudo sh -c "echo file1 > /var/www/html/file1.html"
```

4. 見えるかどうか確認してください:

```
$ elinks -dump http://localhost/file1.html
```

```
フ
ア
イ
ル1
```

次に、rootのホーム・ディレクトリーに小さなファイルをもう一つ作り、`/var/www/html/`に移動します（コピーしないで移動してください! ）。(コピーしないで、移動してください! ) それから、それを表示してみてください:

```
sudo cd /root
sudo sh -c "echo file2 > file2.html"
sudo mv file2.html /var/www/html
$ elinks -dump http://localhost/file2.html
```

禁止

このサーバー上の/file2.htmlにアクセスする権限がありません。

5. セキュリティコンテキストを調べる:

```
cd /var/www/html
```

```
ls -Z file*.html
```

```
-root unconfined_u:object_r:httpd_sys_content_t:s0 file1.html
-root unfinid_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、再度表示する：

```
sudo chcon -t httpd_sys_content_t file2.html
$ elinks http://localhost/file2.html
```

フ

ア

イ

ル2

## 練習33.2: apparmorのセキュリティを探る



### ご注意

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**でテストしたものだが、**openSUSE**など他のAppArmor対応システムでも動作するはずである。

**Ubuntu**では、`/bin/ping`ユーティリティがSUIDを有効にして実行される。この演習では、**ping**を**ping-x**にコピーし、プログラムが機能するように機能を調整する。

その後、**AppArmor**プロファイルを構築し、インストールして、何も変更されてい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、プログラムの機能が向上します。

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する：

```
student@ubuntu:~$ sudo apt install apparmor*
```

2. **ping**のコピー(**ping-x**と呼ぶ)を作成し、それが初期状態で特別な権限や機能を持っていないことを確認する。さらに、通常のユーザーである**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:~$
```

```
student@ubuntu:~$ ping-x -c3 -4 127.0.0.1
```

```
ping: socket: 操作が許可されていません
```

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。
```

```
127.0.0.1からの64バイト: icmp_seq=1 ttl=64 time=0.092 ms
```

```
127.0.0.1からの64バイト: icmp_seq=2 ttl=64 time=0.093 ms
```

```
127.0.0.1からの64バイト: icmp_seq=3 ttl=64 time=0.086 ms
```

```
--- 127.0.0.1のping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 = 0.086/0.090/0.093/0.008ms
```

修正した**ping-x**プログラムは正常に機能するようになった。

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**からの出力は長いので、興味深い行を**grep**で探すことができる:

```
student@ubuntu:~$ sudo aa-status | grep -e "^[[:alnum:]]"-e ping
```

apparmorモジュールがロードされる。

87のプロファイルがロードされる。

51 のプロファイルが強制モード

になっている。

36人のプロフィールが文句を言うモード になっている。

17のプロセスにプロファイルが定義されている。

6 プロセスがエンフォースメントモードである。

11プロセスが不満モード。

0プロセスはアンファインドだが、プロファイルは定義されている。

**ping**のプロファイルがロードされ、実行が有効になっていることがわかる。

5. 次に**ping-x**用の新しいプロファイルを作成する。このステップには2つのターミナル・ウィンドウが必要だ。

最初のウィンドウ (**window1**) では、**aa-genprof**コマンドを実行します。これは、以下をスキャンして**AppArmor**プロファイルを生成する。

**/var/log/syslog**に**AppArmor**のエラーを記録する。

2つ目のウィンドウ(**window2**)は**ping-x**を実行するのに使われる(追加情報は**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ました。

**bin/ping-x**を苦情モードに設定する。

開始する前に、あなたが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。詳細については、以下のウィキページを参照してください:

<http://wiki.apparmor.net/index.php/Profiles>

プロファイリング対象のアプリケーションを別のウィンドウで起動し、今すぐその機能を実行してください。

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イベントごとに、アクセスを許可するか拒否するかを選択する機会が与えられます。

プロファイリング: **/bin/ping-x**

AppArmorイベントのシステムログを取得する]/ [終了する

**ウィンドウ2:**

```
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。

127.0.0.1からの64バイト: icmp\_seq=1 ttl=64 time=0.099 ms

127.0.0.1からの64バイト: icmp\_seq=2 ttl=64 time=0.120 ms

127.0.0.1からの64バイト: icmp\_seq=3 ttl=64 time=0.114 ms

--- 127.0.0.1のping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 = 0.099/0.111/0.120/0.008ms

**window1**にある:

ping-xコマンドが完了したら、今度は`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しなければならない。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**が必要になるかもしれない。スキャンするにはSを入力してください:

var/log/syslogからログエントリを読み込む  
。etc/apparmor.dのAppArmorプロファイルを更新する:

プロファイル

/bin/ping-x

Capability: net\_raw 重

大度: 8

[1-ケイパビリティ・ネット\_raw,]。

(A)llow／[(D)eny]／(I)gnore／Audi(t)／Abo(r)t／(F)inish

ケイパビリティを許可するにはAを入力する:

ケイパビリティnet\_rawをプロファイルに追加。

プロファイル /ネットワ

ークファミリー: inet

ソケットタイプ 生



[1 - ネットワーク inet raw,].

(A)llow / [(D)eny] / (I)gnore / Audi(t) / Abo(r)t / (F)inish

ネットワークファミリーを許可するにはAを入力する:

プロファイルにネットワーク inet raw を追加。

プロファイル / ネットワ

ークファミリー: inet

ソケットタイプ ドグラム

[1 - #include <abstractions/nameservice>]。

2 - ネットワーク inet dgram、

(A)llow / [(D)eny] / (I)gnore / Audi(t) / Abo(r)t / (F)inish

ソケットタイプのデータグラムをプロファイルに追加するには、Aを入力する:

プロファイルに#include <abstractions/nameservice>を追加。

= ローカルプロファイルの変更

以下のローカルプロファイルが変更されました。保存しますか? [1

- /bin/ping-x]をクリックしてください。

(変更履歴) / 選択したプロファイルの保存 / [(変更履歴)] / (リーンプロファイル)間の変更履歴の表示 / その他

Sを入力して新しいプロファイルを保存します:

bin/ping-xのプロファイルを更新しました。

プロファイリング: /bin/ping-x

AppArmorイベントのシステムログの取得] / [終了

Fを入力して終了:

bin/ping-xをenforceモードに設定。

AppArmorプロファイルをenforceモードで

リロード。

新しいプロフィールの投稿をご検討ください!

詳細は以下のウィキページを参照: <http://wiki.apparmor.net/index.php/Profiles>

bin/ping-xのプロファイル生成を終了しました。

6. `etc/apparmor.d/bin.ping-x`に保存されている、作成され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: 最終更新: Tue Oct 17 11:30:47 2017

#インクルード <tunables/global>

/bin/ping-x {

#include <abstractions/base>

#include <abstractions/nameservice>

    ケイパビリティ

    net\_raw、ネットワー

    ク inet raw、

    /bin/ping-x mr、

    /lib/x86\_64-linux-gnu/ld-\*.so mr、

}

7. **aa-genproc**ユーティリティが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。ポリシーは、systemctl reload apparmorコマンドでオンデマンドで再ロードできます。潜在的な問題を回避し、変更が生き残ることを確認するには、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する。IPアドレスでlocalhostにpingを打ちます:

```
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。
127.0.0.1からの64バイト: icmp_seq=1 ttl=64 time=0.057 ms
127.0.0.1からの64バイト: icmp_seq=2 ttl=64 time=0.043 ms
127.0.0.1からの64バイト: icmp_seq=3 ttl=64 time=0.095 ms
```

--- 127.0.0.1のping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だ。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には、**IPv6**のlocalhostアドレスにpingを送信してみます。

これは失敗するはずだ:

```
student@ubuntu:~$ ping-x -c3 -6 ::1
ping: socket: パーミッションが拒否
されました
```

(注意: -6オプションは**IPv6のみ**を使用することを意味し、:1は**IPv6**の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ています。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する:

```
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、先ほどと同じように**aa-genprof**を再実行し、**window2**で**IPv6**ループバックにpingを打ち、次のオプションを追加する。



## 第34章

# システム・レスキュー



|      |                             |     |
|------|-----------------------------|-----|
| 34.1 | レスキューメディアとトラブルシューティング ..... | 544 |
| 34.2 | レスキュー／リカバリーメディアの使用 .....    | 545 |
| 34.3 | システムの救出と回復 .....            | 546 |
| 34.4 | エマージェンシー・ブート・メディア .....     | 547 |
| 34.5 | レスキューメディアの使用 .....          | 548 |
| 34.6 | 緊急モード .....                 | 550 |
| 34.7 | シングルユーザーモード .....           | 551 |
| 34.8 | 研究室 .....                   | 552 |

### 学習目標

このセッションを終えるまでに、あなたは次のことができるようになるはずだ：

- 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、どのように利用可能か、または準備可能かを説明する。
- システムの修復にはレスキューメディアを使用します。
- 緊急モードへの移行方法と、そこで何ができるかを知っておく。
- シングルユーザーモードへの移行方法、そこでできること、緊急モードとの違いを知る。

543



## 34.1 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ング

- レスキュー／リカバー・メディアは、トラブルシューティングにも使用できます。
- 光学メディア（**CD**、**DVD**）と**USB**ドライブがある。
- トラブルシューティング（および修理）に役立つユーティリティが含まれています：
  - ディスク・メンテナンス・ユーティリティ
  - ネットワーク・ユーティリティ
  - その他ユーティリティ
  - ログファイル

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**CD/DVD**やライブ**CD/DVD**、**USBメモリ**からブートする場合は、以下のようなツールを使うことができます。

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のためのディスクユーティリティ：  
**fdisk**、**mdadm**、**pvccreate**、**vgcreate**、**lvcreate**、**mkfs**、その他多数。
- ネットワークデバッグと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：  
**ifconfig**、**route**、**traceroute**、**mtr**、**host**、**ftp**、**scp**、**ssh**。
- 他にも数多くのコマンドが用意されている：



**bash**、**chroot**、**ps**、**kill**、**vi**、**dd**、**tar**、**cpio**、**gzip**、**rpm**、**mkdir**、**ls**、**cp**、**mv**、**rm**など。

## 34.2 レスキュー／リカバリメディアの使用

# レスキュー／リカバリメディアイメージの使用

- 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かもしれない。
- ローカルディスクまたはネットワークからのソフトウェアのインストールをサポートする。

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、それらは通常 `/mnt/sysimage` のどこかにマウントされているはずだ。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、その環境に変更することができる：

```
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求められることもある。

**chroot** 化された環境内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できます。たとえば **RPM** ベースのシステムでは、`--root` オプションを使ってルート・ディレクトリの場所を指定します：

```
sudo rpm -ivh --force --root=/mnt/sysimage /mnt/source/Packages/vsftpd-2*.rpm
```

### 34.3 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・メディアには複数の形態がある
- **緊急モード**に入り、便利な操作を行うことができます。
- また、**緊急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。

遅かれ早かれ、システムは、正しく起動しない、ファイルシステムをマウントできない、デスクトップ表示を開始できないなどの重大な障害を受ける可能性があります。光学ディスクやポータブル**USB**ドライブの形のシステムレスキューメディアを使用して、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**ツール一式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。



## 34.4 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
- スタンドアローンのレスキュー/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/リカバリ機能付きの**ライブLinuxディストリビューション**があります。

緊急ブート・メディアは、ファイルが見つからない、設定ミス、破損している、サービスが誤って設定されているなど、何らかの問題が原因でシステムが起動しない場合に便利です。

レスキューメディアは、rootパスワードが何らかの理由で失われたり、スクランブルされたりしてリセットする必要がある場合にも役立つ。

ほとんどのLinuxディストリビューションは、インストールメディア（CD、DVD、USB）および/または**ライブ**メディアをレスキューディスクとして二重の役割を果たすことを許可しており、これは非常に便利です。専用のレスキューディスクもあります。

ライブメディア（形式は問いません）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に起動可能なオペレーティングシステムを提供します。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、体験したり評価したり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・ドライブのような二次記憶装置がないコンピュータや、ハードディスク・ドライブやファイルシステムが破損したコンピュータで実行できるという点でユニークであり、ユーザーはデータを救出することができます。

## 34.5 レスキューメディアの使用

### レスキューメディアの使用

- 事実上すべてのLinuxディストリビューションには、**GRUB**メニューの選択肢にレスキューイメージがある。もしあなたのシステムがここまで来ていて、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す。
- システムが起動すると、次のいずれかのメニューを選択する必要があります。レスキューするか、次のように入力する。

#### Linux 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。
- おそらく、どのファイルシステムをマウントするかなども尋ねられるだろう。

ライブ・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する場合でも、レスキューとリカバリーのために特別なオペレーティング・システムに入る手順は同じであり、これまで指摘してきたように、1つのメディアが3つの目的すべてに役立ちます。

レスキュー／リカバリーモードは、システムがリムーバブルメディアから起動する際のブートメニューのオプションからアクセスできる。多くの場合、`rescue` と入力する必要があります：

#### 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が、確認しやすい手順を持っているため、すべての可能性をお伝えすることはできない。

次に、どの言語で続行するかなど、いくつかの質問とディストリビューションに依存する選択を求められます。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促されます：**CD/DVD**、**ハードドライブ**、**NFS**、**FTP**、**HTTP**。

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ます、その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。ベンダ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。見つければ `/mnt/sysimage` にマウントされる。シェルのプロンプトが表示され、システムに適切な修正を加えるための様々なユーティリティにアクセスできるようになります。

chrootは、ルート (/) ファイルシステムにアクセスするのに便利です。

## 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できる。
- ほとんどの場合、**USB**ドライブに転送できる：  
`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できる。

### 34.5. レスキューメディアの使用

549

多くのディストリビューションはboot.iso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（名前は異なるかもしれませんが）。ddを使って、これを**USB**のようなキードライブ：

```
dd if=boot.iso of=/dev/sdX
```

システムがリムーバブル・ドライブを/dev/sdXとして認識している場合。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ることに注意してください！

お使いのシステムが**USB**メディアからブートする機能を持ち、**BIOS**がそれ用に設定されている場合、この**USB**ドライブからブートすることができます。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**には存在しないので、この方法に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重要です。

**livecd-tools**や**liveusb-creator**のような便利なユーティリティは、インストールイメージを入手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することができ、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焼くという大変な作業をすべてやってくれます。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。





## 34.6 緊急モード

### 緊急モード

- 緊急モードは、最も最小限の環境で起動する。
- ルートファイルシステムが読み取り専用
- **init**スクリプトが実行されず、ほとんどセットアップが行われない
- **init**が失敗したときに役立つ
- ファイルシステムはまだマウントされており、データ復旧に役立つ
- 緊急モードを選択するには、**GRUB**インタラクティブメニューと対話する必要がある。
- コマンド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る。

**緊急モードでは**、可能な限り最小限の環境で起動する。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（次に説明する）に対する緊急モードの主な利点は、**init**が壊れたり動作しない場合でも、再インストール時に失われる可能性のある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**を押して編集する必要がある。次に、システムにブートを指示する前に、カーネルコマンドラインに**emergency**という単語を追加する。シェルプロンプトが表示される前に、**root**のパスワードが求められます。

ファイルシステムの破損など、さまざまな理由でブートが失敗した場合にも、緊急モードに入ることがあることに注意してください。

## 34.7 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、ログインができない場合に使用される。
- システムはランレベル1にブートする
  - initが開始される。
  - サービスが開始されない。
  - ネットワークが起動していない。
  - すべての可能なファイルシステムがマウントされる。
  - rootアクセスはパスワードなしで許可される。
  - 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する。

システムが起動するものの、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**をお試してください。

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するが、1つだけ例外があり、キーワード **emergency** をキーワード **single** に置き換える。

## 34.8 研究室

### 練習34.1: レスキュー／リカバリーメディアの使用準備



#### 非常に重要

以下の練習では、故意にシステムにダメージを与え、レスキューメディアを使ってリカバリーします。したがって、より野心的なことを試す前に、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のが賢明です。

ですからまず、専用のレスキュー／リカバリ・イメージか、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・イメージのレスキュー・メディアがあることを確認してください。

それを起動し、レスキューメディアから強制的にシステムを起動させる方法を確認し（BIOS設定をいじらなければ



#### ご注意

仮想マシンを使用している場合、論理的には同じ手順だが、2つの違いがある：

- BIOSへのアクセスは、使用するハイパーバイザーによっては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おこう。
- 物理的な光学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する必要があります。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のであれば、明らかに危険は少ない。もし修正不可能な方法でシステムを破壊するのが



#### 非常に重要

怖いのであれば、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・コピーを取っておけば、後でい

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、以下の練習は行わないでください！

### 練習 34.2: 壊れたGRUB設定からの回復

1. GRUB設定ファイル（`/boot/grub/grub.cfg`、`/boot/grub2/grub.cfg`、または`/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行を修正します。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かハードディスクデバイスノードのどちらかで識別される場合は、同様の簡単な変更を行ってください）。オリジナルのバックアップコピーを取っておく。



#### BLSCFGシステムの場合

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る。

2. マシンを再起動する。システムはブートに失敗し、No root device was foundなど并表示される。パニックが発生したこともわかります。
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（機能しているインストールサーバーにアクセスできる場合）。もう一度再起動します。ブートメニューが表示されたら、レスキューモードを選択します。
4. 別の方法として、**GRUB**メニューからレスキューイメージを選択してみることもできます。ほとんどのディストリビューションがこれを提供しています。レスキューメディアを使うのと同じ体験が得られますが、常にうまくいくとは限りません。例えば、ルートファイルシステムが損傷している場合、次のようになります。

何もできない。

5. レスキューモードで、ファイルシステムの検索を求められたら同意する。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する。
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、再起動方法の指示に従います。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 34.3: パスワードの失敗からの回復

1. rootとして（**sudoではなく**）、rootのパスワードを変更する。新しいパスワードが何であるか知らないふりをする。
2. ログアウトし、古いパスワードを使ってrootで再ログインを試みる。明らかに失敗します。
3. レスキューメディアを使って起動し、オプションでレスキューを選択する。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動する。
4. chroot化された環境に入る：  
`chroot /mnt/sysimage`  
をリセットし、rootパスワードを元の値に戻す。
5. 終了し、レスキューメディアを取り出して再起動すれば、正常にログインできるはずですが。

### 🔪練習 34.4: パーティションテーブルの破損からの回復



#### 非常に重要

この運動は危険であり、使用不可能なシステムを残す可能性がある。この練習を行う前に、物事を本当に理解しているかどうか確認してください。



#### ご注意

**GPT**の場合は、`sgdisk`に`-backup-file`と

--パーティショニングの章で説明したロード・バックアップ・オプション

1. rootでログインし、**MBR**を保存する：

```
dd if=/dev/sda of=/root/mbrsave bs=446 count=1
```

の1+0レコード

1+0レコードアウト

446バイト (446 B) のコピー、0.00976759秒、45.7 kB/s

注意：上記のコマンドを正確に実行し、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. これから**MBR**を消去する：

```
dd if=/dev/zero of=/dev/sda bs=446 count=1
```

```
の1+0レコード
```

```
1+0レコードアウト
```

```
446バイト（446 B）のコピー、0.000124091秒、3.6 MB/秒
```

3. システムを再起動する。
4. レスキュー環境で再起動し、**MBR**を復元する：  

```
dd if=/mnt/sysimage/root/mbrsave of=/dev/sda bs=446 count=1
```
5. レスキュー環境を終了し、再起動します。システムは正常に起動するはずです。

## 練習 34.5: インストールイメージを使ったリカバリ



### ご注意

この演習は特に**Red Hat**ベースのシステム用に書かれています。他のディストリビューションファミリーに適切な置換を簡単に構築できるはずです。

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
```

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んでいる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て、必要に応じて他のものを再インストールしてください。

2. レスキュー環境で起動する。
3. レスキュー環境から**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する：

```
rpm -ivh --force --root /mnt/sysimage /mnt/source/Packages/zsh*.rpm
```

-forceオプションは、依存関係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、何度もアップデートが行われている場合、手順全体が破綻する可能性があることに注意してください！

4. 終了して再起動する。
5. **zsh**が再インストールされていることを確認する：

```
rpm -q zsh
```

```
zsh-5.0.2-7.el7.x86_64
```

6. **zsh**

```
....
[coop@q7]/tmp/LFS201%
```



## 第35章

# 決算・評価調査



|      |            |     |
|------|------------|-----|
| 35.1 | 評価調査 ..... | 555 |
|------|------------|-----|

555

## 35.1 評価調査

### 評価調査

この度は**Linux Foundation**コースを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を測るため、また今後のセッションの改善に役立てるためです。

- インストラクターが提供するリンクを使用して、  
トレーニングを評価してください。
- 氏名のスペルを確認し、大文字と小文字を間違えないようにしてください。
- この情報は、あなたの**Cer-**を作成するために使用されます。



記入したEメールアドレスに修了証が送られるの

図35.1: コース概観

で、間違いがないことを確認すること。