



Linux
Professional
Institute

Linux標準教科書 (Ver.2.0.0)

LPI-JAPAN



Linux標準教科書

2012-10-01 版 LPI-Japan 発行

Linux 標準教科書 (Ver.2.0.0)

まえがき

このたび、特定非営利活動法人エルピー・アイ・ジャパンは、Linux 技術者教育に利用していただくことを目的とした教材、「Linux 標準教科書」を開発し、インターネット上にて公開し、提供することとなりました。この「Linux 標準教科書」は、多くの教育機関から、Linux を「基礎」から学習するための教材や学習環境の整備に対するご要望があり、開発したものです。公開にあたっては、「Linux 標準教科書」に添付されたライセンス（クリエイティブ・コモンズ・ライセンス）の下に公開されています。本教材は、最新の技術動向に対応するため、随時アップデートを行なっていきます。また、テキスト作成やアップデートについては、意見交換のメーリングリスト及び Wiki で、誰でもオープンに参加できます。

メーリングリストの URL

<http://list.ospn.jp/mailman/listinfo/linux-text>

Linux 標準教科書 Wiki

<http://www.lpi.or.jp/linuxtext/wiki/>

執筆者・制作者紹介

岡田 賢治（バージョン 1.0 執筆担当）

UNIX/Linux を初めて触ってから 15 年、ユーザ・管理者そして育成に携わってきました。本テキストは、実際に用いた説明方法などを使い、今までのノウハウを集約して執筆したつもりです。LPIC を通じての Linux 技術者の発展と、育成に関わる先生方のお役に立てれば幸いです。

川井 義治（バージョン 1.0 執筆担当）

Linux の講義をするとき、多肢に渡る知識と解説が必要で、前後の内容が複雑に絡み合うため、教材選定に苦労したり、自作教材を使っていました。今回は、多くの内容の中から実践として使える項目を選び、使いやすい並びを目指して書きました。学校教材の選択肢の一つや、個人教材として使って頂ければ幸いです。

宮原 徹（図版作成・企画進行担当）

本教科書は、Linux/オープンソースソフトウェアをこれから勉強する皆さんと、熱心に指導に当たられている先生方の一助になればと思い、作成いたしました。今後、2 度、3 度と改訂を行って、よりよい教科書にしていくために、使ってみての感想や改善提案などお寄せいただければと思います。

佐久間 伸夫（バージョン 1.1 執筆担当）

改訂にあたり、現場でよく使うコマンドやオプションを取り入れてみました。教育機関はもちろん、会社での社員教育にも活用いただければ幸いです。

遠山 洋平（バージョン 2.0 執筆・校正担当）

本教科書がより良い教科書になるよう、校正、加筆させていただきました。お気づきの点などございましたら「Linux 標準教科書開発用 Wiki」まで是非ご報告ください。

田口 貴久（バージョン 2.0 執筆・技術検証担当）

改版にあたり、初心に振り返り自分自身習得しにくかった部分を重点的に見直し調整しました。Linux 技術者を目指す方のお役に立てれば幸いです。

松田 神一（バージョン 2.0 監修）

授業あるいは自習用の教材として使うために、よりわかりやすく、かつ実践的な内容になるよう、内容の改訂を行いました。

木村 真之介（バージョン 1.1 編集協力・Wiki 管理）

改訂にあたり、皆様から頂きました誤植等の情報を元に修正と文章表現の改善を試みました。

高橋 征義（バージョン 2.0 PDF／EPUB 版制作担当）

本教科書の PDF・EPUB 作成のお手伝いをいたしました。Linux エンジニアの方々の技術力向上に貢献できれば幸いです。

著作権

本教材の著作権は特定非営利活動法人エルピーアイジャパンに帰属します。Copyright(C) The Linux Professional Institute Japan. All Rights Reserved.



使用に関する権利

●表示

本教材は、特定非営利活動法人エルピーアイジャパンに著作権が帰属するものであることを表示してください。

●改変禁止

本教材は、改変せず使用してください。本教材に対する改変は、特定非営利活動法人エルピーアイジャパンまたは特定非営利活動法人エルピーアイジャパンが認める団体により行われています。フィードバックは誰でも参加できるメーリングリストで行われていますので、積極的にご参加ください。<http://list.ospn.jp/mailman/listinfo/linux-text>

●非営利

本教材は、営利目的（※）以外で教材として自由に利用することができます。教材として営利目的での利用は、特定非営利活動法人エルピーアイジャパンによる許諾が必要です。本教材を利用した教育において、本教材自体の対価を請求しない場合は、営利目的の教育であっても基本的に使用できます。その場合も含め、LPI-Japan 事務局までお気軽にお問い合わせください。

（※）営利目的の利用とは以下のとおり規定しております。

- 営利企業において、本教科書の複製を用いた研修や講義を行うこと、または非営利団体において有料セミナー等に利用すること

本教科書の使用に関するお問合せ先

特定非営利活動法人エルピーアイジャパン（LPI-Japan）事務局

〒106-0041 東京都港区麻布台 1-11-9 CR 神谷町ビル 7F

TEL : 03-3568-4482

FAX : 03-3568-4483

E-Mail : info@lpi.or.jp

この教科書について

本書は、これから Linux を使ったサーバー構築について学習する方を対象にした教科書です。授業で使用することを想定していますが、自習教材としても使用できます。

本書の記述

本書では、通常の解説のほかに、以下の項目が用意されています。

○実行例

コマンドの動作などを説明するための例です。同じように実行して動作を確認してください。

○実習

実際に一連の操作を行なって動作を確認します。実行例よりも操作が複雑になるので、1つずつしっかりと操作してください。

○章末テスト

その章で解説されているポイントがきちんと理解できたか、確認してください。

想定している実習環境

Linux にはさまざまなディストリビューションが存在しますが、本教科書では CentOS を使用しています。本教科書は、多くの場合 CentOS 以外のディストリビューションを使って学習することも可能ですが、設定ファイルの記述方法やファイルの保存場所、コマンドなどは一部異なる場合がありますのでご注意ください。

本教科書では設定や操作は基本的にコマンドを使って行ないます。

目次

Linux 標準教科書 (Ver.2.0.0)	i
まえがき	i
執筆者・制作者紹介	i
著作権	ii
使用に関する権利	iii
この教科書について	iv
本書の記述	iv
想定している実習環境	iv
 第 1 章 Linux とは	1
1.1 基本ソフトウェアと応用ソフトウェア	2
1.1.1 基本ソフトウェアの役割	2
1.2 UNIX	3
1.2.1 UNIX の誕生	3
1.2.2 さまざまな分離・統合	3
1.2.3 派生 UNIX	3
1.2.4 Linux の誕生	4
1.3 Linux の特徴	5
1.3.1 カーネルとユーザランド	5
1.3.2 Linux を使う	5
1.3.3 シェル	6
1.3.4 ログイン	6
1.4 ディストリビューション	7
1.4.1 ディストリビューションの誕生	7
1.4.2 パッケージ	7
1.4.3 パッケージマネージャ	7
1.5 1 章 章末テスト	8
 第 2 章 Linux のインストール	9
2.1 実習で利用するハードウェア	10
2.2 利用する Linux のディストリビューション	10
2.2.1 インストール DVD の入手方法	11

2.2.2	バージョン	12
2.3	インストールの前に用意するもの	12
2.4	インストールの開始	12
2.5	インストール直後の初期設定	23
2.6	ログインする	27
2.7	コマンドの実行	29
第 3 章	基本的なコマンド	31
3.1	ファイル操作	32
3.1.1	ファイルとディレクトリ	32
3.1.2	ファイルやディレクトリの参照 (ls)	33
3.1.3	ドットファイルの表示	36
3.1.4	オプションの同時指定と順序	36
3.1.5	ファイルのコピー (cp)	37
3.1.6	ファイルの移動 (mv)	39
3.1.7	ファイルの削除 (rm)	41
3.2	ディレクトリの操作 (pwd,cd,mkdir,rmdir)	43
3.2.1	現在のディレクトリの表示 (pwd)	43
3.2.2	ディレクトリの変更 (cd)	43
3.2.3	ディレクトリの作成 (mkdir)	44
3.2.4	ディレクトリの削除 (rmdir)	45
3.2.5	特別なディレクトリ	46
3.2.6	絶対 (パス) 指定と相対 (パス) 指定	47
3.3	ファイルの内容を表示	48
3.3.1	ファイルの内容を表示 (cat)	48
3.3.2	ページャを使った表示	49
3.4	ファイルの検索 (find)	50
3.5	コマンドのパス	51
3.5.1	コマンドのパスを表示 (which)	51
3.6	ヘルプの使い方	51
3.7	マニュアルの使い方	52
3.7.1	セクション	53
3.8	3 章 章末テスト	55
第 4 章	正規表現とパイプ	57
4.1	標準入出力	58
4.2	リダイレクト	59
4.2.1	出力のリダイレクト	59
4.2.2	cat コマンドによるファイル作成	59
4.3	標準エラー出力	61
4.4	パイプ	61

4.5	grep コマンド	63
4.5.1	正規表現	63
4.5.2	さまざまな条件を用いた grep コマンドの実行	64
4.5.3	標準出力にマッチさせる	65
4.6	4 章 章末テスト	66
第 5 章 基本的なコマンド 2		67
5.1	ファイルのタイムスタンプの変更 (touch)	68
5.2	ファイルの一部の取得 (head, tail)	69
5.2.1	head	69
5.2.2	tail	69
5.2.3	特別なオプション -f	70
5.3	テキストファイルのソート (sort)	75
5.3.1	ファイルの準備	75
5.3.2	sort の実行	75
5.3.3	n 列目のデータソート (-k)	76
5.3.4	数値式でのソート	77
5.4	行の重複の消去 (uniq)	79
5.5	文字列の置き換え (tr)	80
5.6	ファイルの比較 (diff)	81
5.7	5 章 章末テスト	84
第 6 章 vi エディタ		85
6.1	vi の基本操作	86
6.1.1	ファイルを開く	86
6.1.2	ファイルを閉じる	86
6.1.3	ファイルを保存する	87
6.1.4	ファイルを保存して終了する	88
6.1.5	ファイルを保存せずに強制的に閉じる	88
6.2	インサートモードとコマンドモード	93
6.2.1	テキストの入力	94
6.2.2	カーソルの移動	96
6.3	編集中の大きな移動	97
6.3.1	ページ単位の移動	97
6.3.2	行を指定した移動	97
6.4	様々な編集操作	98
6.4.1	文字のカット・アンド・ペースト	99
6.4.2	行のカット・アンド・ペースト	100
6.4.3	行のコピー・アンド・ペースト	102
6.4.4	編集の取り消し (アンドウ)	103
6.5	置換と検索	104

6.5.1	文字列の検索	105
6.5.2	文字列の置換	106
6.6	6章 章末テスト	111
第7章 管理者の仕事		113
7.1	グループとユーザ	114
7.1.1	ユーザ	114
7.1.2	ユーザの作成	114
7.1.3	ユーザアカウントの変更	116
7.1.4	ユーザの削除	117
7.1.5	グループ	117
7.1.6	グループの作成	118
7.1.7	グループの登録情報の変更	118
7.1.8	グループを削除	119
7.2	パスワードとパスワードファイル	120
7.2.1	パスワードファイル (/etc/passwd)	120
7.2.2	グループファイル (/etc/group)	121
7.2.3	パスワード	121
7.2.4	シャドウファイル (/etc/shadow)	122
7.3	用意されているユーザとグループ	123
7.3.1	一般のユーザとグループ	123
7.3.2	root ユーザ	123
7.3.3	su コマンド	124
7.3.4	root ユーザでコマンドを実行する sudo コマンド	125
7.4	7章 章末テスト	127
第8章 ユーザ権限とアクセス権		129
8.1	ファイルの所有者と所有グループ	129
8.1.1	所有者の変更	129
8.1.2	所有グループの変更	130
8.2	ファイルとアクセス権	131
8.2.1	ファイルに設定できるアクセス	131
8.2.2	アクセス権の変更	133
8.2.3	ファイル作成のモード	136
8.3	8章 章末テスト	138
第9章 シェルスクリプト		139
9.1	シェルとシェルスクリプト	141
9.2	プログラミング	142
9.2.1	プログラムの例	142
9.2.2	プログラムの要素	142

9.3	シェルスクリプトの作成	142
9.3.1	シェルスクリプトの作成	142
9.3.2	echo コマンド	144
9.3.3	変数	144
9.3.4	read コマンド	146
9.3.5	シェル変数	146
9.3.6	環境変数	147
9.3.7	コメント	147
9.3.8	引用符	148
9.3.9	引数	149
9.3.10	shift 文	149
9.3.11	エスケープシーケンス	150
9.3.12	source コマンド	151
9.4	条件分岐	152
9.4.1	if 文	152
9.4.2	ファイル属性の確認	153
9.4.3	複数の条件を重ねる	154
9.4.4	一対多の条件分岐	155
9.5	繰り返し	157
9.5.1	for 文	157
9.5.2	while/until 文	157
9.5.3	select 文	158
9.5.4	繰り返しの制御	159
9.6	サブルーチン	160
9.6.1	関数	160
9.6.2	return 文	161
9.7	実際のシェルスクリプト	161
9.7.1	起動スクリプト	161
9.7.2	関数のシェルスクリプト	162
9.8	デバッグ	163
9.8.1	sh コマンド	163
9.9	9章 章末テスト	164
第 10 章	ネットワークの設定と管理	165
10.1	TCP/IP とは	166
10.1.1	ローカルエリアネットワークとワイドエリアネットワーク	166
10.1.2	IP とは	166
10.1.3	TCP と UDP とは	168
10.1.4	IP アドレス	168
10.1.5	IP アドレスのクラス	169

10.1.6 プライベート IP アドレス	169
10.1.7 サブネットマスク	169
10.1.8 経路の確認	170
10.1.9 ネットワークインターフェース	172
10.1.10 IP アドレスを確認	172
10.1.11 IP アドレスの設定ファイル	173
10.1.12 インターフェースの設定	175
10.1.13 ルーティングの確認	176
10.1.14 ルーティングの変更	176
10.1.15 DNS を使う設定	178
10.1.16 名前 (FQDN)	179
10.1.17 ポート番号	180
10.1.18 サービスの確認	181
10.1.19 ネットワークセキュリティの設定	182
10.1.20 ファイアウォールの設定	183
10.2 10 章 章末テスト	185
第 11 章 ファイル管理	187
11.1 Linux のファイル管理	189
11.1.1 ファイルシステムとは	189
11.1.2 パーティションとは	189
11.1.3 Linux のディレクトリ構造	189
11.1.4 ファイルシステムの作成方法	191
11.2 ディスクのパーティション	191
11.2.1 パーティション分割する理由	191
11.2.2 パーティションの分割	193
11.2.3 ハードディスクを増設して利用可能にする	194
11.3 ファイルシステム	199
11.3.1 ジャーナリング機能	200
11.3.2 マウント状態の表示	200
11.3.3 ファイルシステムの作成	201
11.3.4 ラベル	202
11.4 マウント	203
11.4.1 マウントポイント	203
11.4.2 マウント (mount コマンド)	203
11.4.3 アンマウント (umount コマンド)	204
11.5 スワップ領域の作成	206
11.5.1 スワップファイルシステムの作成	206
11.5.2 スワップの領域作成と利用	206
11.6 自動マウント	209

11.7	CD/DVD/USB メモリ (リムーバブルメディア) の利用	210
11.7.1	CD/DVD のマウント	210
11.7.2	CD/DVD のアンマウントと取り出し	211
11.7.3	アンマウントできない場合	211
11.8	i ノード	212
11.8.1	i ノード情報の確認	212
11.8.2	i ノード番号の確認	213
11.9	ハードリンクとシンボリックリンク	214
11.10	ディスクを管理するコマンド	216
11.10.1	ファイルシステムのチェックと修復	216
11.10.2	fsck とジャーナリング機能	216
11.10.3	ディレクトリ使用量の確認	217
11.11	11 章 章末テスト	218

第1章

Linux とは

Linuxについて説明するにあたり、それに必要な知識である基本ソフトと応用ソフトの考え方を学習します。基本ソフトの用途、さらにそれらの特徴と、基本ソフトの中でもさらに「カーネル」と「ユーザランド」の働きを学習します。

この章の内容

- 1.1 基本ソフトウェアと応用ソフトウェア
 - 1.1.1 基本ソフトウェアの役割
- 1.2 UNIX
 - 1.2.1 UNIX の誕生
 - 1.2.2 さまざまな分離・統合
 - 1.2.3 派生 UNIX
 - 1.2.4 Linux の誕生
- 1.3 Linux の特徴
 - 1.3.1 カーネルとユーザランド
 - 1.3.2 Linux を使う
 - 1.3.3 シエル
 - 1.3.4 ログイン
- 1.4 ディストリビューション
 - 1.4.1 ディストリビューションの誕生
 - 1.4.2 パッケージ
 - 1.4.3 パッケージマネージャ
- 1.5 章末テスト

1.1 基本ソフトウェアと応用ソフトウェア

動作しているコンピュータには、大きく分けて 2 つの部分があります。ハードウェアとソフトウェアです。ハードウェアとは、コンピュータの機械そのもののことを持します。ソフトウェアとは、ハードウェアで動作しているプログラムを持します。ゲーム機本体（＝ハードウェア）とゲームのソフト（＝ソフトウェア）の関係を想像するとわかりやすいのではないかでしょうか？ソフトウェアにも大きく分けて 2 種類あります。基本ソフトウェアと応用ソフトウェアです。基本ソフトウェアは Operating System(OS) のことを指し、応用ソフトウェアはその上で動くアプリケーションを指します。たとえば Windows や Linux、Mac OS X は基本ソフトウェア、Word や Excel、PowerPoint は応用ソフトウェアになります。

1.1.1 基本ソフトウェアの役割

基本ソフトウェアは応用ソフトウェアが動作する際に必要な「部品」を提供したり、ハードウェアという「資源」を管理するという役割があります。応用ソフトウェアを想像してみてください。応用ソフトウェアにはウィンドウがあり、メニューやツールバーアイコン、ファイルを開いたり保存する際に出てくる確認メッセージなどがあると思います。これらの機能を各応用ソフトウェアが用意していたのでは、応用ソフトウェアの作成が大変になってしまいます。このようにさまざまな応用ソフトウェアが使うであろう共通部品を提供するのが基本ソフトウェアの大変な役割の 1 つです。

それ以外にも「資源を管理する」という役割があります。この場合の「資源」とは、コンピュータが提供できる機能・能力を示しています。例えば、ワープロや表計算の基本ソフトウェアを同時に動かすことができます。本来のコンピュータは同時に 1 つの処理しかできません。それでもかかわらず同時に動作させることができるのは、各ソフトウェアを非常に短いタイミングで切り替えていくからです。これは基本ソフトウェアが司っている機能です。

1.2 UNIX

本教科書では Linux の基本操作をコマンドを実行しながら学ぶことができますが、ここではまず Linux の祖先にあたる UNIX がどうやってできたのか説明します。

1.2.1 UNIX の誕生

1960 年代に UNIX は通信会社 AT&T のベル研究所で誕生しました。当時ベル研究所では、先行研究として MULTICS という OS が開発されていました。一方、研究者である Ken Thompson 氏が研究所の片隅で使われていないコンピュータを発見し、自分で考えた機構を試験運用するために、そのコンピュータに小さな OS を搭載しました。これが UNIX のはじまりであるといわれています。

MULTICS はあまりにも多機能・高機能を目指していたため、研究開発が頓挫してしまいました。Ken Thompson 氏は、自分が作ったその小さな OS に、MULTICS の機能を取り入れ UNIX の原型となる OS が作られていきました。実際、UNIX は非常にコンパクトで小回りも利き、さまざまな実験プロジェクトにも用いられるようになりました。AT&T で文書校正のソフトウェアが必要になった際に、文書校正ソフトウェアを動かすための OS として用いられた経緯もあります。また、当時 AT&T は UNIX を商品として考えていなかったので、送料・メディア代等の経費を支払うだけで、自由に利用を許可していました。さらに、ソースコードといわれる UNIX の設計に関する基本部分をそのまま配布していたので、手に入れたユーザは独自に研究・開発・変更等を行なうことができました。

1.2.2 さまざまな分離・統合

配布された UNIX は、メンテナーの手によって自由に改造が施されました。その結果、さまざまな団体や企業により UNIX が作成されたため、UNIX としてのまとまりがなくなっていました。この問題を解決するため、AT&T が正式にライセンス契約を始め、UNIX を管理するようになりました。以降、AT&T(及び UNIX のライセンス管理団体)と契約を結んだ組織のみが、自社の出した OS を UNIX と宣言できるようになりました。一方、AT&T とライセンス契約を結んでいない組織もあります。これらの組織が開発した UNIX のことを UNIX 互換 OS と呼びます。

1.2.3 派生 UNIX

このように、互換 OS も含めて非常に多くの UNIX が広まっていきました。その派生した中で大きな影響を及ぼしたのがバーカークレー版 UNIX です。バーカークレー版 UNIX は、UNIX の開発者である Ken Thompson 氏がカリフォルニア大学バークレー校にいたときに作成した派生 OS です。この派生 OS の最大の特徴として、当時アメリカ軍内部の通信方式として採用された、IP(インターネットプロトコル) の実装が行われた点があげられます。UNIX がインターネットに強いと言われているのはこの経緯によるものです。

1.2.4 Linux の誕生

UNIX の多くは高いライセンス使用料のもと、企業や大学等で利用することが続いていました。自宅で、しかもパソコンで利用できることが多くのユーザの夢でした。

そんな中、「僕は今、UNIX 互換の OS を作っている」という投稿が、1991 年にネット上に流れました。投稿者は Linus Torvalds という、当時フィンランドの大学生で、コンピュータの機能の学習からそのようなプログラムを書き始めていたようです。そのプログラムはネットワーク上で公開されました。

プログラムの原型は非常に原始的なもので、コンピュータに詳しい人間のみが動かすことが可能でした。設計の基本は UNIX とそっくりでしたが、System V の流れも BSD の流れも持たない独自の UNIX 互換 OS でした。その後、Linus Torvalds が学習のために作り上げた小さなプログラムは、さらなる機能拡張やソフトウェア、ツールを組み合わせ、UNIX 互換の OS として立派に動作するまでになりました。そう、Linux が誕生したわけです。

Linux のプログラムにおいて、一番特徴的なところは、そのライセンス形式でした。Linux のプログラムには、GPL(GNU General Public License: GNU 一般公衆利用許諾) というライセンス形式が採用されています。GPL は GNU プロジェクトのリチャード・ストールマンにより作成されたフリーソフトウェアライセンスで、次の内容が含まれていることが特徴です。

- プログラムを実行する自由
- ソースの改変の自由
- 利用・再配布の自由
- 改良したプログラムをリリースする権利

Linux は多くのディストリビューターにより開発され、非常に多くのユーザに使われるようになりました。無償でかつ自由に利用、改変できるライセンス体系である GPL ライセンスを Linux が採用したためです。ソフトウェアの自由を目指した GPL ライセンスはライセンスの性格上、GPL ライセンスを有したソフトウェアをベースとしたソフトウェアを一般に公開する場合、要求があればソフトウェアのソースを公開しなくてはいけない (=コピーレフト) という絶対的な条件があります。

しかし、このライセンス体系が良いように働いた結果、Linux は一部の個人や組織に独占されることなく、全利用者がその恩恵を受けて発展させていくというスタイルができました。現在、さまざまなディストリビューターにより Linux が公開され、パーソナルコンピュータに容易に Linux がインストールできるのも、Linux が GPL ライセンスを採用したからこそなのかもしれません。

1.3 Linux の特徴

Linux はどういった仕組みで動作するのでしょうか。Linux の特徴についてここでご紹介しましょう。

1.3.1 カーネルとユーザランド

ソフトウェアは、基本ソフトウェアと応用ソフトウェアの 2 つに分かれると説明しました。基本ソフトウェアは、さらに 2 つの領域に分かれます。それらを「カーネル」と「ユーザランド」といいます。それぞれには、以下の特徴があります。

カーネル

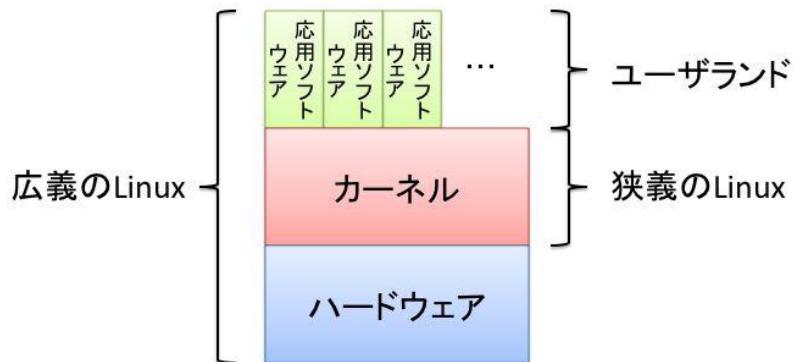
カーネルはオペレーティングシステムの中核となる部分で、ハードウェアと直接やりとりするなどもっとも中心的な機能を受け持つ部分です。カーネルはハードウェアの違いを吸収して、プログラムがどのようなハードウェア上でも同じように動作する役割があります。

ユーザランド

OS が動作するのに必要な、カーネル以外の部分のことです。ファイルシステムやファイル操作コマンド、シェルなどの基本的なソフトウェア群を指します。

1.3.2 Linux を使う

Linux は、基本的にはコマンドで操作します。コマンドはユーザランドで動作します。また、Linux に X Window System と GNOME、KDE、Xfce などのデスクトップ環境を導入することで、Windows や Mac などのようにマウス入力による操作が行なえるようになります。



1.3.3 シェル

前項で「Linux はコマンドで操作する」と説明しました。コマンドは文字通り「命令」のことです、さまざまな用途のコマンドがあります。Linux にはシェルという対話型のコマンド入力環境が用意されています。シェルは入力されたコマンドを理解し、実行します。

シェル自体には大きく 2 つの機能があります。1 つは、前述の通りコマンド入力を受け付けることです。もう 1 つはシェルスクリプトの実行にあります。シェルスクリプトとは、「コマンドの入力を自動化する」ためのものです。1 つのファイルにコマンドを 1 行ずつ記述して作成します。作成したシェルスクリプトを実行することで、コマンドの実行を自動化することができます。また、作成したシェルスクリプトをサーバーの起動時に実行したり、数時間毎に実行したりすることができます。

1.3.4 ログイン

Linux では、利用開始時にユーザ名とパスワードを入力します。このユーザ名とパスワードの組み合わせをアカウントといいます。アカウントを使って Linux を使い始めるのを「ログインする」といいます。ログインするのは、そのコンピュータの利用許可を得るためです。パスワードはログイン名の認証に利用され、例えばログイン名を銀行の口座番号としたとき、パスワードは口座の暗証番号に相当します。

1.4 ディストリビューション

1.4.1 ディストリビューションの誕生

Linux は日々改良が進み、はじめは単機能しか備えていなかったものが、さまざまなハードウェア上で動くように改良されていき、次第に UNIX 互換 OS として機能するまでになりました。

しかし、当初の Linux はインストール作業が非常に困難で、一部のコンピュータのスキルが高いユーザしか使うことができませんでした。そこで、さまざまな団体が Linux を使う上で必要なプログラムをまとめ、簡単な手順で手軽にインストールできるようにしました。これが Linux ディストリビューションのはじまりです。このように Linux ディストリビューションを開発する団体をディストリビューターといい、代表的な Linux ディストリビューターとしては Red Hat や Debian Project、Ubuntu を開発している Canonical などがあり、さまざまな団体によって「Linux」はリリースされ続けています。

1.4.2 パッケージ

パッケージは Linux に対して追加機能を提供するものです。従来、応用ソフトウェアの実行を Linux 上で行なうにはソースをダウンロードして自らビルドする必要がありました。これは非常に時間と手間がかかります。この問題を解決するため、ディストリビューターの手によりビルドして応用ソフトウェアを簡単に導入できるようにするためにパッケージが作られるようになりました。

1.4.3 パッケージマネージャ

応用ソフトウェアの導入には、複数の応用ソフトウェアやこの動作を補足するライブラリが必要な場合があります。依存するプログラムがさらに別のプログラムを依存しているとなると、それらを正しい場所に正しい順序でインストールしなくてはなりません。

さらに開発が盛んなプログラムは、バグの修正や機能改善、セキュリティ脆弱性の修正など常に更新されています。新しいプログラムをインストールするときは、現在稼働している古いプログラムをきれいに削除してインストールします。インストールする作業もさることながら、削除する作業も非常に困難です。

そこでパッケージを簡単にインストールしたり、アップデートするパッケージマネージャというものが標準で用意されるようになりました。今日の Linux ディストリビューションには何らかのパッケージマネージャが必ず用意されており、インターネットに接続されれば最新のパッケージ（と、依存するパッケージやライブラリ）を導入、更新することができるようになっています。

1.5 1章 章末テスト

(1) 基本ソフトウェアとして適切なものを選びなさい。

1. Word
2. Excel
3. Windows
4. Linux

(2) 応用ソフトウェアとして適切なものを選びなさい。

1. Word
2. Excel
3. Windows
4. Linux

(3) Operating System の役割として適切なものを 2 つあげなさい。

(4) Linux について正しく述べているものを選びなさい。

1. AT&T とライセンス契約を結んでいない組織が開発した UNIX のこと。
2. 開発者である Ken Thompson 氏がカリフォルニア大学バークレー校に在籍していたときに作成した派生 OS。
3. Linux はカーネルとユーザランドによって成り立つ。
4. 1960 年代に通信会社 AT&T のベル研究所で誕生した。

(5) パッケージを使って応用ソフトウェアを導入する利点を説明しなさい。

第2章

Linux のインストール

本章では Linux ディストリビューションの CentOS をインストールします。ここでインストールした Linux を使って次章以降でさまざまなコマンドを実行したり各種設定を行なっていきます。

この章の内容

- 2.1 実習で利用するハードウエア
- 2.2 利用する Linux のディストリビューション
 - 2.2.1 インストール DVD の入手方法
 - 2.2.2 バージョン
- 2.3 インストールの前に用意するもの
- 2.4 インストールの開始
- 2.5 インストール直後の初期設定
- 2.6 ログインする
- 2.7 コマンドの実行

2.1 実習で利用するハードウェア

本教科書の実習では、市販されているような一般的な構成の PC に Linux を導入して学習で利用します。この実習で必要なハードウェアの仕様は次の通りです。

マシン本体

Windows や Linux が動作する、いわゆる「パソコン」を想定しています。また、「VirtualBox」のような仮想化ソフトウェアを利用して実習環境を用意することもできます。

実装メモリ

CentOS 6 以降のバージョンで GUI による操作を行なうには、少なくとも 1024MB のメモリが推奨されています。実装メモリが少ない場合はテキストインストールが実行されることがあります。

DVD 光学ドライブ

本教科書の実習ではインストール用 DVD を利用するので、その光学ドライブが DVD を読み取りできる必要があります。マシンによっては光学ドライブが無い機種もあります。そういったときは、USB 等で接続する DVD ドライブを用意します。それを利用することにより、インストール DVD を起動することができます。

ハードディスク

Linux をインストールするためには記憶装置が必要です。ここでは記憶装置としてハードディスクを使います。インストールにはハードディスクに約 10GB の空き領域があれば十分なので、一般的な構成の PC では十分満たしていると思います。またハードディスクをフォーマット（初期化）して Linux をインストールするため、ハードディスクの中身は全部消去されます。その為、ハードディスクをクリアしてもいい PC を利用するか、環境のバックアップを取ってから作業を行なってください。

その他周辺機器

本体や DVD、光学ドライブ、ハードディスクドライブの他にも、一般的に利用するためにはキーボード、マウス、ディスプレイ等の周辺機器が必要です。キーボードは、日本語か英語かで設定が異なりますので、日本語キーボード、英語キーボードの区別を確認しておきます。

2.2 利用する Linux のディストリビューション

本教科書では、CentOS のバージョン 6.3、32 ビット版（以降 CentOS）を利用します。

CentOS 公式サイト

<http://www.centos.org/>

CentOS は、商用ディストリビューションである Red Hat Enterprise Linux の互換ディストリ

ビューションとして提供されています。本家 Red Hat とのバイナリ互換を保ちながら、サポートも同等を目指すという方針で開発されています。利用に際し費用が発生することはない、無償で提供されているディストリビューションです。

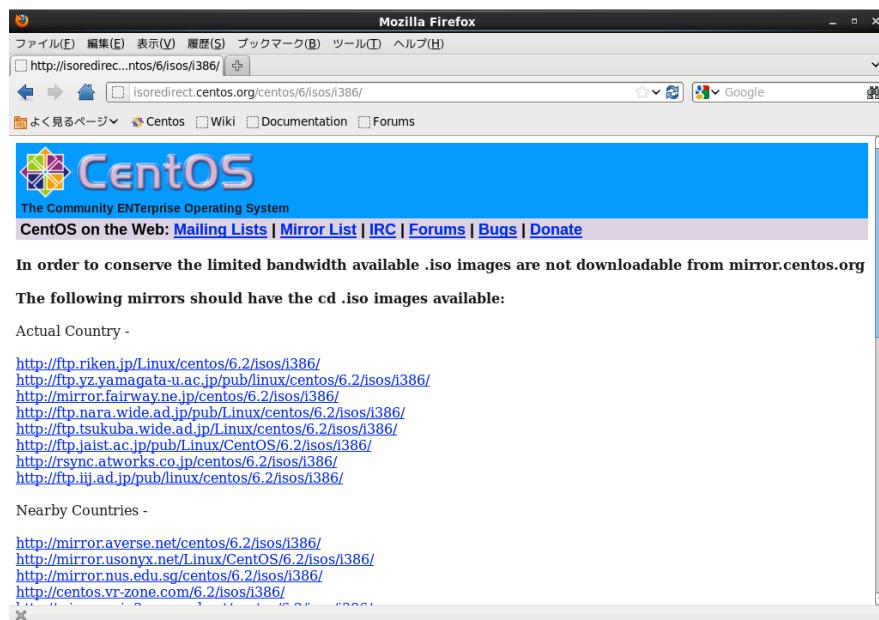
2.2.1 インストール DVD の入手方法

今回インストールには、DVD のインストーラーを利用します。CentOS のインストール用 DVD の入手方法には次の 2 通りが存在します。

● ISO イメージをダウンロードする

CentOS が配布している ISO イメージを、ダウンロードします。ダウンロード元の URL は以下のとおりです。CentOS の最新版の ISO イメージのページへのリンクになっています。

<http://isoredirect.centos.org/centos/6/isos/i386/>



この URL をクリックすると、多くのミラーサイトが表示されます。そのミラーサイトの中から ISO イメージをダウンロードします。例えば、riken（理化学研究所）が提供しているミラーサイトの URL であれば次のようにになります。

ダウンロードサイト

```
http://ftp.riken.jp/Linux/centos/6/isos/i386/
```

ダウンロードサイトを開き、CentOS-6.x-i386-bin-DVD1.iso といった名前の ISO イメージをダウンロードしてください。

2.3 インストールの前に用意するもの

DVD イメージは 2 枚組になっていますが、今回の実習では 1 枚目の DVD のみ使いますので CentOS-6.x-i386-bin-DVD1.iso といった名前の 1 枚目のディスクのみダウンロードします。ISO イメージはサイズが非常に大きいために転送に時間がかかります。ミラーサイトによっては、複数枚の CD の ISO イメージはあるものの DVD の ISO イメージを置いていないサイトも存在しますので、ダウンロードするサイトを選ぶときは注意してください。

また、BitTorrent を使ってのダウンロードも行えます。ダウンロードサイトに.torrent という拡張子のファイル名が置かれているので、このファイルをダウンロード後、BitTorrent に対応したソフトウェアを使ってダウンロードが行えます。

ダウンロードした ISO イメージは、DVD のライティングソフトウェアを使って DVD に書き込んでください。データとしてではなく、イメージとして書き込む点に注意してください。

●雑誌や解説書の付録

CentOS は雑誌に付属していたり、CentOS の解説書が多く出版されています。それらに付属しているインストール DVD を利用してもかまいません。

2.2.2 バージョン

本教科書では、本教科書の作成時点で最新であった CentOS を利用した手順について解説しています。より新しいバージョンの CentOS がリリースされていたら、新しいバージョンを使って実習にあたってください。

2.3 インストールの前に用意するもの

インストール DVD

CentOS のインストール DVD を用意します。

マシンの設定

インストールを開始するにあたり、マシンの設定を確認します。確認する内容は、BIOS で設定する「起動順序」です。起動順序を、必ず光学ドライブを優先にします。光学ドライブよりハードディスクの優先度が高いと、ハードディスクにインストールされている OS が起動してしまいます。

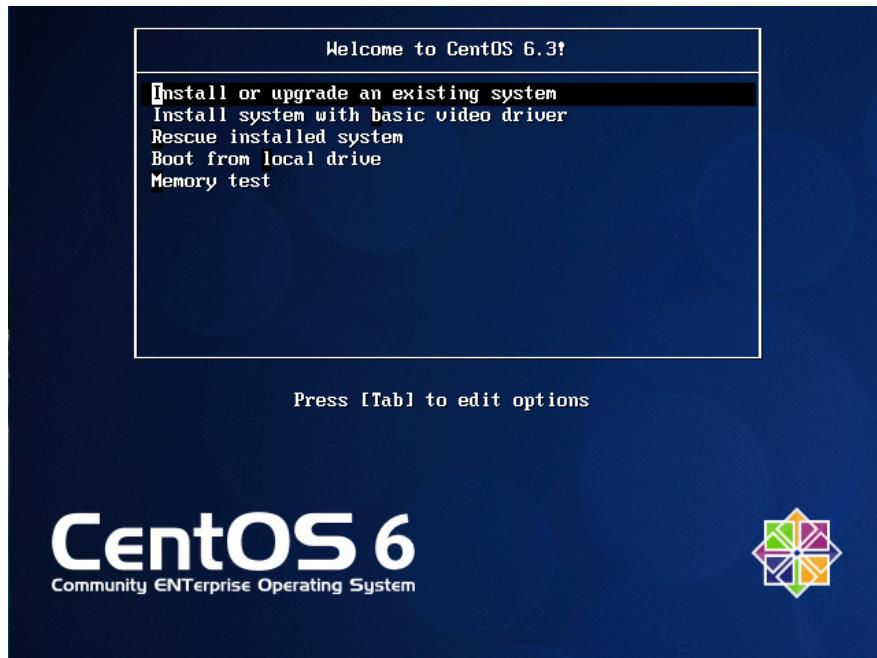
ハードディスク

本教科書では、CentOS をインストールする際にハードディスクの中身を消去します。従ってハードディスクの中身を消去しても良い PC を利用するか、ハードディスクの中身のバックアップを取りながら作業を行ないます。

2.4 インストールの開始

それでは、インストールを開始します。本教科書では比較的パッケージがそろった「Software Development Workstation」を選択してインストールします。想定している環境に応じて適切なものを選んでください。

1. インストール DVD を光学式ドライブにセットし、マシンを起動します。
2. 起動画面が現れます。「Install or upgrade an existing system」と表示されるので、Enterキーを押します。60秒間そのままにしても、自動的にインストールが開始します。

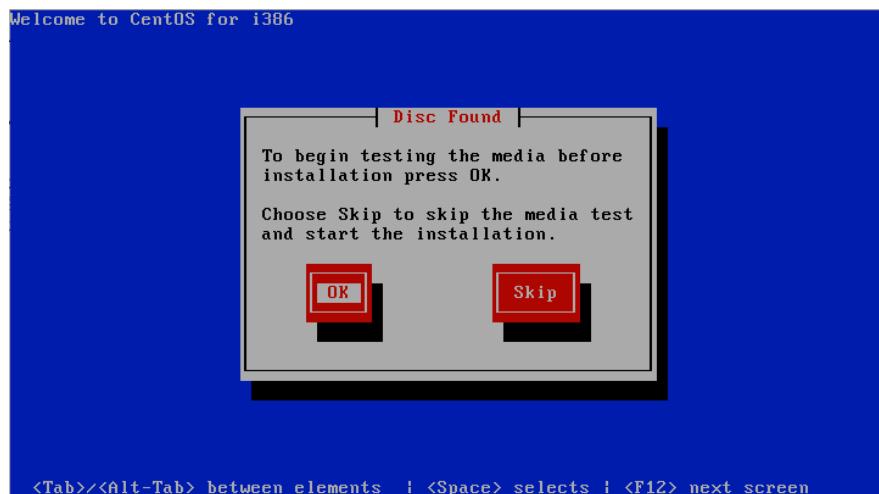


一部のマシンでは内蔵されているグラフィック性能の不足のため、正しくインストーラが起動しないことがあります。この場合は「Install system with basic video driver」を選択してください。

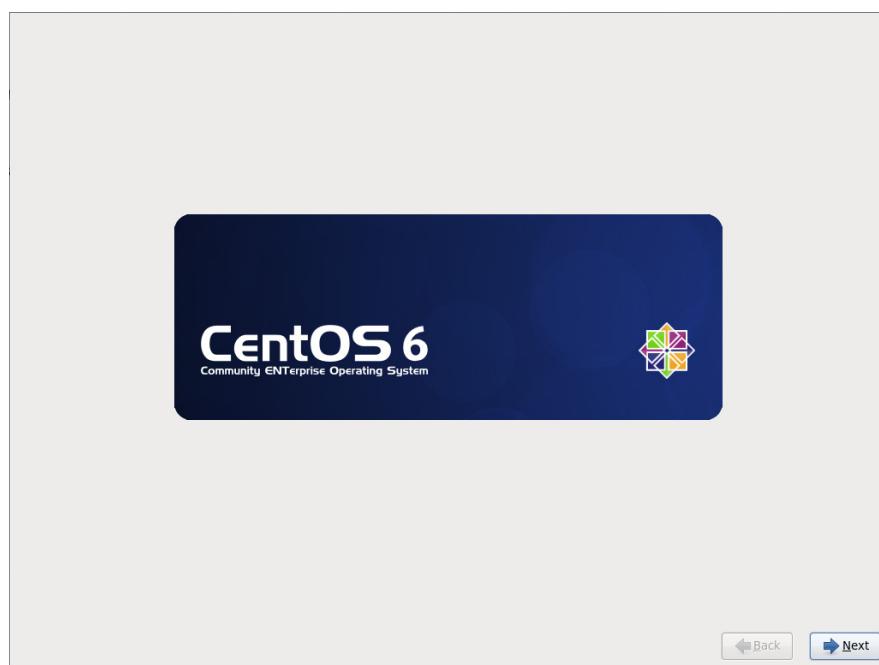
2.4 インストールの開始

3. 「Disc Found」のダイアログが現れます。メディアの内容の整合性をチェックします。チェックにはしばらく時間がかかります。チェックをする場合は「OK」、しない場合は「Skip」を選択します。

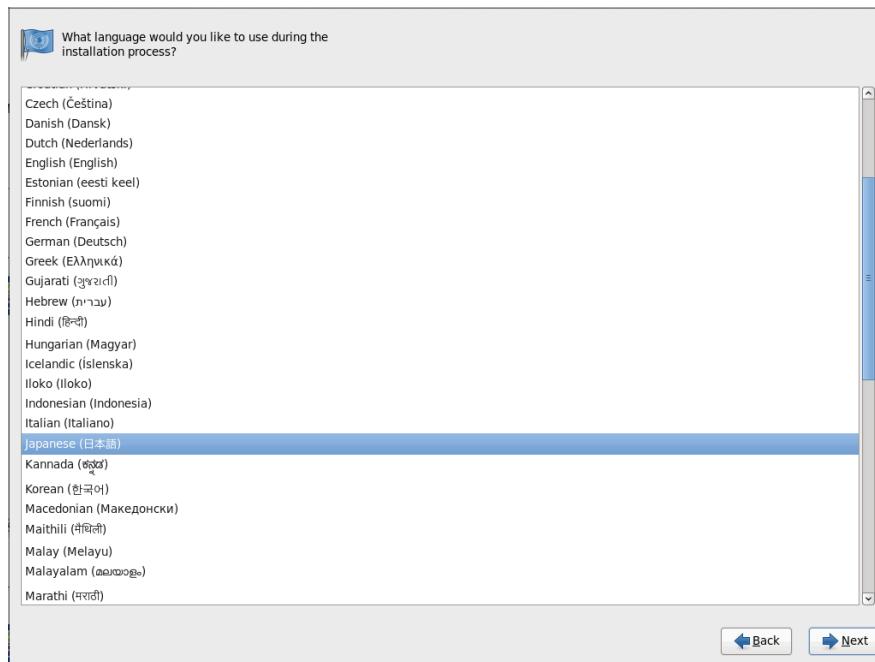
選択項目はカーソルキーか TAB キーで変更できます。選択は Enter キーです。



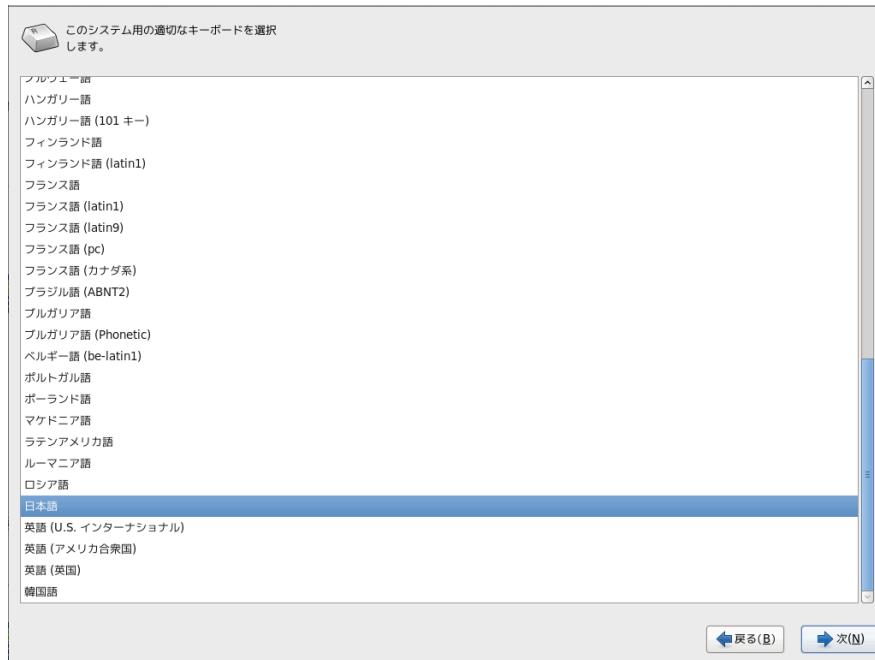
4. インストーラーのトップ画面が表示されるので、右下の「Next」をクリックします。



5. インストール時に利用する言語を選択します。「Japanese(日本語)」を選択し、右下の「Next」をクリックします。

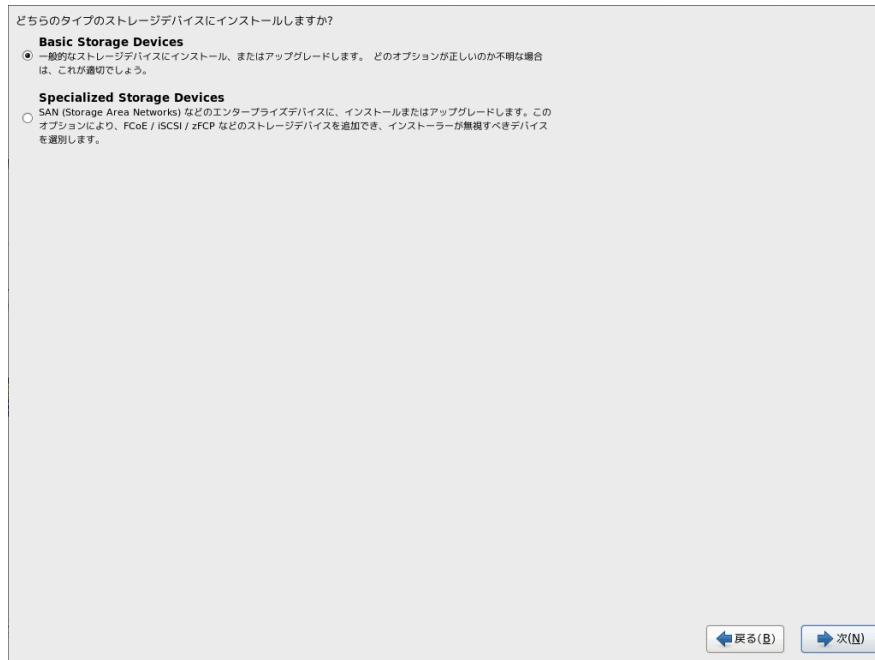


6. キーボードの選択画面が現れます。日本語キーボードのときは「日本語」、英語キーボードのときは「英語（アメリカ合衆国）」を選択し、右下の「次」をクリックします。



2.4 インストールの開始

7. ストレージの選択画面が現れます。「Basic Storage Devices」を選択し、右下の「次」をクリックします。



8. 「ストレージデバイスの警告」のダイアログが現れます。「はい。含まれていません。どのようなデータであっても破棄してください。」をクリックします。



9. ホスト名の指定の画面が現れますので、適切なホスト名を入力します。



10. 左下の「Configure Network」をクリックし、ネットワークの設定画面を呼び出します。

11. 「ネットワーク接続」のダイアログが現れます。設定したいネットワークインターフェース（例では「System eth0」）を選択し、「編集」をクリックします。



2.4 インストールの開始

12. ネットワークインターフェースの設定ダイアログが現れます。

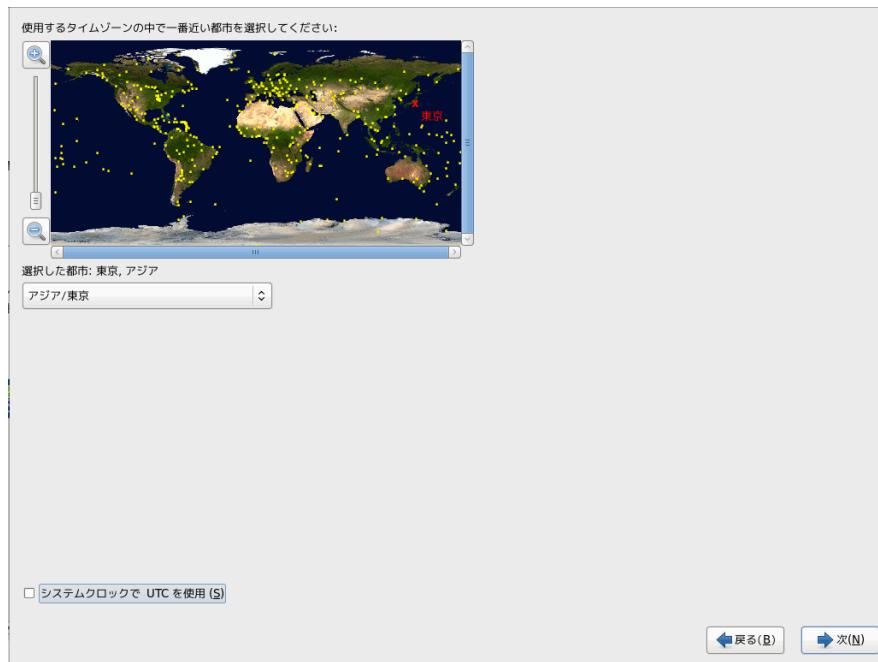
13. 「自動接続する」をチェックして「適用」をクリックします。



14. 「閉じる」をクリックして「ネットワーク接続」ダイアログを閉じます。

15. 右下の「次」をクリックします。

16. 時間帯を設定します。「アジア／東京」が選択されていることを確認し、「システムクロックで UTC を使用」のチェックを外して^{*1}、右下の「次」をクリックします。



17. root のパスワードを入力して、右下の「次」をクリックします。

^{*1} 「システムクロックで UTC を使用」でチェックをはずすのは、PC が持つ内部時計 (= BIOS の時計) を UTC (= 世界標準時) とみなしてしまうためです。多くの場合、内部時計は現在の時刻を現在の時間帯 (=日本では JST) で設定されているので、UTC と見る機能を OFF にするため、チェックをはずします。

2.4 インストールの開始

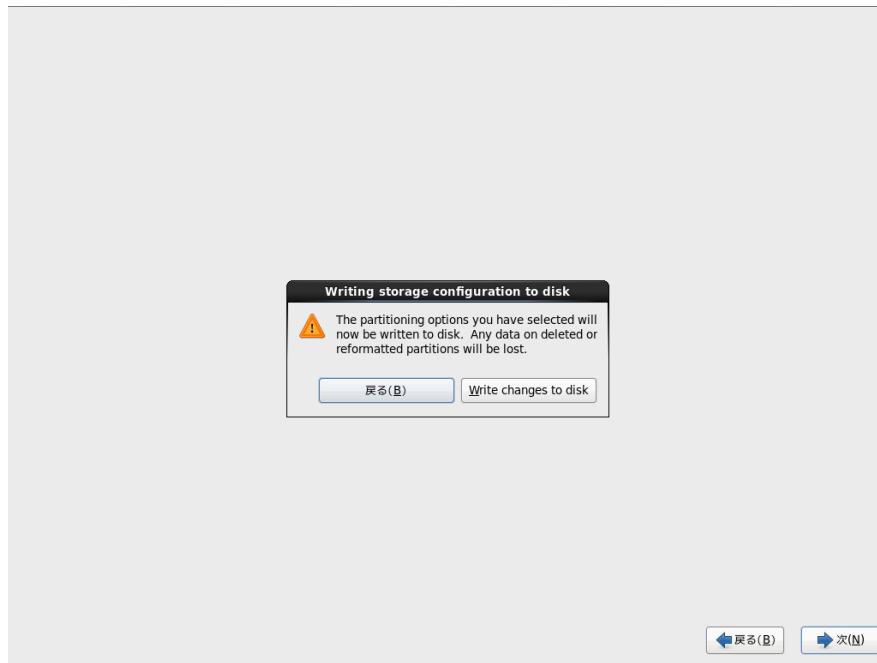
18. 入力した root のパスワードが短かかったりすると、「パスワードが弱すぎます。」というダイアログが表示されます。「キャンセル」をクリックしてもっと複雑なパスワードを再設定するか、「とにかく使用する」をクリックします。



19. ハードディスクをフォーマットする方法を尋ねられます。「Use All Space」を選択し、右下の「次」をクリックします。



「Writing storage configuration to disk」ダイアログが表示されたら、「Write changes to disk」をクリックします。



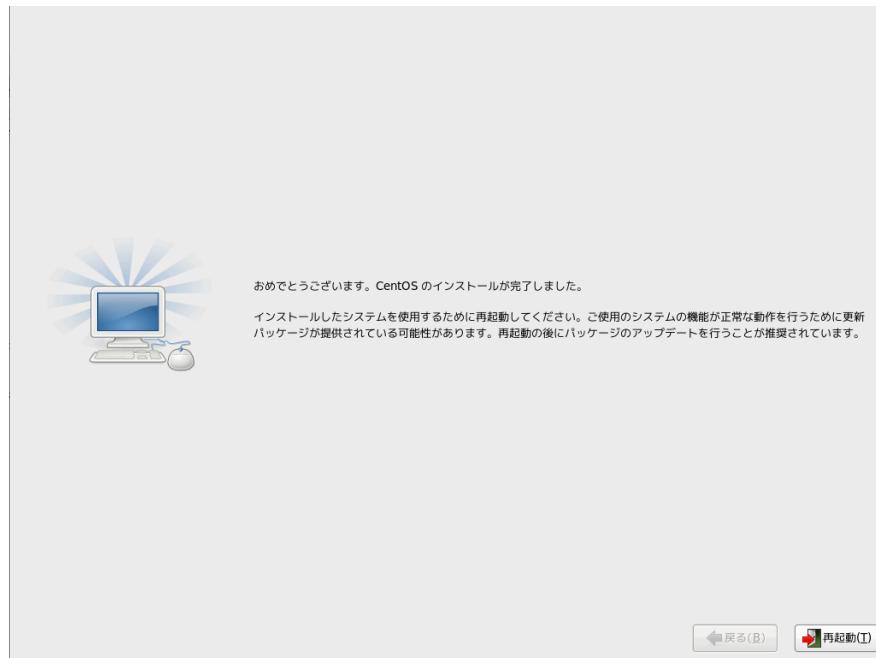
20. インストールするソフトウェアのグループを設定します。「Software Development Workstation」を選択し、右下の「次」をクリックします。



21. CentOS のインストールが始まります。

2.4 インストールの開始

22. インストールが終了すると、再起動が要求されます。右下の「再起動」をクリックします。

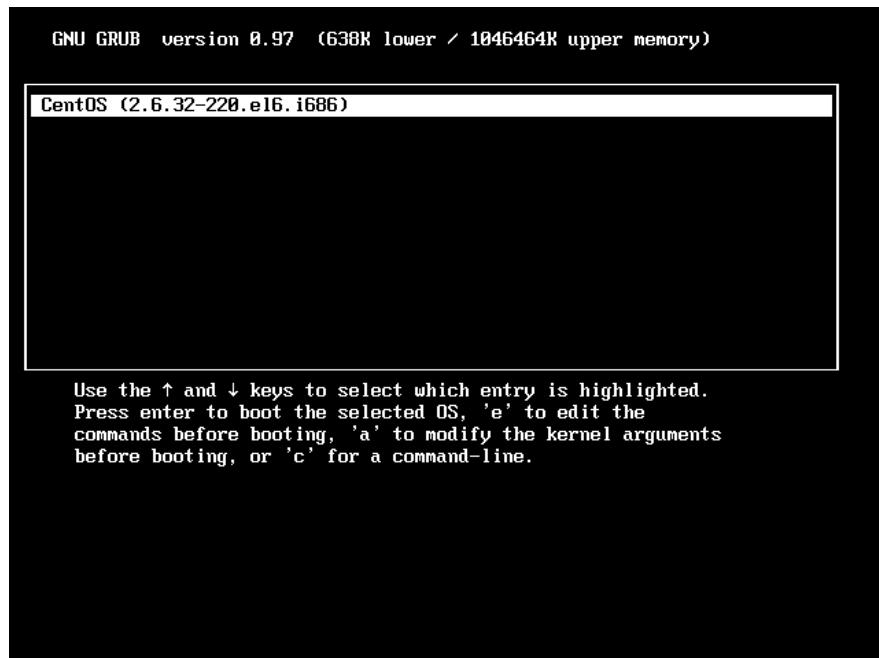


再起動する前に、インストール DVD を取り出します。もしくは、BIOS を起動して起動順序でハードディスクの優先順位を一番最初に設定します。そうしないと、次回の起動時もインストール DVD から起動してしまいます。

2.5 インストール直後の初期設定

インストール直後、最初の起動時にさまざまな設定を求められます。次に、その設定を行ないましょう。

1. インストールしたマシンを起動します。インストール作業から引き続いて作業するときは、再起動になります。ブートローダーが起動する OS の選択を求めるので、そのまま待つか、Enter キーを押します。

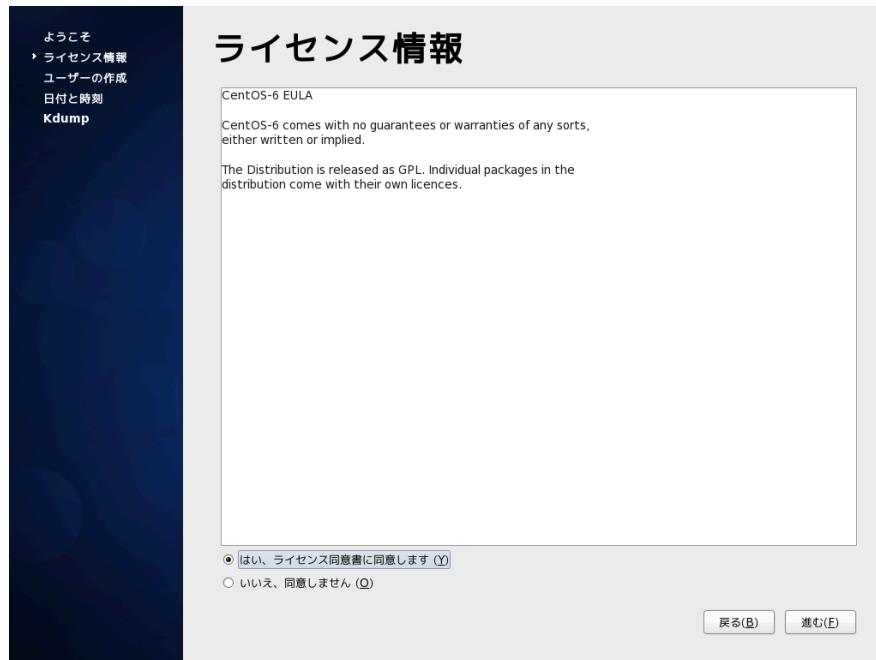


2.5 インストール直後の初期設定

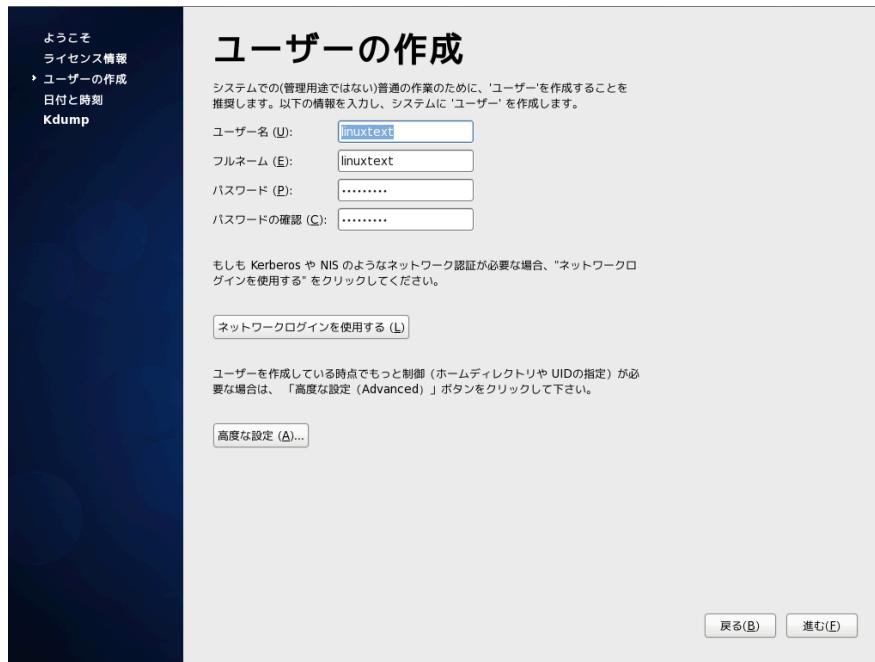
2. 起動プロセスが進み「ようこそ」の画面が現れます。右下の「進む」をクリックします。



3. 「ライセンス情報」の画面が現れます。「はい、ライセンス同意書に同意します」を選択して、右下の「進む」をクリックします。



4. ユーザーの作成の画面が現れます。ユーザ名やパスワードなど入力して、右下の「進む」をクリックします。

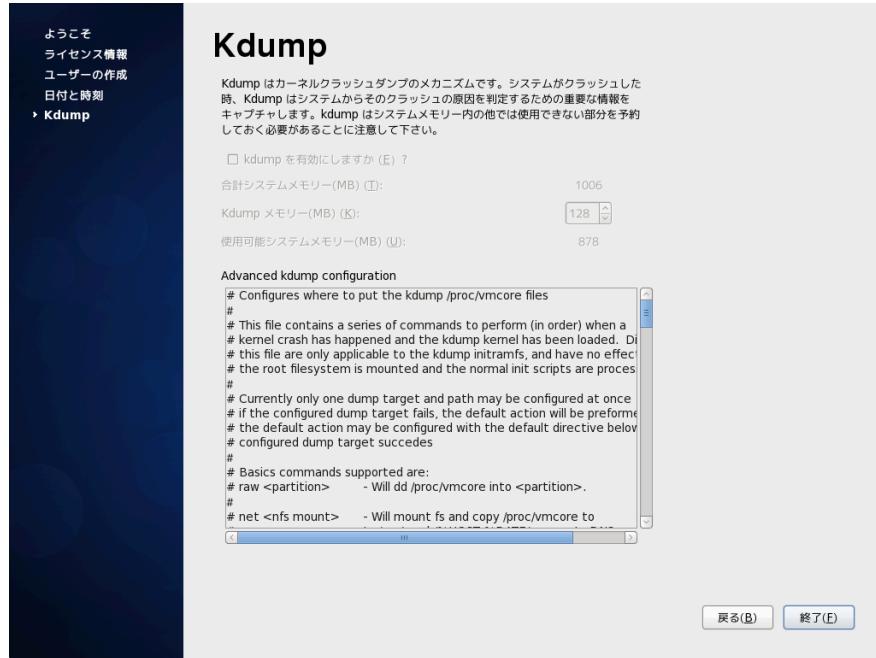


5. 日付と時刻の設定画面が現れます。日付と時刻を正しく設定し、右下の「進む」をクリックします。インターネットへの接続が使用できる場合には、「ネットワーク上で日付と時刻を同期化します」をチェックします。



2.5 インストール直後の初期設定

6. Kdump の設定画面が現れます。メモリが少ない場合には「kdump を設定するにはメモリーが足りません。」のダイアログが表示されるので、「OK」をクリックします。ここでは Kdump を設定せず、右下の「終了」をクリックします。

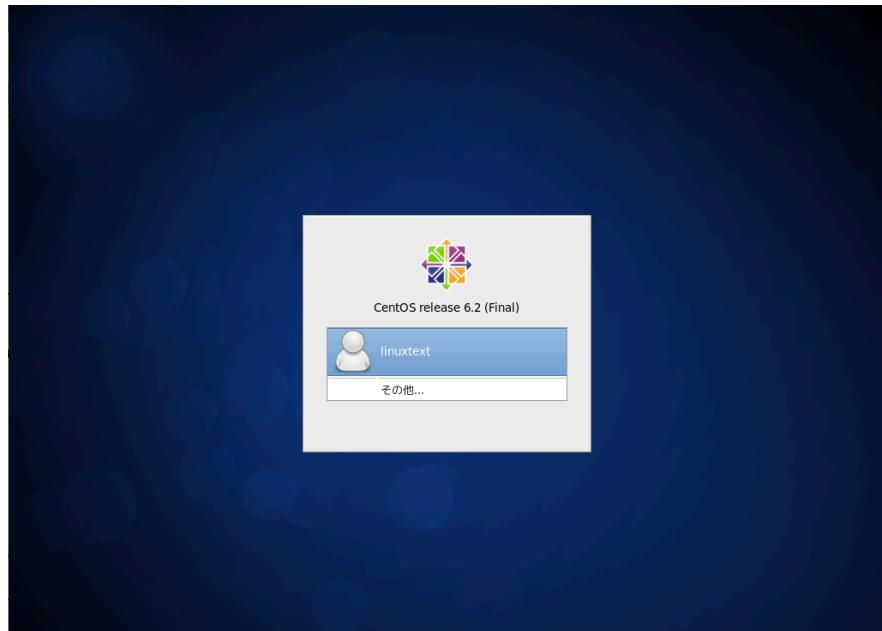


これでインストール直後の初期設定は終了です。

2.6 ログインする

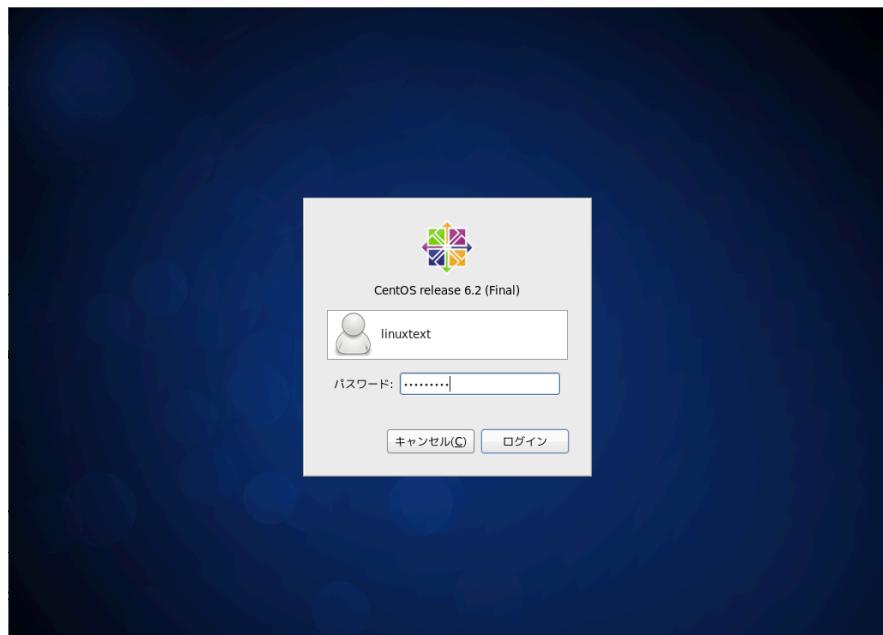
ログイン画面が表示されるので、インストール手順で作成したユーザ「linuxtext」でログインします。

1. ユーザ「linuxtext」をクリックします。

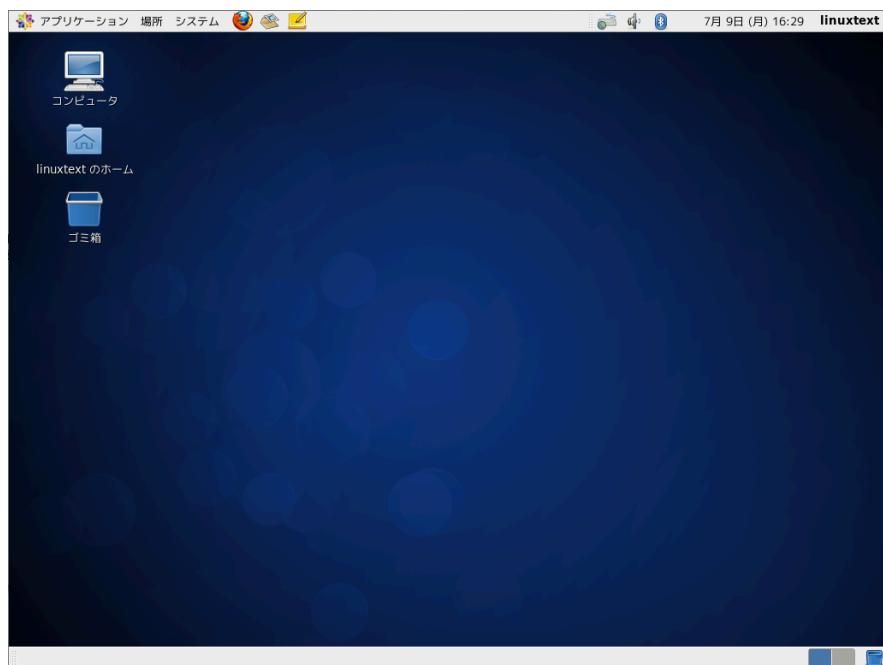


2. 「パスワード」にインストール時に設定したパスワードを入力し、「ログイン」をクリックします。

2.6 ログインする



3. CentOS にログインできました。

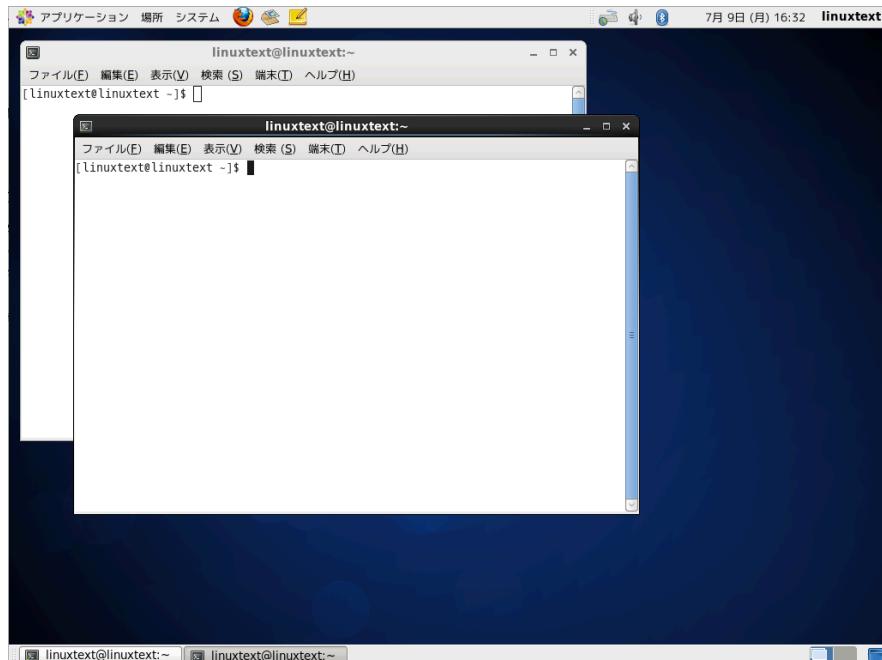


2.7 コマンドの実行

GUI でログインすると、Windows などと同じようにさまざまなグラフィカルなアプリケーションが利用できますが、「端末」を実行すると Linux のコマンドを実行できます。端末を起動するには以下の方法があります。

- 「アプリケーション」メニューから「システムツール」、「端末」を選択する
- デスクトップ上を右クリックし、ポップアップメニューから「端末の中に開く」を選択する

「端末」ウインドウは複数起動できるので、一方で操作をしながら一方でログを表示したり、ユーザーを切り替えて操作することもできます。



第3章

基本的なコマンド

Linux の操作の基本は、コマンドを入力して操作する方法です。この章では、ファイルとディレクトリの基本的な考え方から始まり、ファイルに関するコマンドや Linux を利用する上で必要な基本的なコマンドを学習します。

この章の内容

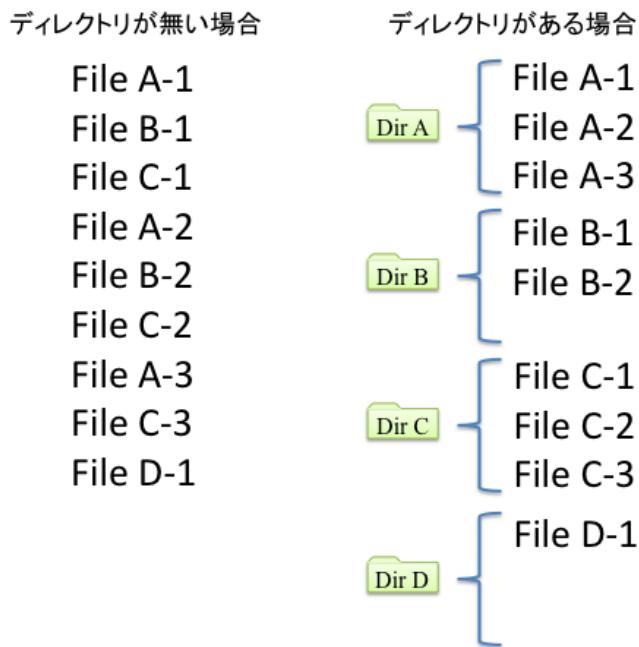
- 3.1 ファイル操作
 - 3.1.1 ファイルとディレクトリ
 - 3.1.2 ファイルやディレクトリの参照 (ls)
 - 3.1.3 ドットファイルの表示
 - 3.1.4 オプションの同時指定と順序
 - 3.1.5 ファイルのコピー (cp)
 - 3.1.6 ファイルの移動 (mv)
 - 3.1.7 ファイルの削除 (rm)
- 3.2 ディレクトリの操作 (pwd,cd,mkdir,rmdir)
 - 3.2.1 現在のディレクトリの表示 (pwd)
 - 3.2.2 ディレクトリの変更 (cd)
 - 3.2.3 ディレクトリの作成 (mkdir)
 - 3.2.4 ディレクトリの削除 (rmdir)
 - 3.2.5 特別なディレクトリ
 - 3.2.6 絶対 (パス) 指定と相対 (パス) 指定
- 3.3 ファイルの内容を表示
 - 3.3.1 ファイルの内容を表示 (cat)
 - 3.3.2 ページャを使った表示
- 3.4 ファイルの検索 (find)
- 3.5 コマンドのパス
 - 3.5.1 コマンドのパスの表示 (which)
- 3.6 ヘルプの使い方
- 3.7 マニュアルの使い方
 - 3.7.1 セクション
- 3.8 章末テスト

3.1 ファイル操作

本章では、ファイル操作を基本としたコマンドによる操作について学びます。CentOS 環境を手元に用意して、実際にコマンドを実行した場合の動きを確認してみましょう。

3.1.1 ファイルとディレクトリ

コンピュータの基本的な作業は、情報（＝データ）を処理することにあります。データをコンピュータ上に保存しておく方法として、ファイルという形式が用いられます。文書を作成したら文書ファイルを保存しますし、デジカメで写真を撮ったら、画像ファイルをパソコンに保存する、といった具合です。ファイルの数が増えると、内容や用途といった目的別に分ける必要が出てきます。このときに必要なものが、ディレクトリという考え方です。ディレクトリは、複数のファイルをまとめておくことができます。また、ディレクトリの中にディレクトリを作成することも可能です。



3.1.2 ファイルやディレクトリの参照 (ls)

ファイルを操作する上で、一番最初に必要になるコマンドは、その一覧を取得するコマンドです。ls(LiSt) が、そのコマンドにあたります。ls コマンドでは、ファイル名やディレクトリ名を指定して情報を取得することができます。ディレクトリ名を指定した場合、そのディレクトリの中にあるファイルやディレクトリ名の一覧を取得することができます。ディレクトリやファイルを ls コマンドで参照する場合、*と?をワイルドカードとして使うこともできます。

書式

```
ls [オプション] [ファイル]
```

オプション

- a
. で始まる隠しファイル等もすべて (All) 出力します。
- l
長 (Long) 形式で出力します。
- t
最終更新時間 (Time) によって、ソートをして出力します。
- r
逆順 (Reverse) にソートをして出力します。

○実習: ls コマンドの実行

それでは早速 Linux をコマンドで操作してみましょう。まずは現在いるディレクトリ内にあるファイルやディレクトリの一覧を表示するコマンドの ls コマンドを実行してみましょう。

```
$ ls
ダウンロード デスクトップ ビデオ 画像
テンプレート ドキュメント 音楽 公開
```

いくつかのディレクトリがあることがわかります。

○実習: 場所を指定したディレクトリー一覧の取得

ls コマンドの後にファイルやディレクトリを指定した場合、指定したディレクトリ内にあるファイルやディレクトリを一覧表示することができます。

3.1 ファイル操作

```
$ ls /usr/
bin  etc  games  include  lib  libexec  local  sbin  share  src  tmp
```

書式とは

書式とは、そのコマンドが用いられる形式を示しています。コマンド実行に必要な要素を、どのように列挙するかを示します。

オプションとは

コマンドはオプションをつけることで、その動作を変えることができます。

○実習: ls コマンドでワイルドカードを使った絞り込み

まずは/etc ディレクトリ上にあるファイルやディレクトリの一覧を表示してみましょう。

```
$ ls /etc/
ConsoleKit          mailcap
DIR_COLORS          makedev.d
(略)
```

ディレクトリ/etc には、多くのファイルやディレクトリが存在していることがわかります。この中から目的のファイルを探すのは大変そうです。そこでファイルを絞り込んでリスト表示してみましょう。

ls コマンドの結果から絞り込む場合、ワイルドカードを使うことができます。例えば、/etc ディレクトリ内の conf ファイルを絞り込んで表示するには、以下のように指定することができます。

```
$ cd /etc
$ ls *.conf
Trolltech.conf      nfsmount.conf
asound.conf         nsswitch.conf
autofs_ldap_auth.conf  ntp.conf
(略)
```

*は任意の文字列を示しており、この間に何文字が入っても検索結果として出力されます。そのため、上記のように多くの conf で終わるファイルがリスト表示されたはずです。

ファイル名の文字数がわかっている場合は?を使って絞り込むこともできます。

```
$ cd /etc
$ ls ???.conf
cas.conf  gai.conf  ntp.conf  sos.conf  yum.conf
```

?は一個につき何らかの一文字を示します。そのため、ファイル名部分が3文字のconfが検索結果として出力されます。

ワイルドカードによるファイルの絞り込みの方法について理解を深めるため、以下の例題を実践してみましょう。

hosts. のあとに何らかの名前のつくファイルを絞り込み

```
$ cd /etc  
$ ls hosts.*  
hosts.allow  hosts.deny
```

hosts. のあとに4文字含まれるファイルを絞り込み

```
$ cd /etc  
$ ls hosts.????  
hosts.deny
```

hosts. のあとに4文字 + 最後の文字がwであるファイルを絞り込み

```
$ cd /etc  
$ ls hosts.????w  
hosts.allow
```

○実習: ls -l コマンドの実行

-lオプションを付けてlsコマンドを実行した場合、情報が長形式で出力されます。

```
$ ls -l /home/linuxtext/  
合計 32  
drwxr-xr-x. 2 linuxtext linuxtext 4096 7月 6 10:05 2012 ダウンロード  
drwxr-xr-x. 2 linuxtext linuxtext 4096 7月 6 10:05 2012 テンプレート  
drwxr-xr-x. 2 linuxtext linuxtext 4096 7月 26 18:33 2012 デスクトップ  
drwxr-xr-x. 8 linuxtext linuxtext 4096 7月 27 14:07 2012 ドキュメント  
drwxr-xr-x. 2 linuxtext linuxtext 4096 7月 6 10:05 2012 ビデオ  
drwxr-xr-x. 2 linuxtext linuxtext 4096 7月 6 10:05 2012 音楽  
drwxr-xr-x. 2 linuxtext linuxtext 4096 7月 6 10:05 2012 画像  
drwxr-xr-x. 2 linuxtext linuxtext 4096 7月 6 10:05 2012 公開  
(略)
```

/home/linuxtextディレクトリ内について、詳細な情報が出てきました。これは、lsコマンドに対して-lをつけたからです。このような-lのことを「オプション」といいます。lsは、ファイルやディレクトリの一覧を表示するコマンドです。コマンドに対してオプションをつけることで、さま

3.1 ファイル操作

ざまな追加機能を指示することができます。

3.1.3 ドットファイルの表示

例えば、長形式で隠しファイルも含めて表示するときは、次のように入力します。

○実習: ls -la コマンドの実行

-la オプションを付けて ls コマンドを実行した場合、詳細な情報が表示できます。前の-l オプションと比べてどういった違いがあるか説明します。

```
$ ls -la /home/linuxtext
合計 196
drwx----- 31 linuxtext linuxtext 4096 7月 27 14:08 2012 .
drwxr-xr-x  4 root      root      4096 7月  9 10:43 2012 ..
-rw-----  1 linuxtext linuxtext 4960 7月 27 08:56 2012 .ICEauthority
drwxrwxr-x  2 linuxtext linuxtext 4096 7月 18 09:52 2012 .abrt
-rw-----  1 linuxtext linuxtext 3148 7月 26 19:30 2012 .bash_history
-rw-r--r--  1 linuxtext      500   18 12月  2 23:40 2011 .bash_logout
-rw-r--r--  1 linuxtext      500   176 12月  2 23:40 2011 .bash_profile
-rw-r--r--  1 linuxtext      500   124 12月  2 23:40 2011 .bashrc
(略)
drwxr-xr-x  2 linuxtext linuxtext 4096 7月   6 10:05 2012 ダウンロード
drwxr-xr-x  2 linuxtext linuxtext 4096 7月   6 10:05 2012 テンプレート
drwxr-xr-x  2 linuxtext linuxtext 4096 7月 26 18:33 2012 デスクトップ
drwxr-xr-x  8 linuxtext linuxtext 4096 7月 27 14:07 2012 ドキュメント
(略)
```

ls -l のときより多くのファイルが現れました。特に.（ドット）で始まるファイルが見られます。ドットで始まるファイルは隠しファイルで、システム設定が書かれているファイルなどで主に使われます。

3.1.4 オプションの同時指定と順序

オプションは、まとめてもまとめなくとも同じ結果になります。さらに、順不同なので入れ替えるても同じ結果になります。ただし、コマンドによっては順序と文法が細かく決まっている場合があるため、コマンドを実行する前にマニュアルやヘルプを参照してオプションの指定方法を確認してください。

実行例

```
$ ls -la
$ ls -al
$ ls -l -a
$ ls -a -l
```

コマンドオプションはハイフンを使って指定することが一般的ですが、コマンドによってはハイ

フンを必要としないコマンドもあるため注意が必要です。

○実習: ls -lt コマンドの実行

-lt オプションを付けて ls コマンドを実行した場合、ファイルの更新時刻順にソートしてリスト表示します。

```
$ ls -lt /home/linuxtext
合計 32
drwxr-xr-x. 8 linuxtext linuxtext 4096 7月 27 14:07 2012 ドキュメント
drwxr-xr-x. 2 linuxtext linuxtext 4096 7月 26 18:33 2012 デスクトップ
drwxr-xr-x. 2 linuxtext linuxtext 4096 7月 6 10:05 2012 ダウンロード
drwxr-xr-x. 2 linuxtext linuxtext 4096 7月 6 10:05 2012 テンプレート
drwxr-xr-x. 2 linuxtext linuxtext 4096 7月 6 10:05 2012 ビデオ
drwxr-xr-x. 2 linuxtext linuxtext 4096 7月 6 10:05 2012 音楽
drwxr-xr-x. 2 linuxtext linuxtext 4096 7月 6 10:05 2012 画像
drwxr-xr-x. 2 linuxtext linuxtext 4096 7月 6 10:05 2012 公開
(略)
```

ファイルの更新時刻が新しい順に並んでいるのがわかります。

3.1.5 ファイルのコピー (cp)

ファイルを扱う上で、複製を作成するなど、コピーが必要になることがあります。それを行なうのが cp(CoPy) コマンドです。cp コマンドでファイルを複製し、新しくできた複製ファイルのファイル名を指定することができます。cp コマンドは、次のような処理を行ないます。

書式

```
cp [オプション] コピー元 コピー先
```

実行例

```
$ cp src-file dst-file
```

ファイルを別名のファイルとしてコピー

src-file がファイル名で、dst-file が存在しないファイル名の場合、src-file のコピーとして dst-file が作成されます。

ファイルをディレクトリにコピー

src-file がファイル名で、dst-file がディレクトリ名の場合、dst-file の下にファイル src-file のコピーが作成されます。

ファイルを新しいファイルで上書き

src-file がファイル名で、dst-file が存在するファイル名の場合、ファイル dst-file はファイル src-file のコピーで上書きされます。

オプション

-i

処理を行なうときに確認をします。ファイルを上書きコピーするときなど、-i オプションを付けていると、コンピュータが確認のための問い合わせをしてきます。誤って上書きコピーすることを防ぐために利用されます。

-r

ディレクトリをコピーします。cp コマンドは基本的にファイルをコピーする機能のみですが、-r オプションを付けていると、ディレクトリの中にある全てのファイル・ディレクトリに対してコピーを行なうことができます。

-p

元ファイルの情報を保存します。ファイルには所有者・属性・更新日時など、ファイルに関するさまざまな情報があります。cp コマンドでコピーをすると、新しいファイルはそれらが全て新しい内容になってコピーされます。新しい内容を作成せず、古い情報を保持したままコピーを作成したい場合、この-p オプションを付けます。

○実習: cp コマンドの実行

cp コマンドを使って、実際にファイルをコピーしてみましょう。

- まずはオプションを付けないで cp コマンドを実行してみましょう。

```
$ cd ~  
$ cp /etc/hosts ~  
$ ls  
hosts  
(略)
```

hosts ファイルをコピーできました。

- 次にファイルを別名としてコピーしてみましょう。

```
$ cd ~  
$ cp /etc/hosts ~/hosts.newname  
$ ls  
hosts  hosts.newname  
(略)
```

hosts.newname という名前でコピーできました。

3. cp コマンドを-p オプションを付けて実行してみます。

```
$ cp -p hosts.newname hosts.sametime
$ ls -l
合計 44
-rw-r--r--. 1 tooyama tooyama 158 5月 31 11:18 2012 hosts
-rw-r--r--. 1 tooyama tooyama 158 5月 31 11:24 2012 hosts.newname
-rw-r--r--. 1 tooyama tooyama 158 5月 31 11:24 2012 hosts.sametime
(略)
```

タイムスタンプが同一のファイルとしてコピーできました。

4. cp コマンドを-r オプションを付けて実行してみます。ここでは2つのディレクトリを mkdir コマンド（後述）で作成して、ファイルをコピーしてみましょう。

```
$ mkdir olddir newdir
(olddir と newdir を作成)
$ cp -r olddir/ newdir/
$ ls newdir/
olddir
```

newdir ディレクトリに olddir ディレクトリをコピーすることができました。

3.1.6 ファイルの移動 (mv)

ファイルを移動 (MoVe) するときに利用するコマンドが、mv コマンドです。また、mv コマンドを使ってファイル名の変更も行えます。

書式

```
mv 移動元ファイル 移動先ファイル
```

オプション

-i

処理を行なうときに確認をします。ファイルを上書きするときなど、-i オプションを付けていると、コンピュータが確認のための問い合わせをしてきます。誤って上書きされることを防ぐために利用されます。

-f

強制的に処理を実行します。一部の処理では、mv コマンドが確認の問い合わせをします。その確認を無視し、強制的に処理を実行するためのオプションが-f です。

3.1 ファイル操作

この mv コマンドを実行することにより、”移動元ファイル”を”移動先ファイル”に移動することができます。次の場合を考えてみましょう。

実行例

```
$ mv src-file dst-file
```

ファイルを別のディレクトリに移動

src-file がファイル名で、dst-file が存在するディレクトリの場合、ファイル src-file がディレクトリ dst-file の下に移動します。

ディレクトリを別のディレクトリに移動

src-file がディレクトリ名で、dst-file が存在するディレクトリの場合、ディレクトリ src-file はディレクトリ dst-file の下に移動します。

ファイル名の変更

src-file がファイル名で、dst-file が存在しないファイル名の場合、ファイル dst-file にファイル名が変更されます。

ディレクトリ名の変更

src-file がディレクトリ名で、dst-file が存在しないディレクトリ名の場合、ディレクトリ dst-file にディレクトリ名が変更されます。

○実習: mv コマンドの実行

それでは早速、mv コマンドを使ってみましょう。

1. まず、ファイルの名前を変更してみます。

```
$ mv hosts.newname hosts.renew
$ ls -l
合計 44
-rw-r--r--. 1 tooyama tooyama 158 5月 31 11:18 2012 hosts
-rw-r--r--. 1 tooyama tooyama 158 5月 31 11:24 2012 hosts.renew
-rw-r--r--. 1 tooyama tooyama 158 5月 31 11:24 2012 hosts.sametime
(略)
```

hosts.newname が hosts.renew ファイルに名前変更されました。

2. ファイルを別のディレクトリに移動してみます。ここでは test ディレクトリを mkdir コマンド（後述）で作成して、ファイルを移動してみましょう。

```
$ mkdir test  
$ mv hosts.renew test/  
$ ls  
hosts hosts.sametime test  
(host.renew が移動されていることを確認)  
$ ls test/  
hosts.renew
```

test ディレクトリに host.renew を移動することができました。

3.1.7 ファイルの削除 (rm)

作成したファイルを削除するときに利用するコマンドが、rm(ReMove) コマンドです。rm コマンドは、削除したいファイルを指定します。

書式

```
rm ファイル名
```

rm コマンドのオプションは次のようにになっています。

オプション

-i
処理を行なうときの確認です。対象ファイルを本当に削除してよいか、確認のための問い合わせをします。誤って削除することを防ぐために利用されます。

-f
強制的に処理を実行します。ファイルによっては、削除に確認を求められる場合がありますが、-f オプションはその確認を無視して、強制的に処理を継続します。

-r
ディレクトリを対象にします。ディレクトリの中の、ファイルやディレクトリも削除します。

実行例

```
$ rm file-delete
```

file-delete というファイルを削除します。rm コマンドは、対象のファイル名を入力します。Linux を含む UNIX 系 OS では、一度削除したファイルを復活させることはできません。また、rm コマンドは、(オプション無しでは) ディレクトリを削除することはできません。ディレクトリの削除は、後述の、rmdir コマンドで説明します。

3.1 ファイル操作

○実習: rm コマンドの実行

rm コマンドを使って実際にファイルを消してみましょう。

```
$ cd ~  
$ rm hosts.sametime  
$ rm -r test  
$ ls -l  
合計 36  
-rw-r--r--. 1 tooyama tooyama 158 5月 31 11:18 hosts  
(略)
```

hosts.sametime と test ディレクトリが削除されました。

共通オプションについて

cp と rm に同じようなオプションがあったことに気がついたでしょうか？ コマンドオプションはいくつかありますが、その 1 つに-r があります。cp に-r オプションを付けた場合はディレクトリ階層全部コピーすることを示し、rm に-r オプションを付けた場合はディレクトリ階層の中をすべて削除します。-r は recursive (再帰) を示すオプションで、「ディレクトリの中のディレクトリの中の・・・」という階層構造の中の中までを意味します。

3.2 ディレクトリの操作 (pwd,cd,mkdir,rmdir)

次にディレクトリについて学習します。ディレクトリの操作に使われるコマンドとして、pwd、cd、mkdir、rmdirについて説明します。

3.2.1 現在のディレクトリの表示 (pwd)

ディレクトリは、階層構造になっています。その階層の中で、現在どの位置にいるか表示するのが、この pwd(Print Working Directory) です。次のように入力します。

書式

```
pwd
```

実行例

```
$ pwd  
/home/penguin
```

現在、このユーザは/home/penguin というディレクトリにいることがわかります。/は、ディレクトリの区切りを示す記号です。

3.2.2 ディレクトリの変更 (cd)

cd(Change Directory) コマンドを実行すると、現在いるディレクトリを変更することができます。cd の後にスペースを入力し、続けて移動先のディレクトリを指定します。ディレクトリを指定しない場合はホームディレクトリ(後述)に移動します。

書式

```
cd [ディレクトリ名]
```

○実習: ディレクトリの移動

それでは早速今回覚えたコマンドを実行してみましょう。pwdで現在いる場所を確認して、cdコマンドでディレクトリの移動を行ないます。

```
$ pwd  
/home/penguin  
$ cd /usr
```

3.2 ディレクトリの操作 (pwd,cd,mkdir,rmdir)

```
$ pwd  
/usr
```

ディレクトリの移動に成功しました。

3.2.3 ディレクトリの作成 (mkdir)

mkdir コマンドを実行すると、ディレクトリを作成できます。mkdir の後にスペースを入力し、続けて作成するディレクトリ名を指定します。

書式

```
mkdir ディレクトリ名
```

オプション

```
-p  
指定されたディレクトリの上位ディレクトリも作成する
```

ディレクトリ dir1 の下にディレクトリ dir2 を作り、さらにディレクトリ dir2 の下にディレクトリ dir3 を作りたいとき、次のように入力します。

実行例：オプションを付けない例

```
$ mkdir dir1  
$ mkdir dir1/dir2  
$ mkdir dir1/dir2/dir3
```

mkdir コマンドを 3 つに分けて実行しました。これは、ディレクトリを作るときはその上位層ができるいないと作成できない、という制約があるためです。-p オプションを付けて実行すると次のように一度に実行できます。

実行例：オプションを付けた例

```
$ mkdir -p dir1/dir2/dir3
```

○実習: mkdir -p コマンドの実行

mkdir を-p オプション付きで早速実行してみましょう。

```
$ mkdir -p dir4/dir5/dir6
$ ls -R dir4
dir4:
dir5
dir4/dir5:
dir6
dir4/dir5/dir6:
```

このコマンドの結果から、dir4 には dir5 ディレクトリが作られ、dir5 ディレクトリには dir6 ディレクトリが作られたことがわかります。また、dir6 ディレクトリの中には何もファイルやディレクトリがないということがわかります。

3.2.4 ディレクトリの削除 (rmdir)

ディレクトリを削除するコマンドとして rmdir コマンドがあります。

書式

```
rmdir ディレクトリ名
```

オプション

-p

指定した階層までのディレクトリを一括で削除します。オプションを付けないで rmdir コマンドを実行した場合は、最下層のディレクトリのみ削除します。ただしいずれの場合も、対象とするディレクトリ内は空でなければなりません。

(例)dict1(dict2 ディレクトリを一括で削除

```
rmdir -p dict1(dict2
```

rmdir の特徴は、「中が空であるときのみ」ディレクトリを削除できることです。ディレクトリの中にファイルが存在していると、ディレクトリは削除できません。

ディレクトリにファイルが存在する場合、ディレクトリを削除するには以下のように rm -r コマンドを用いて行なうことができます。

```
$ ls
directory
$ rmdir directory/
rmdir: failed to remove `directory/': ディレクトリは空ではありません
(directory の削除に失敗)
$ rm -r directory/
(rm -r を実行)
$ ls
(directory ディレクトリが削除された)
```

○実習: ディレクトリの削除

ディレクトリの削除に利用するコマンドの rmdir コマンドと rm コマンドを使ってみましょう。

```
$ mkdir directory1 directory2
(ディレクトリを作成)
$ touch directory2/file
(directory2 ディレクトリ内に file を作成)
$ ls
directory1  directory2
$ rmdir directory1
$ ls
directory2
(directory1 は空なので削除できる)
$ rmdir directory2
rmdir: failed to remove `directory2/': ディレクトリは空ではありません
(directory2 は空ではないため削除できない)
$ rm -r directory2/
$ ls
(directory2 も削除できた)
```

rmdir コマンドはディレクトリを削除するコマンドですが、削除対象のディレクトリは空である必要があります。

空ではないディレクトリを削除するには rmdir コマンドではなく、rm コマンドに-r オプションを付けて実行します。このコマンドを実行すると、特に警告されることなく指定したディレクトリ以下のディレクトリとファイルを削除することができます。

ただし、通常利用時は rmdir を使う癖をつけておくと、必要なディレクトリを誤って削除してしまうミスを軽減できるので安心です。

3.2.5 特別なディレクトリ

Linux では、ディレクトリの中で特別なディレクトリや、それを表す記号があります。以下がその代表的なものです。

カレントディレクトリ (.)

現在いるディレクトリのことです。ファイルやディレクトリ操作では「ドット」を使って表わします。

親ディレクトリ (..)

1 階層上のディレクトリのことです。カレントディレクトリが /home/penguin の場合、/home が親ディレクトリにあたります。

ホームディレクトリ (~)

ユーザの作業開始位置となるディレクトリです。ログイン直後のユーザは、必ずホームディレクトリにいます。ユーザごとに異なるホームディレクトリを使うので、それぞれのユーザにとってホームディレクトリは異なります。ファイルやディレクトリ操作では「~ チルダ」を使って表わします。

ルートディレクトリ (/)

ディレクトリ階層の、最上位階層を示します。ファイルやディレクトリ操作では「/ ルート」を使って表わします。

3.2.6 絶対(パス)指定と相対(パス)指定

カレントディレクトリが /usr/local であるとします。このとき、 /usr/bin/xxx のファイルを指定するには、次の 2通りが存在します。

1. /usr/bin/xxx
2. ../bin/xxx

1. は、最上位のディレクトリ (/) からディレクトリ・ファイル名を指定しています。2. は、「自分が現在いる位置からみて」ディレクトリ・ファイル名を指定しています。1. の方法を絶対(パス)指定といい、2. の方法を相対(パス)指定といいます。絶対(パス)指定は、どのディレクトリにいてもパスを指定できます。逆に、相対(パス)指定は、現在のディレクトリが変わると利用できない、という特徴があります。また、絶対(パス)指定は、常に / から指定するので非常に記述が長くなることがあります。相対(パス)指定は、現在の位置から見て非常に近いディレクトリのときは、指定の記述が短くて済む、という特徴があります。ここでは、penguin というユーザのホームディレクトリが、/home/penguin であるという想定をしています。

○実習：さまざまなディレクトリへの移動

それでは早速、ディレクトリからディレクトリへ、コマンドを実行して移動してみましょう。ここでは例として、penguin ユーザがいると仮定してコマンドを実行しています。

```
$ cd ~
$ pwd
/home/penguin
$ cd ..
$ pwd
/home
$ cd /usr
$ pwd
/usr
$ cd ~
$ pwd
/home/penguin
$ cd ../../
```

```
$ pwd  
/
```

3.3 ファイルの内容を表示

ファイルの内容を表示するコマンドについて説明します。

3.3.1 ファイルの内容を表示 (cat)

ファイルの内容を確認するコマンドの 1 つである cat コマンドを利用する方法について説明します。

書式

```
cat ファイル名  
ファイルの内容を表示することができます。
```

オプション

```
-n  
行番号を付加して表示します。
```

○実習: cat コマンドの実行

ホームディレクトリ内のファイル.bashrc を確認してみましょう。

```
$ cat ~/.bashrc  
# .bashrc  
# Source global definitions  
if [ -f /etc/bashrc ]; then  
(略)  
$ cat -n ~/.bashrc  
1 # .bashrc  
2  
3 # Source global definitions  
4 if [ -f /etc/bashrc ]; then  
(略)
```

cat で内容が確認でき、-n オプションを付けることで、行番号を表示できることが確認できました。

3.3.2 ページャを使った表示

cat コマンドを使ってファイルの内容を表示するとき、行数がたくさんあると表示が流れてしまいます。皆さんを利用するコマンドラインは、多くの場合 25 行に設定されているため、それ以上の行は流れてしまって確認できません。たくさんの行があっても、画面制御を行なってスクロールを途中で止めてくれる機能のことをページングといい、それを実現するコマンドをページャといいます。代表的なページャとしては more や less コマンドがあげられます。使い方は cat コマンドなどと同じで、オプションとしてページングする対象のファイル名をとります。

書式

```
more ファイル名
less ファイル名
```

ページャでファイルを開いた後は、次のようにコマンドを入力すると操作ができます。

表 3.1 more コマンドの場合

項目	内容
スペース	次のページに進む
b	前の画面に戻る
f	次の画面に進む
/単語	単語を検索します。n キーで検索結果をジャンプします
q	ページヤコマンドを終了 (quit) します。

表 3.2 less コマンドの場合

項目	内容
スペース	次のページに進む
b	前の画面に戻る
f	次の画面に進む
↑	前の行に進む
↓	次の行に進む
/単語	単語を検索します。n キーで検索結果をジャンプします
q	ページヤコマンドを終了 (quit) します。

3.4 ファイルの検索 (find)

○実習: ページャを使ったファイル閲覧

ページャを使って、複数行の記述があるファイルの確認を行なってみます。

```
$ more /etc/rc
```

さまざまなコマンド入力後 q を入力して終了させてください。

```
$ less /etc/rc
```

さまざまなコマンド入力後 q を入力して終了させてください。

3.4 ファイルの検索 (find)

ファイルがどこのディレクトリに存在するか、find コマンドにて検索できます。

書式

```
find パス -name ファイル名
```

○実習: ファイルの検索

/etc 配下にある hosts という名前のファイルを検索してみましょう。

```
$ find /etc/ -name hosts
find: `/etc/pki/CA/private': 許可がありません
find: `/etc/pki/rsyslog': 許可がありません
find: `/etc/ntp/crypto': 許可がありません
/etc/sysconfig/networking/profiles/default/hosts
(略)
/etc/hosts
/etc/avahi/hosts
(略)
```

/etc 配下には、一般ユーザには読み込み権限が与えられてないディレクトリが多くあります。find コマンドはパスで指定した配下の全てのディレクトリへの検索を試みますので、必要であれば適切な権限を持ったユーザへ切り替えて実行してください。

3.5 コマンドのパス

コマンドの実体はプログラムです。プログラムもファイルの一種であり、/bin や/sbin といったプログラム用のディレクトリに配置されています。

3.5.1 コマンドのパスを表示 (which)

基本的なコマンドを実行するとき、その実体であるプログラムがどのディレクトリに配置されているかを意識する必要はありません。これは、PATH という環境変数にプログラムが配置されているディレクトリが設定されているからです。which というコマンドを用いると、PATH 環境変数に含まれるディレクトリ配下に配置されているコマンドのパスを表示することができます。

書式

```
which コマンド名
```

○実習: which コマンドの実行

コマンドのパスを調べるために which コマンドを使います。ここでは例として cat コマンドのパスを確認してみます。

```
$ echo $PATH  
/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:/home/linuxtext/bin  
(ユーザが参照できるパスを確認)  
  
$ which cat  
/bin/cat  
(cat コマンドは/bin 配下に配置されているのがわかる)
```

コマンドが存在するディレクトリが PATH 環境変数に含まれていないと、which コマンドの結果はエラーとなります。例えば管理者権限が必要なコマンドへのパスは一般ユーザでは設定されていないため（設定されているディストリビューションもあります）、which コマンドでは確認できないことがあります。

3.6 ヘルプの使い方

コマンドにはヘルプが含まれていることがあります。実行したいコマンドに対して--help オプションなどをつけて実行することで、コマンドの実行に使えるオプションを調べることができます。

3.7 マニュアルの使い方

書式

```
コマンド --help
```

実行例

```
$ ls --help
使用法: ls [オプション]... [ファイル]...
List information about the FILEs (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort.

長いオプションに必須の引数は短いオプションにも必須です.
-a, --all           do not ignore entries starting with .
-A, --almost-all   do not list implied . and ..
(略)
```

3.7 マニュアルの使い方

Linux には、便利なオンラインマニュアルが含まれています。ここではその使い方を紹介します。

書式

```
man コマンド名
```

オプション

```
-k 単語
'単語'が含まれるエントリー一覧を出力します。
```

では、例として ls コマンドのマニュアルを調べてみましょう。

実行例

```
$ man ls
LS(1)                                         LS(1)
名前
ls, dir, vdir - ディレクトリの中身をリスト表示する
書式
ls [OPTION]... [FILE]...
(略)

-1      ファイルのモード・リンクの数・所有者名・グループ名・(バイト単位
```

の) サイズ・タイムスタンプ・名前を (1列形式で) 書き出す。デフォルトでは、表示されるタイムスタンプは最終修正時刻である。オプション `-c` と `-u` のときは、他の 2 つのタイムスタンプを選択する。デバイススペシャルファイルの場合、通例として、サイズを表示する場所がデバイスのメジャー番号とマイナー番号に置き換えられる。

(略)

`ls` コマンドがディレクトリの内容を表示する機能があるということがわかります。また、`-l` オプションにより、ファイルに関するさまざまな情報が表示されることもわかります。そのほか、以下の項目があります。

説明

このプログラムに関する説明です。

POSIX/GNU オプション

プログラムで利用可能なオプションです。指定したオプションによってプログラムの動作が変わります。

関連項目

このプログラムに関連しているコマンドや、他の機能です。ここに列挙している内容を `man` コマンドで調べることができます。

注意

このプログラムに関する注意事項やバグの報告先などの情報が書かれています。

その他、コマンドによりマニュアルにさまざまな項目が表示されます。

○実習: `man` コマンドの実行

`man` コマンドはそのあと指定するコマンドに対するマニュアルを表示するためのコマンドです。ここでは例として `cp` コマンドのマニュアルを見てみます。

```
$ man cp
```

`man` コマンドの表示は、ページヤ機能が利用されます。従ってスクロール等は `less` コマンドの操作と同じになります。

3.7.1 セクション

`ls` のマニュアルで、「LS(1)」という記述があります。これは、`ls` のマニュアルがセクション 1 にある、という意味です。マニュアルのセクションとは、マニュアルの内容を分野毎に分け、その分野を指定したものです。セクションには番号がついており、以下の表のようになっています。

表 3.3 マニュアルのセクション

項目	内容
1	ユーザコマンド
2	システムコール
3	システムライブラリや関数
4	デバイスやデバイスドライバ
5	ファイルの形式
6	ゲームやデモなど
7	その他
8	システム管理系のコマンド
9	カーネルなどの情報

○実習: セクション別のマニュアルの表示

内容が複数のマニュアルに分かれていることがあります。今回は passwd コマンドのマニュアルを開き、別セクションのドキュメントを開いてみましょう。

```
$ man passwd
(略)
関連項目
group(5), passwd(5), shadow(5).
```

マニュアルを読み進めていくと、「関連項目」というものが見つかり、passwd(5) というマニュアルがあることが記述されています。これは「セクション 5 にやはり passwd というエントリがあるので、それも参照できますよ」という意味があります。q で man コマンドを終了させ、セクション 5 のマニュアルを参照します。セクション 5 の passwd のエントリを見るには次のように入力します。

```
$ man 5 passwd
PASSWD(5)           File Formats and Conversions          PASSWD(5)
NAME
passwd - the password file
(略)
```

と出力されます。これは、/etc/passwd というファイルについての説明です。/etc/passwd には、ログイン名とパスワードの情報が記述されていますので、それについてのマニュアルが表示された、ということになります。このように、同じエントリでも複数のマニュアルが対応する場合、セクションを切り替えて参照することができます。

3.8 3章 章末テスト

(1) ファイルやディレクトリを参照するコマンドを記述しなさい。

(2)/media/cdrom/の中身を最新更新日順に並べて表示するコマンドを選びなさい。

1. ls -lf /media/cdrom/
2. ls -la /media/cdrom/
3. ls -lt /media/cdrom/
4. ls -l /media/cdrom/

(3) 今いるパスを表示するコマンドを記述しなさい。

(4)/home/user フォルダにある test というファイルを/root/backup フォルダに移動します。以下の空欄に当てはまるコマンドを書きなさい。

```
# (          ) /home/user/test /root/backup
# (          ) /root/backup
合計 4
-rw-r--r--. 1 root root 19 6月 14 11:14 2012 test
```

(5) パス/etc の中のファイルで、ファイル名の一部に「resolv」という文字列が含まれるファイルを検索するコマンドを記述しなさい。

第4章

正規表現とパイプ

Linux を理解するために、重要な知識がこの正規表現とパイプです。正規表現は Linux のさまざまな処理で使われることが多いですので、しっかり理解してください。

この章の内容

- 4.1 標準入出力
- 4.2 リダイレクト
 - 4.2.1 出力のリダイレクト
 - 4.2.2 cat コマンドによるファイル作成
- 4.3 標準エラー出力
- 4.4 パイプ
- 4.5 grep コマンド
 - 4.5.1 正規表現
 - 4.5.2 さまざまな条件を用いた grep コマンドの実行
 - 4.5.3 標準出力にマッチさせる
- 4.6 章末テスト

4.1 標準入出力

前章では、さまざまなコマンドを利用しました。それぞれのコマンドはひとつのプログラムです。Linux のプログラムには、「1 つの入り口と 2 つの出口」があります。それを、それぞれ標準入力・標準出力・標準エラー出力といいます。

標準入力はプログラムに入ってくるデータのことを示します。標準入力先は一般的にはキーボードになっています。標準出力はプログラムの実行結果を書き出す先のことを示します。標準出力は一般的にはプログラムを実行した端末のディスプレイになっています。標準エラー出力はエラーメッセージを書き出す先のことを示します。標準エラー出力は一般的にはプログラムを実行した端末のディスプレイになっています。

例えば ls とコマンドを打った場合、カレントディレクトリのファイルとディクトリの一覧が画面上に表示されます。このようにコマンドを実行した結果が画面上に表示されることを「標準出力に出力された」と表現します。



4.2 リダイレクト

コンソールに標準出力された文字列はリダイレクトを使ってファイルに書き込むことができます。リダイレクトは>を使って表します。

4.2.1 出力のリダイレクト

次のコマンドは ls の結果を出力をリダイレクトして、ファイルに書き込む例です。>の記号を用いることでリダイレクトができます。

実行例

```
$ ls > ls-output
```

コマンドを実行すると ls-output というファイルが作成されます。cat コマンドで、ファイル ls-output の中身を確認してください。ls を実行したときと同じ内容が含まれています。このようにリダイレクトを用いると、出力先を変更しファイルに出力を格納することができます。

○実習: ファイルへのリダイレクト

実際に ls の結果を ls-output に出力してみましょう。

```
$ cd /etc  
$ ls > ~/ls-output  
$ cat ~/ls-output
```

ls-output というファイルが作成されてその中に ls の出力結果が保存されました。すでに ls-output が存在しているときは、前の ls-output が削除されて新しい ls-output が作成されます。上書きせずに追記したい場合は、アペンド (>>) の記号を用います。

4.2.2 cat コマンドによるファイル作成

前項では、標準出力をリダイレクトすることでファイルを作成しました。cat コマンドとリダイレクトを組み合わせると、自由な内容でファイルを作成することができます。

○実習: cat コマンドによるファイル作成

前の説明でファイルの内容表示に使った cat コマンドでもリダイレクトを使うことでファイルの作成を行なうことができます。早速実行してみましょう。

4.2 リダイレクト

```
$ cat > cat-output
Hello
This is cat redirect.
(「Ctrl」Dを押す)
$
```

「Ctrl」D は EOF(End Of File) を示すキーで、データ入力の終わりを示します。Linux ではデータの読み込みが最後になると、この EOF が入力され終わる決まりがあります。cat は EOF を受け取ったことで入力が終わったと判断し、リダイレクトを終了します。

4.3 標準エラー出力

コンピュータを操作しているとさまざまなエラーが起きることがあります。エラーは画面とログに出力されます。

例えば以下のコマンドを実行する場合を考えてみましょう。実行例では tekitou というディレクトリの内容を ls-l-output というファイルに出力しようとしています。

実行例

```
$ ls -l tekitou > ls-l-output
(tekitou というディレクトリ内の詳細出力を ls-l-output にリダイレクト)
ls: cannot access tekitou: そのようなファイルやディレクトリはありません
(tekitou というディレクトリが見つからなかったというエラーメッセージ)
```

しかし、tekitou というディレクトリが存在しなかった場合、「そのようなファイルやディレクトリはありません」といったエラーメッセージが標準出力されます。

このようにエラーメッセージは通常は標準出力されますが、リダイレクトを用いて任意のファイルにファイル出力することもできます。

実行例

```
$ ls -l tekitou 2> ls-l-output
(エラー出力を ls-l-output にリダイレクト)
```

この場合、エラーメッセージは画面に現れず、指定したファイルに出力されます。コマンド中で指定している"2"は、標準エラー出力を示しています。標準出力は"1"を指定します。標準出力と標準エラー出力を 1 つのファイルに出力したい場合、次のように入力します。

実行例

```
$ ls -l tekitou > ls-l-output-second 2>&1
```

標準出力とエラー出力を ls-l-output-second にリダイレクトしました。ファイルの指定と、1,2 の指定の位置を間違えないようにしてください。

4.4 パイプ

標準入力からデータを入れることにより、特別な処理ができるコマンドがあります。次のコマンドを入力してみましょう。

実行例

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

4.4 パイプ

画面にファイル一覧が流れますが /usr/bin にはコマンドが 1000 以上あるため、そのままではすべてのコマンドを画面に表示しきることができません。ここで、次のように入力します。

実行例

```
$ ls -l /usr/bin | less
```

less は前章で学習したページャの less コマンドのことです。コマンドとコマンドを「| (パイプ)」でつなげることでパイプの前のコマンドを後ろのコマンドの標準入力とすることができます。

GNOMEなどのデスクトップ環境がインストールされている場合は、端末上でカーソルキーを押すことでページングすることができるため、操作する端末が目の前にある場合は、ページャにパイプで繋ぐ必要はありませんかもしれません。しかし端末をリモート操作している場合やデスクトップ環境がインストールされていない Linux 環境の場合は、パイプでつなげる方法を知っておくと大変便利です。



標準エラー出力で説明した、標準出力と標準エラー出力を 1 つのファイルに出力する場合の方法も利用可能です。次のように入力してください。

実行例

```
$ ls -l xxx /usr/bin 2>&1 | less
```

xxx は存在しませんが、その旨のメッセージはリダイレクトにより標準出力に出力されます。

○実習: コマンドの実行結果を less コマンドでページング表示

/usr/bin のファイル一覧を less でページングしながら表示します。

```
$ ls -l /usr/bin | less
:
:
:
(q を入力して終了)
```

4.5 grep コマンド

grep コマンドは、ファイルの中からデータを検索します。また、| grep とすることで、標準入力から入ったデータに対し検索を行なうことも可能です。

書式

```
grep [オプション] 検索条件 [指定ファイル]
```

指定ファイルの部分は、1つでもかまいませんし、*のようなワイルドカードを利用した複数ファイルの指定でも可能です。ワイルドカードとは、不特定の文字列を表現するのに利用される記号のことです。検索条件として、正規表現が用いられます。正規表現は、grep のみならず Linux で用いられるパターンマッチや、多くのプログラミング言語でも利用されている、非常に重要な機能です。

4.5.1 正規表現

正規表現は文字列のみならず、意味のある記号を用いることで、高度な検索条件を与える表現方法です。主に用いられる記号は次の通りです。

表 4.1 正規表現で使われる記号と意味

記号	意味
~	行頭を表す
\$	行末を表す
.	任意の一字を意味する
*	直前文字の0回以上の繰り返しを意味する
[...]	..の中の任意の一字を意味する
[^...]	..の文字が含まれないことを意味する。
¥	正規表現の記号をエスケープする。

表 4.2 正規表現の利用例

記号	意味
~a	aで始まっている行
b\$	bで終わっている行
a.b	aとbの間に1文字入っている
[ab]ab	aもしくはbに続くab(aab,bab)
[^ab]ab	aもしくはbで始まらない(not)で、abが続くもの (例: xab, zab等)

4.5 grep コマンド

○実習: grep コマンドの実行

grep コマンドを用いて、文字列の検索を行ないましょう。

/etc ディレクトリにあるファイルで abc という文字列を含むものを検索するには、以下のようにコマンドを実行します。

```
$ grep abc /etc/*
:
/etc/services:abcvoice-port      3781/tcp
```

/etc ディレクトリにあるファイルで行の先頭が xy ではじまっているものを検索するには、以下のようにコマンドを実行します。

```
$ grep ^xy /etc/*
:
/etc/services:xyplex-mux      173/tcp
:
```

4.5.2 さまざまな条件を用いた grep コマンドの実行

大文字と小文字の区別に関する条件を指定したり、否定の条件を指定することができます。

オプション

```
-e 文字列を検索パターンとして扱う。
-i 検索パターンと入力ファイルの双方で、英大文字と小文字の区別を行わない。
-v 検索パターンとマッチしなかった行を選択する。
```

○実習: さまざまな条件を用いた grep コマンドの実行

grep コマンドにオプションを付けて文字列を検索してみましょう。/etc ディレクトリにあるファイルについて、さまざまな条件で検索してみます。

```
$ grep abc /etc/*
(/etc ディレクトリにある、 abc という文字列を含むファイル)
(略)
/etc/services:hostname      101/tcp      hostnames      # usually from sri-nic
/etc/services:hostname      101/udp      hostnames      # usually from sri-nic
(略)
```

```
$ grep -i hostname /etc/*
(/etc ディレクトリにある、大文字または小文字の hostname という文字列を含むファイル)

/etc/rc.sysinit:      HOSTNAME=localhost
(略)
/etc/rc.sysinit:      # Reset the hostname.
(略)

$ grep abc /etc/* | grep -v tcp
(/etc ディレクトリにある、 abc という文字列を含むが tcp という文字列は含まないファイル)
(略)
/etc/services:abcvoice-port  3781/udp          # ABCvoice server port
(略)
```

先ほど表示された行 (/etc/services:abcvoice-port 3781/tcp) が、tcp という除外の文字列を含むため、今回は表示されません。

4.5.3 標準出力にマッチさせる

grep コマンドはファイルの内容に一致させる以外に、標準入力からの入力をマッチさせることができます。

○実習: 標準出力の結果を grep で検索

標準出力の結果を grep で検索してみましょう。以下のコマンドを早速実行してみましょう。

/usr/bin にあるコマンド一覧を出力します。今回の例では grep -e d\$と指定したため、ファイル名が d で終わるファイルを /usr/bin から検索しています。

```
$ ls /usr/bin/ | grep -e d$
:
:
xxd
yppasswd
$
```

4.6 4章 章末テスト

(1)grep コマンドを利用し、行頭が a または b で始まる行を抽出できる正規表現を以下の 4 つの中から選びなさい

1. grep -e ^(ab)
2. grep -e ^"ab"
3. grep -e ^{ab}
4. grep -e ^[ab]

(2) 「ls /usr/bin | grep -e ^a..\$」 というコマンドを実行した場合、どのような結果になるか答えなさい。

(3) パス/etc の中にあるファイルでファイル名が conf で終わるものを grep とパイプを用いて記述しなさい。

(4)ls -l を実行した結果を ls-result ファイルに書き込む場合のコマンドを記述しなさい。

(5)chkconfig --list の実行結果から iptables だけを絞り込んで表示するコマンドを grep を使って記述しなさい。

第5章

基本的なコマンド 2

本章では、基本的なコマンドとして touch, head, tail, sort, uniq, tr, diff を学習します。主にテキストファイルを処理するために利用すると便利なコマンドです。

この章の内容

- 5.1 ファイルのタイムスタンプの変更 (touch)
- 5.2 ファイルの一部の取得 (head, tail)
 - 5.2.1 両者共通の形式
 - 5.2.2 特別なオプション -f
- 5.3 ファイルのソート (sort)
 - 5.3.1 ファイルの準備
 - 5.3.2 sort の実行
 - 5.3.3 n 列目のデータソート (-k)
 - 5.3.4 数値式でのソート
- 5.4 行の重複の消去 (uniq)
- 5.5 文字列の置き換え (tr)
- 5.6 ファイルの比較 (diff)
- 5.7 章末テスト

5.1 ファイルのタイムスタンプの変更 (touch)

ファイルには、タイムスタンプ（最終更新日）が必ず存在します。タイムスタンプはls コマンドの-l オプションを付けることで確認できます。その最終更新時間を変更するのが、touch コマンドです。

書式

```
touch [オプション] ファイル名
```

touch コマンドを実行すると、ファイルのタイムスタンプが現在日時に変更されます。なお、オプションにより、新しいタイムスタンプとなる日時を指定することもできます。ファイルが存在しない場合、touch コマンドは中身が空である 0 バイトのファイルを作成します。

○実習: ファイルのタイムスタンプの変更と確認

テスト用のファイルを作成して、作成したファイルのタイムスタンプを変更してみましょう。早速以下のコマンドを実行してください。

```
$ ls -l hosts.bak
-rw-rw-r-- 1 okada okada 187 Jun 4 10:06 hosts.bak
$ touch hosts.bak
$ ls -l hosts.bak
-rw-rw-r-- 1 okada okada 187 Jun 5 09:50 hosts.bak
$ touch -t 06030800 hosts.bak
$ ls -l hosts.bak
-rw-rw-r-- 1 okada okada 187 Jun 3 08:00 hosts.bak
```

-t はファイルの更新時刻を指定するオプションです。時間は [[CC]YY]MMDDhhmm[.SS] 形式（[] 内は省略可）で指定します。これで、ファイルの最終更新時刻が変わったことが確認できます。

○実習: touch コマンドによるファイル作成

touch コマンドはタイムスタンプの変更をするだけのコマンドではありません。touch コマンドで、指定したファイル名のファイルが存在しない場合、空のファイルを作成します。

```
$ ls -l touched-file
$ touch touched-file
$ ls -l touched-file
-rw-rw-r-- 1 okada okada 0 Jun 5 09:55 touched-file
```

ファイルが作成されました。ファイルの中身は空なので、サイズが 0 バイトになっています。

5.2 ファイルの一部の取得 (head, tail)

ファイルの先頭や末尾など一部分のみを見る場合は head コマンドや tail コマンドが使えます。これらのコマンドを実行することでファイルの内容の一部を見ることができます。

5.2.1 head

書式

```
head [オプション] ファイル名
```

オプション

```
-n 行  
先頭から指定した行を標準出力します。  
-c バイト  
先頭から指定したバイト分を標準出力します。
```

head はファイルの先頭部分を標準出力します。オプションを付けない場合は、先頭から 10 行を標準出力します。

ファイル名には対象のファイル名を入力します。ファイル名の部分を空白にした場合、もしくは-（ハイフン）を指定した場合は、標準入力からのデータに対して処理を行ないます。従って次のコマンドを実行した場合、3 つのコマンドの実行結果は同一になります。

実行例

```
$ head FILE      FILE の先頭 10 行を表示  
$ cat FILE | head - 標準入力からの内容のうち先頭 10 行表示  
$ cat FILE | head 標準入力からの内容のうち先頭 10 行表示
```

3 つとも、FILE というファイルの先頭 10 行を表示します。コマンドを実行時に行数を指定しなかった場合は先頭 10 行が表示されます。オプションを付けることにより表示する行数を変更できます。

5.2.2 tail

書式

```
tail [オプション] ファイル名
```

5.2 ファイルの一部の取得 (head, tail)

オプション

-n 行

末尾から指定した行を標準出力します。

-c バイト

末尾から指定したバイト分を標準出力します。

tail はファイルの終わり部分を標準出力します。オプションを付けない場合は、末尾から 10 行を標準出力します。

ファイル名には対象のファイル名を入力します。ファイル名の部分を空白にした場合、もしくは- (ハイフン) を指定した場合は、標準入力からのデータに対して処理を行ないます。従って次のコマンドを実行した場合、3 つのコマンドの実行結果は同一になります。

実行例

```
$ tail FILE          FILE の末尾 10 行を表示
$ cat FILE | tail - 標準入力からの内容のうち末尾 10 行表示
$ cat FILE | tail   標準入力からの内容のうち末尾 10 行表示
```

3 つとも、FILE というファイルの末尾 10 行を表示します。コマンドを実行時に行数を指定しなかった場合は末尾 10 行が表示されます。オプションを付けることにより表示する行数を変更できます。

5.2.3 特別なオプション -f

tail は本来ファイルの終わり部分を表示するコマンドです。しかし、ファイルによっては、その終わり部分が隨時変わることがあります。tail には-f というオプションが存在します。-f オプションを付けることにより、変更をリアルタイムでモニタすることができます。よく用いられる目的に、ログファイルのモニタがあげられます。ログとは、サービスの動作状況が outputされるファイルです。システムの変更やサービスの動作などでメッセージがログに記載されると、その内容が出力されます。

書式

```
tail -f ファイル名
```

リアルタイムでログファイルなどの変更を見ることが可能です。

○実習: tail コマンドの実行

実際に tail コマンドと-f オプションの働きを確認してみましょう。tail コマンドの動きを確認するため、あらかじめ以下のコマンドを入力してテストで使うためのコマンドを準備してください。

```
$ man less > ~/manual-less
```

テスト用のファイルができたら、このファイルに対して tail コマンドを実行してみましょう。

```
$ tail manual-less
```

You may distribute under the terms of the Less License.

[参考訳]

Copyright (c) 1994-2000 Kazushi (Jam) Marukawa, 日本語化ルーチンのみ。

この部分に関するコメントは jam@pobox.com へ送って下さい。

このパッチは Less ライセンスの下で配布できる。

Version 358: 08 Jul 2000

LESS(1)

manual-less ファイルの末尾 10 行が標準出力されます。

○実習: tail -n コマンドの実行

tail コマンドは-n オプションを付けて標準出力する行数を指定することができます。以下の例は末尾 5 行を標準出力する例です。

```
$ tail -n 5 manual-less
```

このパッチは Less ライセンスの下で配布できる。

Version 358: 08 Jul 2000

LESS(1)

○実習: tail -f コマンドの実行

最後に-f オプションを付けてファイルを開いてみましょう。

1. 作成した manual-less を tail -f コマンドで開きます。

```
$ tail -f manual-less
```

(略)

Version 358: 08 Jul 2000

LESS(1)

5.2 ファイルの一部の取得 (head, tail)

2. tail コマンドに-f オプションを付けたため、ファイルの終わり部分が標準出力されますが、シェルに制御が戻りません。

```
linuxtext@linuxtext:~
```

only
Comments about this part to: jam@pobox.com
You may distribute under the terms of the Less License.

[参考訳]
Copyright (c) 1994-2000 Kazushi (Jam) Marukawa, 日本語化ルーチンのみ。
この部分に関するコメントは jam@pobox.com へ送って下さい。
このパッチは Less ライセンスの下で配布できる。

Version 358: 08 Jul 2000 LESS(1)
[linuxtext@linuxtext ~]\$ tail -f manual-less
You may distribute under the terms of the Less License.

[参考訳]
Copyright (c) 1994-2000 Kazushi (Jam) Marukawa, 日本語化ルーチンのみ。
この部分に関するコメントは jam@pobox.com へ送って下さい。
このパッチは Less ライセンスの下で配布できる。

Version 358: 08 Jul 2000 LESS(1)

3. 「アプリケーション→システム→端末」をクリックして端末をもう一つ開いてください。

```
linuxtext@linuxtext:~
```

only
Comments about this part to: jam@pobox.com
You may distribute under the terms of the Less License.

[参考訳]
Copyright (c) 1994-2000 Kazushi (Jam) Marukawa, 日本語化ルーチンのみ。
この部分に関するコメントは jam@pobox.com へ送って下さい。
このパッチは Less ライセンスの下で配布できる。

Version 358: 08 Jul 2000 LESS(1)

```
linuxtext@linuxtext:~
```

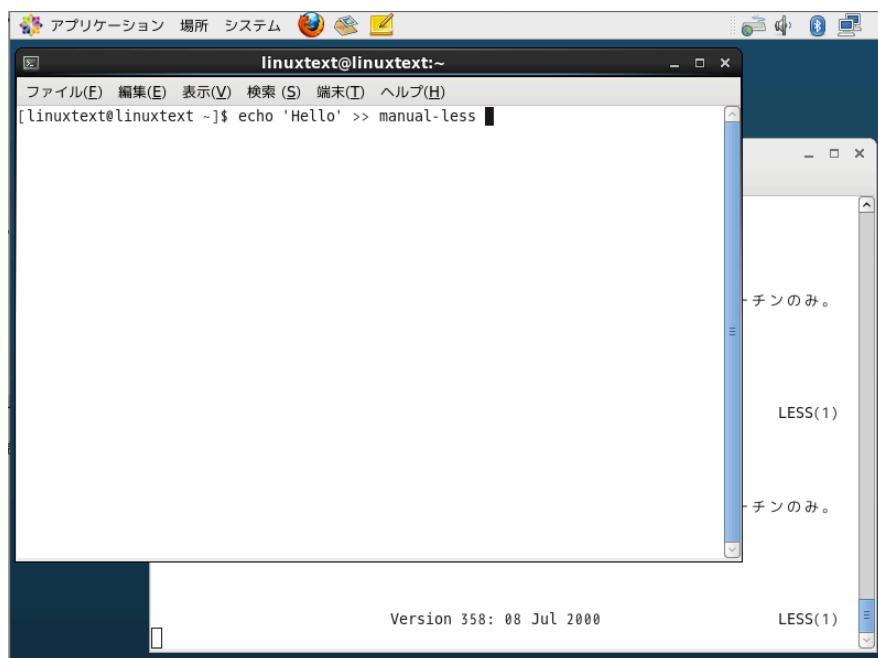
only
Comments about this part to: jam@pobox.com
You may distribute under the terms of the Less License.

[参考訳]
Copyright (c) 1994-2000 Kazushi (Jam) Marukawa, 日本語化ルーチンのみ。
この部分に関するコメントは jam@pobox.com へ送って下さい。
このパッチは Less ライセンスの下で配布できる。

Version 358: 08 Jul 2000 LESS(1)

4. 新しい端末で以下のようにコマンドを実行します。

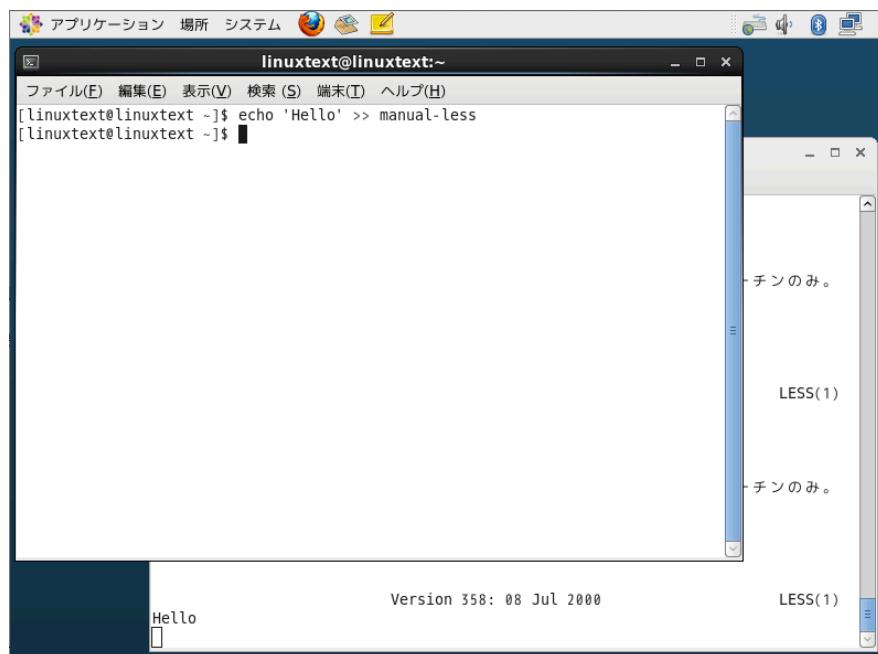
```
$ echo 'Hello' >> manual-less
```



5. manual-less ファイルにデータが追加されます。

```
$ tail -f manual-less  
(略)  
Version 358: 08 Jul 2000  
Hello  
LESS(1)
```

5.2 ファイルの一部の取得 (head, tail)



新しい端末から入力した'Hello'という文字列が表示されています。tail -fによるファイルの読み込みは「Ctrl」Cキーを押して処理を中断するまで継続されます。

tail -fはこの実習から分かるように、ファイルを動的に読み込むことができます。Web サーバーのアクセスログやエラーログを見る場合に便利なコマンドです。

5.3 テキストファイルのソート (sort)

テキストファイルの中身をソートするには sort コマンドが使えます。オプションでどのような順序でソートするか指定することができます。

書式

```
sort [オプション] ファイル名
```

オプション

-r
逆順でソートする

-k n
n 列目のデータをソートする

-n
数値としてソートする

5.3.1 ファイルの準備

sort の機能を確認するために、サンプルとなるファイルを作成します。このファイルを元に、sort を動作させます。内容は、3人の名前と数値です。左から順に '名前, 名字, 点数' となっていると考えてください。

```
$ cat > score
yoshinori kawazu 85
keiichi oka 70
toru minemura 100
(Control-d を押す)
$
```

5.3.2 sort の実行

テストファイルの作成が終わったら、sort コマンドとオプションを付けてさまざまな条件でソートしてみましょう。

○実習: sort コマンドの実行

まずはオプションを付けずにソートしてみます。

5.3 テキストファイルのソート (sort)

```
$ sort score
keiichi oka 70
toru minemura 100
yoshinori kawazu 85
```

オプションを特に付けない場合は各行の 1 文字目をアルファベット順 ($k \rightarrow t \rightarrow y$) でソートされます。

○実習: -r オプションを付けた逆順によるソート

sort コマンドにオプション-r を付けた場合はどうなるでしょうか。

```
$ sort -r score
yoshinori kawazu 85
toru minemura 100
keiichi oka 70
```

アルファベットの逆順 ($y \rightarrow t \rightarrow k$) にソートされます。

5.3.3 n 列目のデータソート (-k)

「sort を実行する」では行の 1 文字目を鍵としてソートを行いました。これは名前でソートしているのと同じです。では、名字を鍵としてソートするにはどうしたら良いでしょうか？ sort では-k オプションを付けると、ソートの鍵として利用する列の番号を指定します。

○実習: -k オプションを付けたソート

名字順でソートしてみましょう。2 列目の名字を鍵としてソートするので、オプションに 2 を指定します。

```
$ sort -k 2 score
yoshinori kawazu 85
toru minemura 100
keiichi oka 70
```

2 列目の名字の部分を基準にソートされます ($k \rightarrow m \rightarrow o$)。

○実習: さまざまなオプションを付けたソート

オプションとしてさらに-r を指定すると、逆順でソートできます。

```
$ sort -k 2 -r score
keiichi oka 70
toru minemura 100
yoshinori kawazu 85
```

-r を指定することで先ほどの逆順でソートされました (o → m → k)。

5.3.4 数値式でのソート

今度は得点の列 (3列目) を鍵としてソートしてみましょう。コマンドは次のようにになります。

実行例

```
$ sort -k 3 score
toru minemura 100
keiichi oka 70
yoshinori kawazu 85
```

並び方が少し変ですね。大きい順で並ぶのであれば 100 → 85 → 70 になるべきですし、小さい順で並ぶのであれば 70 → 85 → 100 になるべきです。しかし、実行例のような結果になるのは、これら 3つの数字が「文字として」認識されているからです。sort は前述の通り、1 文字目しかみていません。従ってアルファベットをソートしたのと同様に、数値も最初の 1 文字を見てソートしています (1 → 7 → 8)。

これを辞書式ソートといって、文字を辞書に出てくる順にソートできます。アルファベットであればそれでかまいませんが、3列目は得点=数値であり、得点順にソートする場合はこの辞書式ソートは不適切です。これに対して、数値としてソートするのを数値式ソートといい、sort に-n オプションを付けることで実行できます。

5.3 テキストファイルのソート (sort)

○実習: 数値式ソートの実行

それでは早速数値式ソートを実践してみます。文字列の 3 列目を基準として数値式ソートしましょう。

```
$ sort -n -k 3 score
keiichi oka 70
yoshinori kawazu 85
toru minemura 100
```

リストが得点の低い順にソートされました。

○実習: 数値式ソートの実行 (逆順)

オプションにさらに-r を付加すると、逆順でソートすることができます。

```
$ sort -n -r -k 3 score
toru minemura 100
yoshinori kawazu 85
keiichi oka 70
```

-r を付加したことにより、得点の高い順にソートすることができました。

5.4 行の重複の消去 (uniq)

uniq コマンドを使うことで直前の行と同じ内容があった場合、対象行を出力しません。連続している同じ内容の行を、1行にまとめることができます。

書式

```
uniq ファイル名
```

○実習: uniq 確認用のファイル作成

uniq コマンドの動きを確認するため、以下のような内容のファイルを uniq-sample として保存しておいてください。

```
$ cat > uniq-sample
AAA
BBB
AAA
CCC
CCC
DDD
```

○実習: uniq コマンドの実行

作成した uniq-sample に対して、uniq コマンドを早速実行してみましょう。

```
$ uniq uniq-sample
AAA
BBB
AAA
CCC
DDD
```

uniq の実行により、重複して記述されていた文字列 CCC が1行にまとめられました。このように uniq コマンドを使うことで全データから重複しない要素だけをピックアップすることができます。AAA は2行ありますが、結果はそのまま出力されています。これは、AAA の行は連続していないため、uniq の処理対象にはならないためです。

5.5 文字列の置き換え (tr)

tr コマンドを使って、標準入力からのデータを文字毎に置き換える (TRanslate) ことができます。

書式

```
tr 文字列 1 文字列 2
```

tr コマンドは文字を文字毎に別の文字に置き換えることができます。置き換える元のデータは標準入力からのデータを対象にします。実行例で動きを説明しましょう。

実行例

```
$ cat FILE | tr abc ABC
```

この例では以下のようなことを行ないます。

- cat コマンドで FILE を開く。
- tr コマンドで a,b,c をそれぞれ A,B,C に置き換える。

○実習 tr コマンドによる置き換え

では、次の例を実行したらどういった結果になるでしょうか。実際にファイルを作成して実行してみてください。

ファイルを作成

```
$ cat > translate
Android
iPhone
Windows Phone
(「Ctrl」Dを押す。)
```

実行

```
$ cat translate | tr on ON
ANdRoid
iPhOnE
WiNdOws PhOnE
```

tr コマンドについて理解できたでしょうか。tr コマンドに 2 つのパラメーターを渡すと、文字毎に指定した文字を別の文字に置き換えます。従って例題のファイルに含まれる o と n の文字が、大

文字の O と N に置き換えられたというわけです。

tr コマンドの結果は標準出力に出力されますが、以下のようにリダイレクトすればファイルに出力することもできます。

tr の結果をリダイレクトを使ったファイル出力

```
$ cat translate | tr on ON > translate2
$ cat translate2
ANdroid
iPhONe
WiNdOws PhONe
```

ファイル translate には変更が記録されず、tr の実行結果はファイル translate2 にファイル出力されます。

5.6 ファイルの比較 (diff)

Linux にはファイルを比較するコマンドがあります。主に使われる用途として、変更の有無を調べる場合に用いられます。diff の結果は標準出力されますが、リダイレクトすればファイルに出力することもできます。

書式

```
diff [オプション] ファイル1 ファイル2
```

オプション

```
-c
context diff 形式で差分を出力します。

-u
unified diff 形式で差分を出力します。
```

実行例

```
$ diff file1 file2
$ diff -c file1 file2
$ diff -u file1 file2
```

5.6 ファイルの比較 (diff)

○実習: diff コマンドの実行

diff コマンドの動きを確認するため、以下のように実際に異なるファイルの file1 と file2 を作成しておいてください。

```
$ echo "test text" > file1  
$ echo "test text" > file2  
$ echo "new line" >> file2
```

ファイルが作成できたら、diff コマンドでファイルを比較してみましょう。

diff コマンドによる比較

```
$ diff file1 file2  
1a2  
> new line
```

diff -u コマンドによる比較

```
$ diff -u file1 file2  
--- file1      2012-07-06 11:00:00.098086703 +0900  
+++ file2      2012-07-06 12:00:00.394135769 +0900  
@@ -1 +1,2 @@  
 test text  
+new line
```

diff -c コマンドによる比較

```
$ diff -c file1 file2  
*** file1      2012-07-06 11:00:00.098086703 +0900  
--- file2      2012-07-06 12:00:00.394135769 +0900  
*****  
*** 1 ****  
--- 1,2 ----  
 test text  
+ new line
```

オプションを付けずに diff コマンドを実行した場合は、異なる部分を標準出力します。今回紹介したオプションを付けて diff コマンドを実行した場合、ファイルの更新時刻を含めたより多くの差分情報が表示されます。

たとえばここで、file2 の内容を別の文字列で上書きして diff コマンドを実行してみてください。

```
$ echo "overwrite text" > file2  
(file2 を別の文字列で上書き)  
$ echo "new line" >> file2
```

file1 と file2 の共通点はなくなりましたので、diff コマンドを使って比較すると以下のように標準出力されます。

diff コマンドによる比較

```
$ diff file1 file2  
1c1,2  
< test text  
---  
> overwrite text  
> new line
```

3つの差分があると表示されていますが、少しあわかりにくいですね。今度は diff -u コマンドで比較してみましょう。

diff -u コマンドによる比較

```
$ diff -u file1 file2  
--- file1      2012-07-06 13:21:21.219249366 +0900  
+++ file2      2012-07-06 13:26:51.664814277 +0900  
@@ -1 +1,2 @@  
-test text  
+overwrite text  
+new line
```

diff -u コマンドによる比較は、削除された行の頭にマイナス記号、追加された行にプラス記号が付加されて標準出力されます。標準出力の結果から、file1 と file2 の共通していた文字列である「test text」が削除され、新たに「overwrite text」と「new line」という行が差分追加されているということがわかります。オプション-u は文字列の増減が激しい、文書やプログラムのソースコードを比較する場合に有用であるといえます。

5.7 5章 章末テスト

(1)sort コマンドの実行結果となるように、適切なオプションを付けなさい。

```
$ sort (      ) price
Digital Camera 9800
32-inch TV 49800
Blue-ray Recorder 59800
```

(2)uniq-sample というファイルに対して uniq コマンドを実行したときの実行結果を答えなさい。

```
$ cat uniq-sample
red
red
blue
red
blue
```

(3)file1 と file2 の内容の違いを比較するコマンドを記述しなさい。

(4) 以下のコマンドを実行したところ、何も結果が表示されずコマンド待ち状態になりました。何が考えられるか答えなさい。

```
$ diff test1 test2
$
```

(5) パス /etc の ls コマンドの実行結果が保存されたファイル ls/etc の終わり 8 行分を表示する場合のコマンドを記述しなさい。

```
$ (      ) ls/etc
wpa_supplicant
xdg
xinetd.d
xml
yp.conf
yum
yum.conf
yum.repos.d
```

第6章

vi エディタ

Linux の設定をするためにしばしば利用される vi(vim) を操作しながら、エディタの基本的な機能を学んでみましょう。本章では vi の機能のうち、ファイルの開閉やテキストの検索や置換、ページ移動や編集方法など基礎的な操作を解説します。

この章の内容

- 6.1 vi の基本操作
 - 6.1.1 ファイルを開く
 - 6.1.2 ファイルを閉じる
 - 6.1.3 ファイルを保存する
 - 6.1.4 ファイルを保存して終了する
 - 6.1.5 ファイルを保存せずに強制的に閉じる
- 6.2 インサートモードとコマンドモード
 - 6.2.1 テキストの入力
 - 6.2.2 カーソルの移動
- 6.3 編集中の大きな移動
 - 6.3.1 ページ単位の移動
 - 6.3.2 行を指定した移動
- 6.4 様々な編集操作
 - 6.4.1 文字のカット・アンド・ペースト
 - 6.4.2 行のカット・アンド・ペースト
 - 6.4.3 行のコピー・アンド・ペースト
 - 6.4.4 編集の取り消し(アンドウ)
- 6.5 置換と検索
 - 6.5.1 文字列の検索
 - 6.5.2 文字列の置換
- 6.6 章末テスト

6.1 vi の基本操作

vi はページャであり、エディタでもあります。vi でファイルを開いた場合はコマンドモード（view モード）といって、ページングによる行やページ単位の移動や、行の削除、コピー、カット、ペーストなどの編集操作を行なうことができるモードと文字入力を受け付けるインサートモードがあります。

エディタを開始する方法は vi コマンドに続けて編集したいファイルのファイル名を指定することで作業を開始できます。エディタを終了する方法は「コマンドモード」で終了コマンドを実行することで vi エディタによる編集作業を終えることができます。

それでは早速、基本操作であるファイルの開閉とファイルの保存と終了の方法について実際にコマンドを実行してみましょう。

6.1.1 ファイルを開く

vi コマンドでファイルを開くには vi コマンドの後に編集したいファイルを指定します。

書式

```
vi [ファイル名]
```

6.1.2 ファイルを閉じる

vi コマンドを使って開いたファイルを閉じるには、「ESC」キーを押したあと、:q と入力します。q は quit の略です。ファイルを閉じることができます。

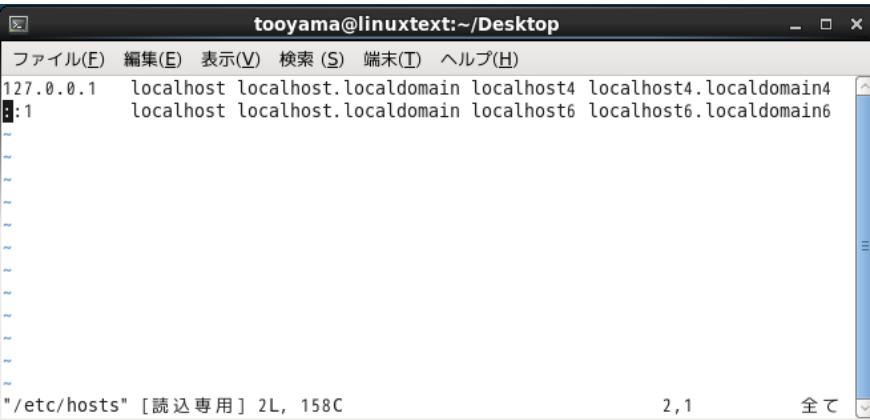
○実習: vi によるファイルの開閉

vi を使って、既存の設定ファイルである /etc/hosts ファイルを開き、内容を確認したら閉じてみましょう。

1. まずは既存のファイルを開いてみます。

```
$ vi /etc/hosts
```

2. ファイルが開きました。

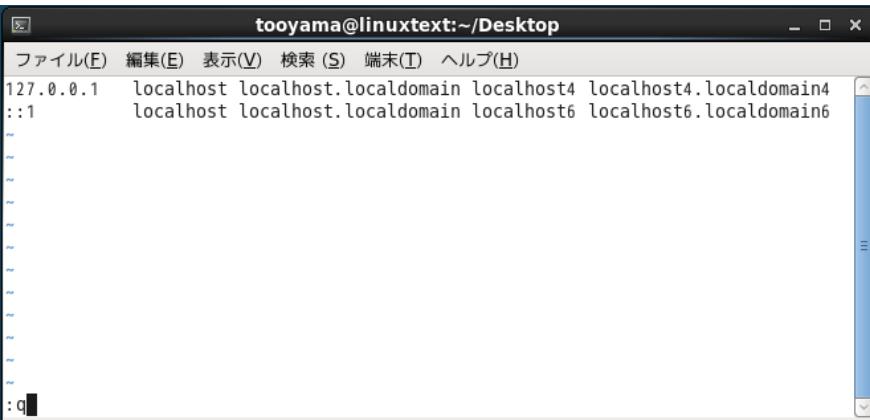


A screenshot of a terminal window titled "tooyama@linuxtext:~/Desktop". The window shows the contents of the "/etc/hosts" file. The file contains the following entries:

```
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1      localhost localhost.localdomain localhost6 localhost6.localdomain6
```

The status bar at the bottom indicates ""/etc/hosts" [読み専用] 2L, 158C". The bottom right corner of the status bar has a dropdown menu icon.

3. :q コマンドを実行してファイルを閉じます。



A screenshot of a terminal window titled "tooyama@linuxtext:~/Desktop". The window shows the contents of the "/etc/hosts" file. The status bar at the bottom indicates ""/etc/hosts" [読み専用] 2L, 158C". In the bottom left corner of the editor area, the command ":q" is visible.

4. vi コマンドが終了して、コマンド入力待機状態に戻ります。

6.1.3 ファイルを保存する

vi コマンドを使って開いたファイルを保存するには、「ESC」キーを押したあと、:w と入力します。w は write の略です。ファイルに変更内容が書き込まれて、編集作業を継続できます。

test.txt というファイルを開き、ファイルを保存してみましょう。

```
$ vi test.txt
```

↓

6.1 vi の基本操作

```
:w  
(コマンドを入力)
```

↓

```
"test.txt" 1L, 5C 書込み
```

6.1.4 ファイルを保存して終了する

vi コマンドを使って開いたファイルを保存したあとファイルを閉じるには、「ESC」キーを押したあと、:wq と入力します。w により変更内容が書き込まれ、q によりファイルを閉じます。

test.txt というファイルを開き保存したあと、ファイルを閉じて終了してみましょう。

```
$ vi test.txt
```

↓

```
:wq
```

↓

```
$ vi test.txt  
$
```

6.1.5 ファイルを保存せずに強制的に閉じる

vi コマンドを使って開いたファイルを保存せずに閉じるには、「ESC」キーを押したあと、:q! と入力します。通常、編集済みのファイルはそのままでは閉じることができません。:q コマンドのあとに!をつけることで、ファイルを保存せずに強制的に終了できます。

test.txt というファイルを開き、ファイルを強制的に閉じてみましょう。

```
$ vi test.txt
```

↓

```
:q!
```

↓

```
$ vi test.txt  
$
```

○実習: vi によるファイルの操作

vi のさまざまな操作を行なってみましょう。vi の詳細な操作手順はこのあと説明しますので、ここでまずは手順に従って操作してみてください。

1. 以下のコマンドを実行します。

```
$ vi editfile.txt
```

2. 内容が空のファイルが開きます。



6.1 vi の基本操作

3. 「i」キーを押して「インサート」モードに切り替えます。



4. キーボードで「Apple」と入力しましょう。



5. 「ESC」キーを押して「インサート」モードから抜けます。

6. :w コマンドを実行してファイルを保存します。



7. ファイルが保存され、「書込み」と表示されます。



8. 「コマンド」モードで「A」キーを押して行末まで移動し、「Enter」キーを押して改行します。

6.1 vi の基本操作

9. 次の行に「Orange」と入力しましょう。



10. 「ESC」キーを押して「インサート」モードから抜けます。

11. :q コマンドを実行して、ファイルを閉じてみます。

12. 変更した内容が保存されていないため、エラーが表示されます。



13. :wq コマンドを実行すると、ファイルを閉じることができることを確認します。

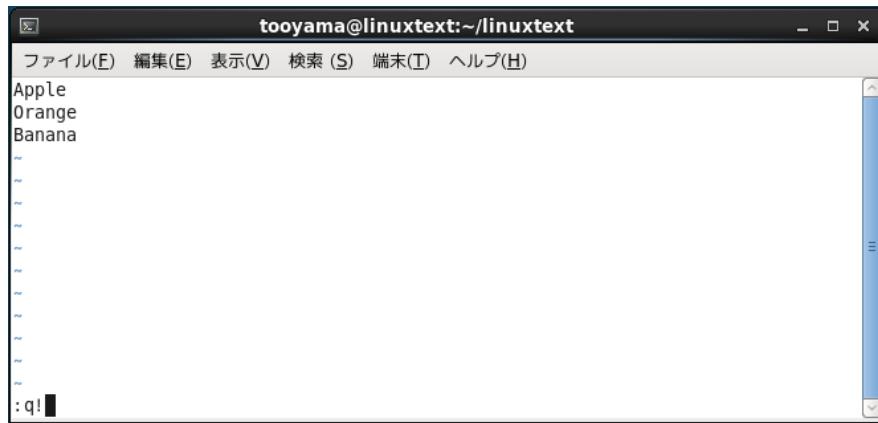
14. もう一度ファイル editfile.txt を開きます。

15. 「コマンド」モードで「A」キーを押して行末まで移動し、「Enter」キーを押して改行します。

16. 次の行に「Banana」と入力しましょう。

17. 「ESC」キーを押して「インサート」モードから抜けます。

18. :q!コマンドを実行すると、ファイルを閉じることができるので確認します。



19. ファイルを開き、「Banana」という文字列がファイルに書かれていないことを確認します。



vi はファイルの操作や閲覧を行なう「コマンドモード」と、ファイルの編集を行なう「インサートモード」の切り替えという概念はあるものの、使い慣れているエディタとほぼ同じように操作できることがわかったと思います。

次の手順以降でさらにいろいろな vi の機能を学習しましょう。

6.2 インサートモードとコマンドモード

vi エディタを起動すると、コマンドを入力するコマンドモードでファイルが開かれます。文字列を打ち込むには i コマンドもしくは a コマンドなどでインサートモードへ切り替えます。vi コマンドはスクリーンエディタと呼ばれ、基本的にカーソルがある位置に対してコマンドを実行します。

6.2.1 テキストの入力

文字列を入力するインサートモードは、コマンドでインサートモードへ切り替える必要があります。インサートモードへの切り替えにはさまざまなコマンドがありますが、ここでは **i** コマンドと、**a** コマンドについて説明します。

○実習: テキストの入力

早速ファイルを開いて文字列を入力してみましょう。書き換え可能なファイルである `viinsert.txt` を新規作成してみます。

```
$ vi viinsert.txt  
"viinsert.txt" [新ファイル] 0,0-1 全て
```

文字列を入力するにはインサートモードへ移行します。カーソルがある位置(文字の前)にテキストを入力するには **i** コマンド(insert コマンド)を使います。

早速文字を入力してみましょう。

1. まず、インサートモードに切り替えるため、**i** コマンドを入力します。
2. 画面左下に挿入と表示されます。

```
-- 挿入 --
```

3. 文字を入力します。

```
test
```

インサートモードではカーソルがある位置にテキストを挿入しましたが、カーソルのある文字の後にテキストを入力するモードもあります。早速文字を入力してみましょう。入力モードから抜け
るには「Esc」キーを入力してください。

1. まず、インサートモードに切り替えるため、**a** コマンド append コマンド)を入力します。
2. 画面左下に挿入と表示されます。

```
-- 挿入 --
```

3. 文字を入力します。

```
testTEST
```

前の手順で入力した文字列の末尾に今回入力した TEST が追加されました。

カーソルの位置ではなく、行の先頭にテキストを入力するには I コマンドを使います。早速文字を入力してみましょう。入力モードから抜けるには「Esc」キーを入力してください。

1. まず、インサートモードに切り替えるため、I コマンドを入力します。
2. 画面左下に挿入と表示されます。

```
-- 挿入 --
```

3. 文字を入力します。

```
1234testTEST
```

前の手順で入力した文字列の先頭に今回入力した 1234 が追加されました。

行の最後にテキストを入力するには A コマンドを使います。早速文字を入力してみましょう。入力モードから抜けるには「Esc」キーを入力してください。

1. まず、インサートモードに切り替えるため、A コマンドを入力します。
2. 画面左下に挿入と表示されます。

```
-- 挿入 --
```

3. 文字を入力します。

```
1234testTEST6789
```

前の手順で入力した文字列の末尾に今回入力した 6789 が追加されました。

6.2 インサートモードとコマンドモード

文字を間違えたときは「Delete」や「BS」でカーソルの前の1文字が削除できます。「Delete」キーや「BS」キーの動作は使う環境により変わってきます。ファイルを変更したので、:wq コマンドで入力した文字列をファイルへ保存してから終了してください。

```
:wq  
"viinsert.txt" [新] 1L, 17C 書込み
```

6.2.2 カーソルの移動

テキストファイルを編集していると、入力間違いに気がついたり、後からの変更が出てくるので、既に入力したテキストを修正したいときがあります。追加する文字がある場合は、カーソルを修正したいところへ移動して文字を入力します。

○実習：カーソルの移動

ファイルを開いて文字列を入力した後でカーソルを移動する練習をするため、書き換え可能なファイルである vimove.txt を新たに開きます。

```
$ vi vimove.txt
```

以下の例文を入力してください。

```
Apple  
Orange  
Banana  
Melon
```

カーソルの左右移動

vi でカーソルを左へ移動するには「h」を、カーソルを右へ移動するには「l」と入力します。 CentOS 6 で利用できる vi コマンドでは、カーソルキーの左右を使ってカーソルの左右移動することもできます。

カーソルの上下移動

カーソルの上下移動は「j」で下へ、「k」で上へ移動することが可能です。CentOS 6 で利用できる vi コマンドでは、カーソルキーの上下を使ってカーソルの上下移動することもできます。

コマンドを使った行頭への移動

0（ゼロ）コマンドを実行すると、カレント行の行頭へ移動することができます。

コマンドを使った行末への移動

\$コマンドを実行すると、カレント行の行末へ移動することができます。

6.3 編集中の大きな移動

vi コマンドは小さなファイルから大きなファイルまでを編集できます。1画面に収まらない大きなファイルを編集する場合は1文字ずつカーソルを移動するコマンドだけでは大変でしょう。ページ単位で移動するコマンドや、指定の行へ移動するコマンドや、目的の文字列へ移動するコマンドを使います。

6.3.1 ページ単位の移動

素早く編集するために画面をページ(デフォルト指定の行数)単位で前後へ移動するコマンドを使います。

○実習: ページ単位の移動

ページ単位の移動の実習を行なうため、vi コマンドで既にある /var/log/dmesg ログファイルを開いてください。実習が終わったら編集内容を保存せずに終了します。

```
$ vi /var/log/dmesg
"/var/log/dmesg" [読み専用] 299L, 14056C
```

次のページへの移動は「Ctrl」f コマンドです。前のページへの移動は「Ctrl」b コマンドです。

```
:q!
$
```

実習が終わったら、今回は利用したログファイルを保存しないで終了するために:q!コマンドで vi を終了してください。

6.3.2 行を指定した移動

素早く編集するために指定の行へカーソルを移動するコマンドを使います。

○実習: 行の移動

開いたファイルから指定行への移動を行なう実習を行なうため、vi コマンドで /var/log/dmesg ファイルを開いてください。

6.4 様々な編集操作

```
$ vi /var/log/dmesg  
"/var/log/dmesg" [読み専用] 299L, 14056C
```

今回開いた dmesg ファイルのように、大きなファイルではページ単位の移動だけでは大変なこともあります。ここではファイルを効率的に閲覧するために行数を指定して移動したり、行を移動するコマンドを使ってみましょう。

行番号を指定した移動

行番号を指定して移動するには、これまでにコマンドを入力する時に出てきた:の後に行番号を指定します。たとえば 10 行目へ移動したい場合は、「:10」と入力してください。10 行目へカーソルが移動します。

```
:10
```

「:1」と入力すると 1 行目へ戻ります。

```
:1
```

コマンドを使った文書頭への移動

gg コマンドを実行すると文書頭、つまり 1 行目へ戻ることができます。

コマンドを使った文書末への移動

G コマンドを実行すると文書末、つまり最終行へ移動することができます。

実習が終わったら、今回は利用したログファイルを保存しないで終了するために:q!コマンドで vi を終了してください。

```
:q!  
$
```

6.4 様々な編集操作

vi の編集の基礎である「カット・アンド・ペースト」や「コピー・アンド・ペースト」、アンドウについて説明します。

6.4.1 文字のカット・アンド・ペースト

文字のカット・アンド・ペーストの機能はコマンドモードで実行します。文字をカットするコマンドと、カットした文字をペーストするコマンドであるコマンドを利用します。

表 6.1 カット・アンド・ペーストで使うコマンド

コマンド	内容
x	1 文字削除
dd	1 行削除
yy	1 行コピー
n yy	n 行コピー
p	カーソルの文字の次または次の行にペースト
P	カーソルの文字の前または前の行にペースト
u	カット、ペーストを一回取り消し（アンドウ）

○実習：文字のカット・アンド・ペースト

vi コマンドで vicutpaste.txt ファイルを開き、カット・アンド・ペーストするためのサンプル文字列を入力してください。ここでは以下の画面と同じになるように i コマンドでインサートモードにしたあと、以下のように入力します（わざと間違えています）。

```
$ vi vicutpaste.txt
This is aa pen.
That is na apple.
(略)
"vicutpaste.txt" [新ファイル] 0,0-1 全て
```

上記の例文は 2 つの誤りがあります。一つ目は aa と連続している箇所、そしてもう一つは na とミスマッチしている箇所です。

これを正しい文に修正しましょう。

1. まずコマンドモードにしてカーソルを aa の位置に移動します。
2. x コマンドで aa の a を 1 文字削除してください。

```
This is a pen.
That is na apple.
```

これで一行目の文の修正ができました。

先ほど文字の削除に利用した x コマンドは、x コマンドで削除した最後の文字はバッファと呼ばれる一時的な箱に入ります。そのため、p コマンドを実行するとバッファに保存された文字を取り

6.4 様々な編集操作

出すことができます。

次に二行目の文を修正しましょう。

1. まずコマンドモードにしてカーソルを na の n の位置に移動します。
2. x コマンドで na の n を 1 文字削除してください。
3. x コマンドで n が削除されました。

```
This is a pen.  
That is a apple.
```

4. a にカーソルがある状態で p コマンドを入力します。

```
This is a pen.  
That is an apple.
```

a のあとに n が挿入され、これで二行目の文の修正ができました。

p コマンドはカーソルがある文字の後に文字をペーストします。なお、カーソルの前に文字をペーストしたいときは P コマンド (Paste コマンド) を使います。

6.4.2 行のカット・アンド・ペースト

行単位のカット・アンド・ペーストとコピー・アンド・ペーストの機能はコマンドモードで、行のカットをするコマンドと、カットした文字列をペーストするコマンドを使います。

○実習: 1 行の文字列をカット・アンド・ペースト

vi コマンドで vicutlinepaste.txt ファイルを開き、サンプル文字列を入力してください。ここでは以下の画面と同じになるように i コマンドでインサートモードにしたあと、「This is a pen.」「Enter」
「That is an apple.」「Esc」と入力します。

```
$ vi vicutlinepaste.txt  
This is a pen.  
That is an apple.  
(略)  
"vicutlinepaste.txt" [新ファイル] 0,0-1 全て
```

この例文を使って一行カットしてペーストしてみましょう。

行単位でカットしたい場合は dd コマンドを使います。カットした行のテキストはバッファへ入るため、p コマンドまたは P コマンドを実行すると文字列をペーストできます。

「This is a pen」を dd コマンドで一行カットしたあと、p コマンドと P コマンドを実行して違い

を確認してみましょう。

p コマンド

```
That is an apple.  
This is a pen.
```

文字列がカーソルの下にペーストされます。

P コマンド

```
This is a pen.  
That is an apple.
```

文字列が カーソルの上にペーストされます。

実習が終わったら、:wq コマンドでファイルを保存して閉じてください。

6.4.3 行のコピー・アンド・ペースト

行単位で文字列をコピー・アンド・ペーストすることもできます。まず yy コマンドで行全体をコピーします。yy コマンドを実行すると、文字列はバッファに入ります。バッファ内のテキストは p コマンドもしくは P コマンドでペーストできます。

○実習: 1 行の文字列のコピー・アンド・ペースト

vi コマンドで vicopypaste.txt ファイルを開き、サンプル文字列を入力してください。ここでは以下の画面と同じになるように i コマンドでインサートモードにしたあと、「This is a pen. 「Enter」 That is an apple. 「Esc」」と入力します。

```
$ vi vicopypaste.txt
This is a pen.
That is an apple.
(略)
"vicopypaste.txt" [新ファイル]          0,0-1      全て
```

今回は 2 行目をコピーしたいと思います。2 行目にカーソル移動して yy コマンドでコピーしてください。コピーした内容は p コマンドでペーストしてください。実行結果は以下のようになります。

```
This is a pen.  
That is an apple.  
That is an apple.
```

yy コマンドでコピーした内容を p コマンドでペーストした場合、現在カーソルがある行の下に挿入されます。

コピーしたい行数を指定することもできます。指定する方法は yy コマンドの前に行数を指定するだけです。3 行コピーしてペーストしてみましょう。複数行コピーするには「コピーしたい行数 yy」のように指定します。

次の例では 3 行をコピーしたいと思います。gg コマンドを入力して文書頭まで戻ったあと 3yy とコマンドを入力してください。yy コマンドでコピーした文字列はバッファに入ります。p コマンドを実行してペーストしてください。

```
This is a pen.  
This is a pen.  
That is an apple.  
That is an apple.  
That is an apple.  
That is an apple.
```

※太字の部分がペーストした部分です。

実習が終わったら、:wq コマンドでファイルを保存して閉じてください。

6.4.4 編集の取り消し (アンドウ)

テキストの編集で何か失敗をしたとき、それを元に戻したいことがあります。そのためのコマンドがアンドウ (undo) で、u コマンドを入力します。vi エディタでは、すべての編集操作について、u コマンドでそれを取り消すことができます。

○実習：文字編集の取り消し

vicutpasteundo.txt という以下の内容のファイルを作成して u コマンドを実行し、早速カット・アンド・ペーストの取り消しを試してみましょう。

6.5 置換と検索

1. ファイルを作成します。

```
$ vi vicutpasteundo.txt
This is a pen.
(略)
```

2. x コマンドで、文字列を全部削除してください。
3. u コマンドで、削除した文字列が一文字ずつ挿入されるのを確認します。
4. :q!コマンドでファイルを閉じます。
5. もう一度、ファイル vicutpasteundo.txt を開いてください。
6. yy コマンドを押して、行をコピーしてください。
7. p コマンドで、「This is a pen.」をペーストできることを確認します。
8. u コマンドで、ペーストした文字列が削除されるのを確認します。

実習が終わったら、:wq コマンドでファイルを保存して閉じてください。

6.5 置換と検索

巨大であったり、全体像を把握していないファイルはカーソル移動やページ移動のコマンドだけを駆使して目的の位置へたどりつくのは大変です。テキストファイルの中にある文字列を検索するコマンドを使うと、素早く目的の位置に移動できます。また、置換コマンドを使えば文章中の文字列を探して別の文字列に置き換えることも素早く行なうことができます。

表 6.2 検索と置換に関するコマンド

コマンド	内容
/検索文字列	文字列の検索
n	下方向へ再検索
N	上方向へ再検索
:対象の行 s/検索文字列/置換文字列/オプション	文字列を置換する

6.5.1 文字列の検索

テキストの中から目的の文字列へカーソルを移動するためには / コマンドを使います。/コマンドのあと、検索したい文字列を続けて入力します。

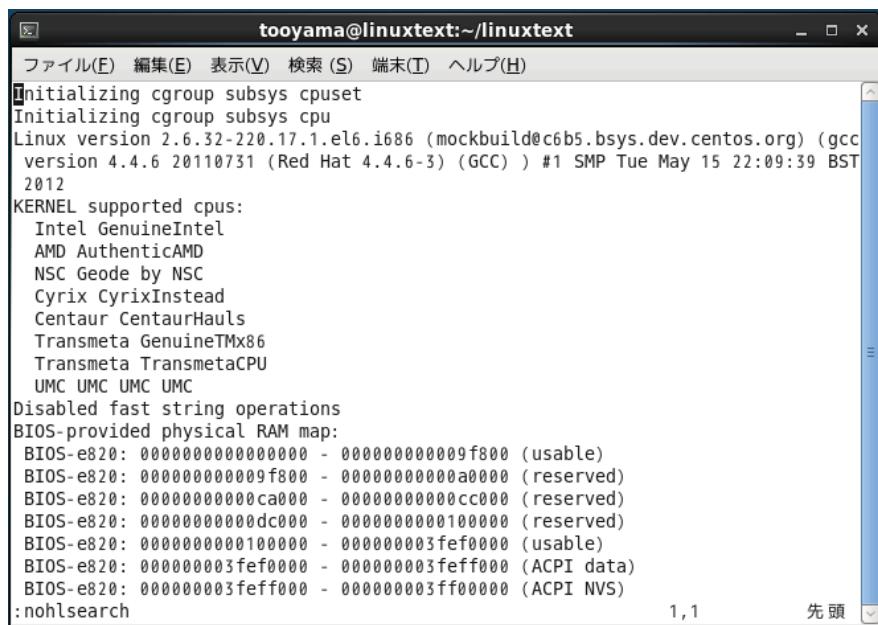
書式

```
/文字列
```

○実習: 文字列の検索

vi コマンドで /var/log/dmesg ファイルを開き、このファイルから特定の文字列を検索してみましょう。

```
$ vi /var/log/dmesg
"/var/log/dmesg" [読み専用] 299L, 14056C
```



```
tooyama@linuxtext:~/linuxtext
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(I) ヘルプ(H)
Initialization cgroup subsys cpuset
Initialization cgroup subsys cpu
Linux version 2.6.32-220.17.1.el6.i686 (mockbuild@c6b5.bsys.dev.centos.org) (gcc
version 4.4.6 20110731 (Red Hat 4.4.6-3) (GCC) ) #1 SMP Tue May 15 22:09:39 BST
2012
KERNEL supported cpus:
Intel GenuineIntel
AMD AuthenticAMD
NSC Geode by NSC
Cyrix CyrixInstead
Centaur CentaurHauls
Transmeta GenuinetMx86
Transmeta TransmetaCPU
UMC UMC UMC UMC
Disabled fast string operations
BIOS-provided physical RAM map:
BIOS-e820: 0000000000000000 - 00000000000f800 (usable)
BIOS-e820: 00000000009f800 - 00000000000a0000 (reserved)
BIOS-e820: 0000000000ca000 - 00000000000cc000 (reserved)
BIOS-e820: 0000000000dc000 - 0000000000100000 (reserved)
BIOS-e820: 0000000000100000 - 000000003fef0000 (usable)
BIOS-e820: 000000003fef0000 - 000000003feff000 (ACPI data)
BIOS-e820: 000000003feff000 - 000000003ff00000 (ACPI NVS)
:nohlsearch
```

文字列を検索するコマンドは / です。/コマンドの後に探したい文字列を入力し、「Enter」キーを入力するとカーソルが文字列へ移動します。ここでは CPU を検索するため、「/CPU」を入力しました。

ファイルの中には同じ文字列が複数現れることもあります。/コマンドで 1 回検索した文字列は、n コマンドまたは N コマンドで繰り返し探せます。n キーを押すと 1 つ次の文字列を再検索し、N

6.5 置換と検索

キーは 1 つ前の文字列を再検索し、n キーまたは N キーを押した回数だけ再検索できます。

文書末まで再検索を行なうと、「下まで検索したので上に戻ります」と表示され、もう一度再検索を文書頭から行なうことができます。

文字列を検索すると検索した文字列がハイライト表示されます。このハイライト表示は次の文字列を検索するまでハイライト表示が残ります。ハイライトを消すには:nohlsearch コマンドを実行してください。

実習が終わったら、今回は利用したログファイルを保存しないで終了するために:q!コマンドで vi を終了してください。

```
:q!  
$
```

6.5.2 文字列の置換

vi コマンドでは文字の置換を行なうことも簡単です。置換する対象は行範囲を指定することも、文書全体を対象にすることもできます。

書式

```
:対象の行 s/検索文字列/置換文字列/オプション
```

表 6.3 置換方法の指定

コマンド	内容
:ns/old/new	n 行目の最初の old を new に置換して終了
:ns/old/new/g	n 行目の全ての old を new に置換して終了
:%s/old/new/g	ファイル全体の検索語句を置換する
:%s/old/new/gc	置換の度に確認を求める

文字列を置換するコマンドは「:対象の行 s/検索文字列/置換文字列/オプション」と記述できます。対象の行を省略すると現在カーソルがある行だけが置換対象になります。オプションに g を指定すると 1 行にある複数の検索文字列を置換します (g を指定しないと行ごとに 1 回だけ置換します)。

○実習: 文字列の置換

vi コマンドで vireplace.txt ファイルを開き、サンプル文字列を入力したものを用意してください。このファイルを使って文字列の置換を試してみましょう。

```
$ cat vireplace.txt
lemon orange banana melon lemon
orange banana melon lemon orange
banana melon lemon orange banana
melon lemon orange banana melon
lemon orange banana melon lemon
orange banana melon lemon orange
banana melon lemon orange banana
melon lemon orange banana melon
```

指定行の文字列置換

指定行を置換対象とする場合は:指定行 s/old/new/.. のように記述します。

1. まず、1行目の banana を BANANA に置換してみましょう。以下のように操作してみてください。

```
:1s/banana/BANANA/g
```

2. 結果は以下のようになつたはずです。

```
lemon orange BANANA melon lemon
(略)
```

3. 次に、orange を ORANGE に置換してみましょう。以下のように操作してみてください。

```
:1s/orange/ORANGE/
```

4. 結果は以下のようになつたはずです。

```
lemon ORANGE BANANA melon lemon
(略)
```

今回は1行目には banana や orange が重複していなかつたため、いずれも1つの文字列のみ置換されました。次に g オプションありと g オプションなしの時の動作の違いを確認してみましょう。

1. まず、2行目の orange を ORANGE に置換してみましょう。以下のように操作してみてください。

6.5 置換と検索

```
:2s/orange/ORANGE/g
```

- 結果は以下のようになったはずです。

```
ORANGE banana melon lemon ORANGE  
(略)
```

g オプションを付けた場合、指定した行にあるすべての文字列を別の文字列に置換します。

- 次に、3 行目の banana を BANANA に置換してみましょう。以下のように今回は g オプション無しで操作してみてください。

```
:3s/banana/BANANA/
```

- 結果は以下のようになったはずです。

```
BANANA melon lemon orange banana  
(略)
```

g オプションを付けなかった場合、指定した行にある最初の該当文字列のみ、別の文字列に置換して終了します。

ファイル全体の文字列置換

ファイル全体を置換対象とする場合は:%s/old/new/g のように記述します。

- ファイル全体の melon を*MELON* に置換してみましょう。以下のように操作してみてください。

```
:%s/melon/*MELON*/g
```

- 結果は以下のようになったはずです。

```
lemon ORANGE BANANA *MELON* lemon  
ORANGE banana *MELON* lemon ORANGE
```

```
BANANA *MELON* lemon orange banana
*MELON* lemon orange banana *MELON*
lemon orange banana *MELON* lemon
orange banana *MELON* lemon orange
banana *MELON* lemon orange banana
*MELON* lemon orange BANANA *MELON*
```

行数ではなく % を指定した場合、置換範囲はファイル全体を検索範囲とします。

ファイル全体の中にある melon という文字列を*MELON*に置換できました。一括置換は非常に便利ですが、時には「確認しながら置換したい」と思う場合があると思います。次の手順で説明します。

確認しながらファイル全体を文字列置換

次に gc オプション付きで、確認しながら文字列置換してみましょう。

1. ファイル全体の lemon を@LEMON@ に置換してみましょう。以下のように操作してみてください。

```
:%s/lemon/@LEMON@/gc
```

2. 実行すると確認メッセージが出てきます。この時点では置換は実行されません。

```
lemon ORANGE BANANA *MELON* lemon
ORANGE banana *MELON* lemon ORANGE
BANANA *MELON* lemon orange banana
*MELON* lemon orange banana *MELON*
lemon orange banana *MELON* lemon
orange banana *MELON* lemon orange
banana *MELON* lemon orange banana
*MELON* lemon orange BANANA *MELON*
(略)
@LEMON@ に置換しますか? (y/n/a/q/l/^E/^Y)
```

3. y と入力すると置換が実行され、次の lemon を@LEMON@ に置換するか確認メッセージが出ます。

```
(略)
@LEMON@ に置換しますか? (y/n/a/q/l/^E/^Y) y
```

↓

6.5 置換と検索

```
@LEMON@ ORANGE BANANA *MELON* lemon  
ORANGE banana *MELON* lemon ORANGE  
BANANA *MELON* lemon orange banana  
*MELON* lemon orange banana *MELON*  
lemon orange banana *MELON* lemon  
orange banana *MELON* lemon orange  
banana *MELON* lemon orange banana  
*MELON* lemon orange BANANA *MELON*  
(略)  
@LEMON@ に置換しますか? (y/n/a/q/l/^E/^Y)
```

y を入力すると文字列置換して、n を入力すると文字列置換しません。該当文字列が見つからなくなるまで、確認メッセージが出ます。q を入力すると、文字列置換を中止できます。

6.6 6 章 章末テスト

vi を使用中の場合に、以下を実現するためのコマンドを答えなさい。

- (1) カレント行から 3 行をコピーする
- (2) 文書内の abc という語句を検索する。
- (3) 編集中のファイルを保存せずに強制終了する。
- (4) 文書の 100 行目に移動する。
- (5) 文書中の「高野豆腐」を「絹ごし豆腐」に置換する。ただし g オプションを付けて実行する。

第7章

管理者の仕事

本章は Linux でシステムの管理を行なう場合に使われるコマンドについて焦点を当てます。ユーザーの作成や削除、ユーザーパスワードの設定、グループを作成してユーザーを割り当てたりグループ設定の変更やグループの削除について理解を深めましょう。

この章の内容

- 7.1 グループとユーザ
 - 7.1.1 ユーザ
 - 7.1.2 ユーザの作成
 - 7.1.3 ユーザアカウントの変更
 - 7.1.4 ユーザの削除
 - 7.1.5 グループ
 - 7.1.6 グループの作成
 - 7.1.7 グループの登録情報の変更
 - 7.1.8 グループを削除
- 7.2 パスワードとパスワードファイル
 - 7.2.1 パスワードファイル (/etc/passwd)
 - 7.2.2 グループファイル (/etc/group)
 - 7.2.3 パスワード
 - 7.2.4 シャドウファイル (/etc/shadow)
- 7.3 用意されているユーザとグループ
 - 7.3.1 一般のユーザとグループ
 - 7.3.2 root ユーザ
 - 7.3.3 su コマンド
 - 7.3.4 root ユーザでコマンドを実行する sudo コマンド
- 7.4 章末テスト

7.1 グループとユーザ

Linux を利用するにはユーザアカウントが必要です。任意のユーザでログインすると、Linux を利用することができます。これはログインしたユーザが Linux というシステムの利用権限を持っているためです。グループを使えば複数のユーザを束ねることができます。

グループとユーザを適切に設定することで、ファイルやディレクトリ、任意のプログラムやシェルスクリプトなどを必要なユーザにのみ参照・編集する権限や実行する権限を与えることができます。

グループとユーザを作成・変更・削除するコマンドは管理コマンドなので、管理者である root ユーザで実行する必要があります。以降の章でも、多くのシステム管理用のコマンドを実行する際は root ユーザで実行してください。間違えて一般ユーザで実行しようとしても何も起こらないか、エラーが表示されて正常に処理を行なえません。

本教科書の実行例で冒頭が\$になっているものはユーザ権限でも実行可能ですが、#になっているコマンドは root 権限で実行する必要があります。学校などの環境によっては root ユーザを使えない場合も多々ありますのでご了承ください。root ユーザになるコマンドについては、この章で後述します。

7.1.1 ユーザ

メモリやファイルなどのさまざまな資源を利用するためにはユーザという最小単位で権限を定義できます。インストール時から用意されているユーザに加え、システム管理者が必要に応じてユーザを定義できます。ユーザの定義は/etc/passwd ファイルに記述します。Linux では/etc/passwd ファイルをエディタで直接編集する代わりに、useradd コマンドで新しいユーザを追加し、usermod コマンドでユーザの定義を変更し、userdel コマンドでユーザを削除することが推奨されています。

7.1.2 ユーザの作成

新しくユーザを作成するには useradd コマンドを使います。ユーザには数字であるユーザ ID を割り振ります。ユーザは必ずグループに所属します。作成したユーザをログインユーザとして使用する場合は、後述する passwd コマンドにてパスワードを登録する必要があります。

書式

```
useradd ユーザ名
```

オプション

-c コメント
コメント(文字列)を指定します。

-g グループ名

プライマリグループ名を指定します。グループ名は/etc/group ファイルで定義したグループ名です。

-G グループ名
補助グループを指定します。

-d
ホームディレクトリを指定します。

-s
シェルを指定します。デフォルトで/bin/bash が指定されているディストリビューションが多く、ログインしないユーザは nologin を指定するなどします。

-u ユーザ ID 番号
ユーザ ID 番号を指定します。

○実習: ユーザの作成

ユーザを作成したあと、ユーザが作成できたか home ディレクトリを確認します。

```
# useradd owl
(ユーザを作成)
# ls /home/
owl
(home ディレクトリを確認)
```

ユーザー owl を作成できました。

○実習: ユーザー ID を指定したユーザの作成

次にユーザー ID を指定してユーザーを作成してみましょう。

まずはユーザ ID 番号が 1001 のユーザが/etc/passwd ファイルに登録されているか調べます。users グループに所属するユーザ ID が 1001 の penguin ユーザを作ります。/etc/passwd ファイルに penguin ユーザが作成されたか grep コマンドで調べてください。

```
# grep 1001 /etc/passwd
(当てはまる行がなければユーザ ID は未登録)
# useradd -g users -u 1001 penguin
(ユーザを作成)
# grep penguin /etc/passwd
penguin:x:1001:100::/home/penguin:/bin/bash
(作成されたユーザを表示)
```

7.1.3 ユーザアカウントの変更

ユーザのアカウントを変更するには usermod コマンドを使います。

書式

```
usermod ユーザ名
```

オプション

-c コメント

コメント(文字列)を変更します。

-g グループ名

プライマリグループ名を変更します。グループ名は/etc/group ファイルで定義したグループ名です。

-G グループ名

補助グループを変更します。

-l ユーザ名

既存のユーザ名を変更します。

-u ユーザ ID 番号

ユーザ ID 番号を変更します。

○実習: ユーザアカウントのコメントの変更

penguin ユーザのコメントを調べてから、コメントを指定します。penguin ユーザのコメントが追加されているかを確認してください。

```
# grep penguin /etc/passwd
penguin:x:1001:100::/home/penguin:/bin/bash
(ユーザアカウントを表示)
# usermod -c "Linux Professional Institute Certification" penguin
(コメントを追加)
# grep penguin /etc/passwd
penguin:x:1001:100: Linux Professional Institute Certification:/home/penguin:/bin/bash
(ユーザアカウントを表示して変更内容を確認)
```

7.1.4 ユーザの削除

ユーザを削除するには userdel コマンドを使います。

書式

```
userdel ユーザ名
```

オプション

```
-r  
ホームディレクトリを削除します。
```

○実習: ユーザアカウントの削除

penguin ユーザの情報を表示してから、penguin ユーザを削除します。penguin ユーザが登録されているいないことを確認してください。

```
# grep penguin /etc/passwd  
penguin:x:1001:100:Linux Professional Institute Certification:/home/penguin:/bin/bash  
(登録されている penguin ユーザの情報を確認)  
# userdel penguin  
(penguin ユーザを削除)  
# grep penguin /etc/passwd  
(penguin ユーザの情報を確認)  
#  
(ユーザが削除されたので表示されません)
```

7.1.5 グループ

複数のユーザの権限をまとめて扱うためにグループを使います。ユーザは必ず 1 つ以上のグループに所属していて、主に所属するグループをプライマリグループと呼びます。最初から用意されているグループに加え、システム管理者が必要に応じてグループを定義できます。グループの定義は/etc/group ファイルに記述します。Linux では/etc/group ファイルをエディタで直接編集する代わりに、groupadd コマンドで新しいグループを追加し、groupmod コマンドでグループの定義を変更し、groupdel コマンドでグループを削除することが推奨されています。

7.1.6 グループの作成

新しくグループを作成するには groupadd コマンドを使います。グループには数字のグループ ID を割りります。

書式

```
groupadd グループ名
```

オプション

```
-g グループ ID 番号  
グループ ID 番号を指定します。
```

○実習: 新しく自分が使うグループの作成

グループ ID1001 が/etc/group ファイルに登録されているか調べます。グループ ID1001 の lpic グループを追加します。/etc/group ファイルに lpic グループが作成されたか grep コマンドで調べてください。

```
# grep 1001 /etc/group  
(グループ ID がすでに利用されていないことを確認)  
# groupadd -g 1001 lpic  
(新しい lpic グループを作成)  
# grep lpic /etc/group  
lpic:x:1001:  
(作成したグループを表示)
```

7.1.7 グループの登録情報の変更

グループの定義を変更するには groupmod コマンドを使います。

書式

```
groupmod [-g gid] [-n new-group-name] 変更対象のグループ
```

オプション

-n
既存のグループ名を変更する場合に指定します。

-g
既存のグループ ID を変更します。 100 未満のグループ ID はシステムで使われているため、指定できません。

実行例

```
$ groupmod -n penguin dolphin  
(dolphin グループを penguin グループに名前変更)  
$ groupmod -g 777 penguin  
(penguin グループ ID を 777 に変更)
```

○実習：登録したグループ名の変更

lpic グループの情報を表示します。lpic グループの名前を linux に変更したあと、linux グループの情報を確認してください。

```
# grep lpic /etc/group  
lpic:x:1001:  
(登録されているグループの表示)  
# groupmod -n linux lpic  
(グループ名の変更)  
# grep linux /etc/group  
linux:x:1001:  
(変更されたグループの表示)
```

7.1.8 グループを削除

グループを削除するには groupdel コマンドを使います。groupdel コマンドでは登録されているグループの情報を削除します。ユーザが所属していないグループのみ削除できます。

書式

```
groupdel グループ名
```

7.2 パスワードとパスワードファイル

○実習：登録したユーザの削除

linux グループを作成してから、linux グループを削除します。linux グループが削除されていることを確認してください。

```
# grep 1001 /etc/group  
("1001"を鍵として/etc/group 内を検索※)  
# groupadd -g 1001 linux  
(グループ ID1001 のグループ linux を作成)  
# grep linux /etc/group  
linux:x:1001:  
(作成したグループを確認)  
# groupdel linux  
(グループ linux を削除)  
# grep 1001 /etc/group  
(削除されたグループを確認)  
#  
(グループが削除されたので 1 行も表示されません)
```

※グループ ID 1001 のグループを作成するため

7.2 パスワードとパスワードファイル

グループの定義は/etc/group ファイル (グループファイル) に、ユーザの定義は/etc/passwd ファイル (パスワードファイル) に記述されています。ユーザを利用するにはパスワードが必要で、パスワードは/etc/shadow ファイル (シャドウファイル) に暗号化されて記録されます。パスワードの変更は passwd コマンドを使っておこないます。

7.2.1 パスワードファイル (/etc/passwd)

ユーザの情報は/etc/passwd ファイル (パスワードファイル) に保存され、1 行に 1 ユーザの情報を:で区切って記述します。パスワードファイルに登録された 1 ユーザの内容 (1 行) は次の様になります。

```
account:password:UID:GID:GECOS:directory:shell
```

※ GECOS = General Electric Comprehensive Operating System

従来はパスワードファイルに暗号化されたパスワードを記述していましたが、多くのディストリビューションはセキュリティを考慮してシャドウファイルにパスワードを記述しています。パスワードファイルはエディタで直接編集するべきではありません。useradd コマンドなどのコマンドを使って操作することが推奨されます。

表7.1 パスワードファイルの内容

項目	内容
account	そのシステムでのユーザ名。大文字を含まないようにする
password	以前はユーザの暗号化されたパスワード、現在は 'x' です
UID	ユーザ ID 番号
GID	ユーザが属するプライマリグループ ID 番号
GECOS	ユーザの名前またはコメントのフィールド
directory	ユーザのホームディレクトリ
shell	ログイン時に起動されるユーザのコマンドインターフェタ

7.2.2 グループファイル (/etc/group)

グループの情報は /etc/group ファイル (グループファイル) に保存され、1行に1グループの情報を:で区切って記述します。グループファイルに登録された1つのグループの内容(1行)は次の様になります。

```
group_name:password:GID:user_list
```

表7.2 グループファイルの内容

項目	内容
group_name	グループの名前。
password	以前は暗号化されたグループのパスワード、またはパスワードが不要なら空欄
GID	グループ ID 番号
user_list	グループに所属するユーザ名のリスト。それぞれのユーザ名はコンマで区切られる。

グループファイルはエディタで直接編集するべきではありません。groupadd コマンドなどのコマンドを使って操作することが推奨されます。

7.2.3 パスワード

ユーザの権限を使うにはユーザ名とパスワードを使って認証します。作成したユーザはパスワードを登録するとログインできるようになります。ユーザのパスワードを登録・変更するには passwd コマンドがあります。パスワードの登録にはパスワードが必要なので、初めてパスワードを変換するのはシステム管理のための root ユーザ(スーパーユーザ)である必要があります。

7.2 パスワードとパスワードファイル

書式

```
passwd [ユーザ名]
```

ユーザのパスワード登録と変更ユーザのパスワードを登録したり、変更します。

○実習: パスワードの設定

root ユーザで、penguin ユーザのパスワードを変更します。パスワードが変更できたら、コンソールから一回ログアウトし、再びログインをしてください。

```
# passwd penguin
ユーザー penguin のパスワードを変更。
新しいパスワード: xxxxxxxx
新しいパスワードを再入力してください: xxxxxxxx
passwd: 全ての認証トークンが正しく更新されました。
# exit
(root ユーザをログアウト)

CentOS release 6.3 (Final)
Kernel 2.6.32-279.1.1.el6 on an i686
localhost login: penguin
(ユーザ名を入力)
Password: XXXXXXXX
(パスワードを入力)
Last login: Wed Jul 18 10:10:10 from 192.168.1.100
$
```

パスワードは複雑なものを設定するように心がけてください。英数文字だけでなく小文字と大文字を混ぜたり記号をいれるなど、他人から推測されにくい、適切なパスワードを設定してください。

7.2.4 シャドウファイル (/etc/shadow)

ユーザのパスワードはパスワードファイルではなく、シャドウファイル (/etc/shadow) に保存されます。シャドウファイルに登録された 1 つのユーザ (1 行) の内容は次の様になります。

```
account:password:last_changed:may_be_changed:must_be_changed:warned:expires:disabled:reserved
```

シャドウファイルはエディタで直接編集するべきではありません。日付に使われている基準の 1970 年 1 月 1 日は Linux システムが基準としている時間です。

表7.3 シャドウファイルの内容

項目	内容
account	ユーザ名
password	暗号化されたパスワード
last_changed	1970年1月1日から、最後にパスワードが変更された日までの日数
may_be_changed	パスワードが変更可能となるまでの日数
must_be_changed	パスワードを変更しなくてはならなくなる日までの日数
warned	パスワード有効期限が来る前に、ユーザが警告を受ける日数
expires	パスワード有効期限が過ぎ、アカウントが使用不能になるまでの日数
disabled	1970年1月1日からアカウントが使用不能になる日までの日数
reserved	予約フィールド

7.3 用意されているユーザとグループ

Linuxはインストールしてすぐにそのシステムを利用できるように、ユーザとそのユーザが所属するグループが用意されています。ユーザやグループは必要に応じて複数追加できます。

7.3.1 一般のユーザとグループ

Linuxにログインするにはアカウントが必要です。アカウントを作成するとユーザー名と同様の名前のグループが作られ、ユーザーはそのユーザーグループに所属しているとシステムに登録されます。

グループは複数のユーザをまとめるためにあります。個別のユーザを所属部署などの単位でグループ化することができます。グループを使うことでシステム上にあるディレクトリのアクセス権を設定して、特定のグループに属したユーザーのみアクセスできるディレクトリを作成するといった使い方や、特定グループに属しているユーザのみ後述のrootユーザになることができる運用が可能になります。

7.3.2 rootユーザ

rootユーザはシステム設定の変更や、プログラムのインストールや削除、ユーザを作成・削除する事ができる、利用に制限がない特別なユーザです。rootユーザはアクセス権に関係なくすべてのユーザーのディレクトリへのアクセス、コンテンツの読み書きが行なえるなどの点で一般ユーザとは異なります。

このようにrootユーザでログインできれば全ての操作を行なうことができてしまうため、rootユーザのアカウントは厳格に管理する必要があります。

7.3.3 su コマンド

su コマンドはすでに別のユーザでログインしているユーザが、一時的に他のユーザになるためのコマンドです。su コマンドを実行する際、オプションとしてユーザーを指定しない場合は root ユーザでシェルを起動します。

オプションを付けずに su コマンドを実行した場合はカレントディレクトリを変更せずに root ユーザでログインします。カレントディレクトリを root のホームディレクトリに変更してログインするには「su -」、もしくは「su - root」と実行することでカレントディレクトリを変更した上で root ユーザでログインできます。

root ユーザでログインすると、システム管理用のコマンドを実行することができます。複数人で Linux システムを管理している場合は、root ユーザで直接ログインして作業をすると、root ユーザとしての履歴だけが残り、誰がどんな作業をしたのかの履歴が残りません。一般ユーザから root ユーザへ切り替えると、root ユーザで作業を開始した時間などは直ぐにわかります。安全や管理を考えると、一般的のユーザでログインしてから root ユーザの権限を取得し、システム作業することが望ましいでしょう。

書式

```
su  
su - [ユーザ]
```

オプション

```
su - (もしくは su - root)  
root ユーザになることができます。
```

```
su - user  
指定したユーザになることができます。
```

○実習: root ユーザになる su コマンドの実行

それではユーザから root ユーザになってみましょう。ユーザ権限では見られないログファイルを root 権限になると閲覧できるのを確認してみましょう。

```
$ cat /var/log/messages  
cat: /var/log/messages: 許可がありません  
  
$ su -  
(root ユーザへ移行)  
パスワード: xxxxxxx  
# cat /var/log/messages
```

```
Mar 20 08:51:46 localhost syslogd 1.4.1: restart.  
Mar 20 08:51:46 localhost kernel: klogd 1.4.1, log source = /proc/kmsg started.  
(略)
```

7.3.4 root ユーザでコマンドを実行する sudo コマンド

sudo コマンドを使えば、スーパーユーザ（root）権限でコマンドを実行できます。普段作業するときはユーザ権限で行ない、必要に応じて sudo を使いコマンドを実行することで、su コマンドでユーザを root に切り替えることなく、root 権限が必要な設定やプログラムを実行することができます。

-u オプションを付けて sudo コマンドを実行すると、任意のユーザーでコマンドを実行できます。オプションを付けないで sudo コマンドを実行した場合は root 権限でコマンドを実行します。

CentOS では初期設定のままでは sudo コマンドは利用できません。sudo コマンドを使うにはユーザを wheel グループというスーパーユーザ（root）特権を持つグループに登録する必要があります。

sudo の設定は /etc/sudoers ファイルを編集することでユーザが sudo コマンドを利用できるようになります。/etc/sudoers ファイルは visudo コマンドを実行すると編集できます。

書式

```
sudo コマンド  
スーパーユーザ (root) 権限でコマンドを実行
```

例 1:

```
sudo cat /var/log/message
```

オプション

```
-u ユーザ  
指定したユーザでコマンドを実行します。
```

○実習: 一般ユーザから管理コマンドの実行

sudo を使いたい penguin ユーザを wheel グループの所属にします。visudo コマンドで /etc/sudoers ファイルを編集して wheel グループの定義を有効にします。su コマンドで penguin ユーザに移行してから sudo コマンドを試してください。

```
$ visudo  
(ユーザ penguin で visudo を実行)  
visudo: /etc/sudoers: Permission denied
```

7.3 用意されているユーザとグループ

```
visudo: /etc/sudoers: Permission denied
(管理権限を持たないユーザでは visudo を実行できない)
# su -
パスワード: xxxxxxxx
# usermod -G wheel penguin
(wheel グループにユーザ penguin を追加)
# vigr
(wheel グループにユーザ追加されたか確認)
wheel:x:10:root
↓
wheel:x:10:root,penguin

# visudo
(wheel グループを有効に設定)
# %wheel      ALL=(ALL)      ALL
↓
%wheel      ALL=(ALL)      ALL
# su - penguin
(root ユーザが su するときは、パスワード入力不要)
$ sudo visudo
(ユーザ penguin は管理グループに所属するユーザなので visudo を実行可能)
[sudo] password for penguin: xxxxxxxx
(コマンドを実行するユーザのパスワードを入力する)
```

/etc/sudoers ファイルにある以下のオプションを有効にすると、wheel グループに所属するユーザが sudo コマンドを実行した場合、パスワードを聞かれることなくコマンドを実行できるようになります。この設定はデフォルトでは無効化されています。

```
# %wheel      ALL=(ALL)      NOPASSWD: ALL
```

7.4 7 章 章末テスト

- (1) useradd で user を作りましたが、ログインすることができません。ログインできるようにするには何が必要か答えなさい。
- (2) ユーザ user2 を削除する場合のコマンドを答えなさい。
- (3) グループ penguin を作成する場合のコマンドを答えなさい。
- (4) すでに作成済みのユーザ penguin のパスワードを変更する場合のコマンドを答えなさい。
- (5) su コマンドに-オプションを付けない場合と付ける場合の違いは何か、答えなさい。

第8章

ユーザ権限とアクセス権

ディレクトリとファイルの所有者と所有権について理解します。ファイル所有者と所有権を設定・変更するコマンドとして chown コマンド、chmod コマンド、chgrp コマンドを説明します。

この章の内容

- 8.1 ファイルの所有者と所有グループ
 - 8.1.1 所有者の変更
 - 8.1.2 所有グループの変更
- 8.2 ファイルとアクセス権
 - 8.2.1 ファイルに設定できるアクセス権
 - 8.2.2 ファイルのモード変更
 - 8.2.3 ファイル作成のモード
- 8.3 章末テスト

8.1 ファイルの所有者と所有グループ

ファイル作成者のユーザ ID とグループ ID がファイルの所有者と所有グループとなります。所有者と所有グループはファイルの情報として重要な属性です。所有者は chown コマンドで変更でき、所有グループは chgrp コマンドで変更できます。

8.1.1 所有者の変更

ファイルの所有者を変更するには chown コマンドを使います。

書式

```
chown ユーザ [. グループ] ディレクトリ  
chown ユーザ [. グループ] ファイル
```

ユーザとグループを変更するには、root ユーザである必要があります。ディレクトリとファイル

8.1 ファイルの所有者と所有グループ

は区別なく変更できます。ユーザとグループの区切りに:を使うことも可能です。

オプション

-R
ディレクトリを対象にします。ディレクトリの中のディレクトリやファイルを再帰的にたどって変更します。

○実習: 一般ユーザと root ユーザでファイル所有者を変更

まず、一般ユーザでファイルを作成し、所有者の変更を試みます。その後、root ユーザに移行し、所有者の変更を試みます。ファイルの所有者が変更されているか確認してください。

```
$ touch user
(ファイルを作成)
$ ls -l user
(ファイルの所有者と所有グループを表示)
-rw-r--r-- 1 penguin lpic 0 2月 9 12:39 user
(現在の所有はユーザ penguin/lpic グループ)
$ chown nobody user
(一般ユーザでファイルの所有者を変更)
chown: `users' の所有権を変更中: 許可されていない操作です
(一般ユーザで変更不可エラー)
$ su
(root ユーザへ移行)
パスワード: xxxxxxxx
(root ユーザのパスワード入力)
# chown nobody user
(nobody ユーザへ所有者を変更)
# ls -l user
-rw-r--r-- 1 nobody lpic 0 2月 9 12:39 user
(変更後の所有者を表示)
```

上の実行例では、所有者が penguin、所有グループが lpic になっていますが、実際の環境ではコマンドを実行しているあなたのユーザーとグループになります。

8.1.2 所有グループの変更

ファイルの所有グループを変更するには chgrp コマンドを使って行ないます。

書式

```
chgrp グループ ディレクトリ
chgrp グループ ファイル
```

グループを変更できます。ディレクトリとファイルは区別なく変更できます。

オプション

-R
ディレクトリを対象にします。ディレクトリの中のディレクトリやファイルを再帰的に変更します。

○実習: 一般ユーザと root ユーザでファイル所有グループを変更

一般ユーザでファイルを作成し、所有グループの変更を試みます。その後、root ユーザに移行し、所有グループの変更を試みます。ファイルの所有グループが変更されているか確認してください。

```
$ touch groups
(ファイルを作成)
$ ls -l groups
(ファイルの所有グループを表示)
-rw-r--r-- 1 penguin lpic 0 2月 9 12:39 groups
(現在の所有グループ)
$ chgrp nobody groups
(一般ユーザでファイルの所有グループを変更)
chgrp: changing group of `groups': 許可されていない操作です
(一般ユーザで権限の変更が不可であるとエラー表示)
$ su
(root ユーザへ移行)
パスワード: xxxxxxxx
(root ユーザのパスワード入力)
# chgrp nobody groups
(nobody グループへ所有グループを変更)
# ls -l groups
-rw-r--r-- 1 penguin nobody 0 2月 9 12:39 groups
(変更後の所有グループを表示)
```

8.2 ファイルとアクセス権

ファイルはファイルを所有するユーザ、ファイル所有グループからファイル所有者を除いたユーザ、他のユーザの3つのレベルで権限を設定できます。ファイルにはユーザで分けた3つのレベルごとに読み、書き、実行の3つの権限があります。ファイルのモードを変更するには chmod コマンドを使います。

8.2.1 ファイルに設定できるアクセス

ファイルのモードとしては読み書き実行の3つの権限があります。所有ユーザ、所有グループで所有者以外のユーザ、所有グループ以外のユーザごとに3つの権限を設定できます。ls コマンドに-l オプションを付けて表示される1つ目のカラムがファイルのモードを示しています。モードを表す1つ目のカラムは次の様になります。



表 8.1 ファイル種別の r,w,x の意味

項目	内容
r	読み込み
w	書き込み
x	実行またはディレクトリの移動

rwx は、ユーザとグループとその他の 3 つに対して指定できます。r が表示されればファイルの読み込みが可能で、w が表示されればファイルの書き込みが可能です。x が表示されればファイルをプログラムとして実行できるか、または、ディレクトリであればディレクトリへ移動できます。

○実習: ファイル所有グループの確認

それでは、システムファイルのファイルモードを確認してみましょう。

```

$ ls -l .bashrc
-rw-r--r-- 1 penguin linux 124 2月 6 02:44 .bashrc
(所有者が読み書き、所有グループが読み、その他が読み)
$ ls -l /usr
合計 272
drwxr-xr-x 3 root root 4096 2月 6 01:45 X11R6
drwxr-xr-x 2 root root 4096 2月 8 13:07 arc
drwxr-xr-x 2 root root 69632 2月 9 04:02 bin
drwxr-xr-x 2 root root 4096 3月 30 2007 etc
drwxr-xr-x 2 root root 4096 3月 30 2007 games
drwxr-xr-x 84 root root 12288 2月 6 02:04 include
drwxr-xr-x 6 root root 4096 11月 11 11:39 kerberos
drwxr-xr-x 108 root root 69632 2月 7 11:41 lib
drwxr-xr-x 13 root root 4096 2月 9 04:02 libexec
drwxr-xr-x 11 root root 4096 2月 6 01:43 local
drwxr-xr-x 2 root root 16384 2月 7 11:41 sbin
drwxr-xr-x 226 root root 12288 2月 6 02:05 share
drwxr-xr-x 5 root root 4096 2月 6 08:52 src
lrwxrwxrwx 1 root root 10 2月 6 01:43 tmp -> ../var/tmp
(/usr ディレクトリ以下のディレクトリは全てのユーザが読み込み可能で所有者 (root) が書き込み可能)
  
```

8.2.2 アクセス権の変更

ファイルのアクセス権を変更するには chmod コマンドを使います。

書式

```
chmod モード [, モード]... ディレクトリ
chmod モード [, モード]... ファイル
chmod 8進数表記のモード ディレクトリ
chmod 8進数表記のモード ファイル
```

ファイルのモードを所有ユーザと所有グループとそれ以外のユーザについて設定します。モード指定の書き方で次の2通りの記述方法があります。

- モードの書式を複数書き、カンマで区切って指定。
- 8進数3桁で各ユーザのレベルを指定。

オプション

-R
ディレクトリを対象にします。ディレクトリの中のディレクトリを再帰的に(ディレクトリの中にディレクトリがあれば、中のディレクトリを全てたどって)変更します。

パーミッション	ユーザー			グループ			その他		
	r	w	x	r	w	x	r	w	x
8進数	4	2	1	4	2	1	4	2	1
設定値	合計値			合計値			合計値		

モードは次の様に u(所有ユーザ)、g(所有グループ)、o(他のユーザ)に対して、r(読み)、w(書き)、x(実行またはディレクトリの変更)を設定したり(=)、加えたり(+)、取り消したり(-)します。u, g, o の全てに同じ権限を指定するときは a を指定します。

○実習: ファイルモードの変更

ユーザのファイルのモードを確認します。ファイルのモードを変更し、モードを再確認してください。`-rw-r-r-`を`rw-rw-r-`としたい場合は、グループの w(書き込み権限)を加えたいので`g+w`とします。`-rw-rw-r-`を`-w-rw-rw-`としたい場合は、ユーザの r(読み込み権限)を取り除き、その他の

8.2 ファイルとアクセス権

w(書き込み権限)を加えたいので u-r,o+w とします。-w-rw-rw-を-w-rwxrwx としたい場合は、グループとその他に x(実行権限)を加えたいので go+x とします。

```
$ touch chownfile
(ファイルを作成)
$ chmod u+rwx,go+rwx chownfile
(ファイルモードを「rw-r--r--」に変更)
$ ls -l chownfile
-rw-r--r-- 1 penguin lpic 124 3月 27 19:09 chownfile
(ファイルモードを表示)

$ chmod g+w chownfile
(グループに w(書き込み権限)を追加)
$ ls -l chownfile
-rw-rw-r-- 1 penguin lpic 124 3月 27 19:09 chownfile
(ファイルモードを表示)
$ chmod u-r,o+w chownfile
(ユーザから r(読み込み権限)を除去、その他に w(書き込み権限)を追加)
$ ls -l chownfile
(--w-rw-rw- 1 penguin lpic 124 3月 27 19:09 chownfile
(ファイルモードを表示)
$ chmod go+x chownfile
(グループとその他に x(実行権限)を追加)
$ ls -l chownfile
--w-rwxrwx 1 penguin lpic 124 3月 27 19:09 chownfile
(ファイルモードを表示)
```

○実習: ファイルモードを 8 進数で変更

ユーザのファイルのモードを確認します。ファイルのモードを変更し、モードを再確認してください。-rw-rw-r-としたい場合は 8 進数の 664 を指定し、-w-rw-rw-としたい場合は 8 進数の 266 を指定し、-w-rwxrwx としたい場合は 8 進数の 277 を指定してください。

```
$ touch chownfile
(ファイルを作成)
$ chmod u+rwx,go+rwx chownfile
(ファイルモードを「rw-r--r--」に変更)
$ ls -l chownfile
-rw-r--r-- 1 penguin lpic 124 3月 27 19:09 chownfile
(ファイルモードを表示)

$ chmod 664 chownfile
(ファイルモードを 664 に変更)
$ ls -l chownfile
-rw-rw-r-- 1 penguin lpic 124 3月 27 19:09 chownfile
(ファイルモードを表示)

$ chmod 266 chownfile
(ファイルモードを 266 に変更)
$ ls -l chownfile
```

```
--w-rw-rw- 1 penguin lpic 124 3月 27 19:09 chownfile
(ファイルモードを表示)

$ chmod 277 chownfile
(ファイルモードを 277 に変更)
$ ls -l chownfile
--w-rwxrwx 1 penguin lpic 124 3月 27 19:09 chownfile
(ファイルモードを表示)
```

モードの指定に setuid ビットと setgid ビットと sticky(スティッキー) ビットという、特殊な属性があります。setuid ビットあるいは setgid ビットが付いたプログラムを実行すると、ファイル所有者あるいは所有グループの権限で実行されます。具体的には、root ユーザ所有で setuid ビットがセットされたプログラムは、一般ユーザが実行した場合でも root ユーザが実行した場合と同じ動作をします

sticky ビットが付いたディレクトリ内のファイルは所有者以外が削除できなくなります（Linux では、ファイルについてのスティッキービットは無視されます）。Linux では一般的に /tmp ディレクトリにスティッキービットが付与されています。

○実習: ファイルモードで setuid/setgid/sticky の変更・確認

ユーザのファイルのモードで、setuid ビットと setgid ビットと sticky ビットを確認します。

```
$ touch idbitfile
(ファイルを作成)
$ chmod u+rwx,go+rwx idbitfile
(ファイルモードを「rw-r--r--」に変更)
$ ls -l idbitfile
-rw-r--r-- 1 penguin lpic 0 3月 28 07:58 idbitfile
(ファイルモードを表示)

$ chmod u+s idbitfile
(setuid ビットを追加)
$ ls -l idbitfile
-rwSr--r-- 1 penguin lpic 0 3月 28 07:58 idbitfile
(ファイルモードを表示)

$ chmod u-s,g+s idbitfile
(setuid ビットを除去、 setgid ビットを追加)
$ ls -l idbitfile
-rw-r-Sr-- 1 penguin lpic 0 3月 28 07:58 idbitfile
(ファイルモードを表示)

$ chmod +t idbitfile
(sticky(スティッキー) ビットを追加)
$ ls -l idbitfile
-rw-r-Sr-T 1 penguin lpic 0 3月 28 07:58 idbitfile
(ファイルモードを表示)
```

8.2.3 ファイル作成のモード

ファイルを新規に作成すると、ユーザーごとに規定されたパーミッションである 644、もしくは 664 といったパーミッションが設定されてファイルが作成されます。umask コマンドを使うことで、指定したパーミッションでファイルを作成するように制限できます。

書式

```
umask [8進数のモードのマスク値]
```

現在のマスク値を表示

umask コマンドに続けてマスク値を指定しなかった場合、現在のマスク値を表示できます。

マスク値を変更

umask コマンドに続けてマスク値を指定すると、コマンドを実行後に作成されるファイルが指定したパーミッションで作成できます。umask コマンドは許可しないビットを指定します。umask コマンドで指定するマスク値は、すべてのユーザーが読み書きできるパーミッション 666 から、設定したいパーミッションを引き算することでマスク値を求めることができます。

	ユーザー			グループ			その他		
デフォルトのモード	r	w	-	r	w	-	r	w	-
デフォルト 8進数	4	2	-	4	2	-	4	2	-
umask 値		0			?			?	
umask 値 8進数			-		2	-		2	-
8進数	4	2	-	4	-	-	4	-	-
設定値	6			4			4		

○実習: umask を変更してファイルを作成

umask を変更後にファイルを作成したときに、作成したファイルのパーミッションが違っていることを確認します。

```
$ umask  
0022  
(現在のマスク値を表示)  
  
$ touch umask0022  
(ファイルを作成)  
$ umask 070  
(マスク値を 070 に変更)  
$ touch umask0070  
(ファイルを作成)  
$ umask 072  
(マスク値を 072 に変更)  
$ touch umask0072  
(ファイルを作成)  
$ ls -l umask00*  
-rw-r--r-- 1 penguin lpic 0 3月 22 13:49 umask0022  
-rw----rw- 1 penguin lpic 0 3月 22 13:50 umask0070  
-rw----r-- 1 penguin lpic 0 3月 22 13:50 umask0072  
(作成したファイルの権限を確認)
```

touch コマンドはモードが 0666 のファイルを作ろうとしますが、umask によりマスクされるために上記のモードのようなファイルができます。-S オプションを付けると、モードを表示または設定するとき、8進数ではなく意味がわかりやすい形式を使用できます。

```
$ umask -S  
u=rwx,g=rx,o=rx  
(現在のマスク値を表示)  
  
$ touch umask0022  
(ファイルを作成)  
$ umask -S u=rw,g=,o=rw  
u=rw,g=,o=rw  
(マスク値を 070 に変更)  

```

umask コマンドによるモードの制限は、umask コマンドを実行したシェル内でしか有効でないことに注意してください。デフォルトのモードは.bashrc 等のログインシェル内で設定することができます。

8.3 8章 章末テスト

(1) 以下のコマンドを実行した場合の実行結果を選びなさい。

```
# umask 022
# touch test
# chgrp nobody test
# ls -l test
```

1. -rw-r--r-. 1 nobody root 18 5月 29 18:51 2012 test
2. -rw-rw-r-. 1 root nobody 18 5月 29 18:51 2012 test
3. -rw-r--r-. 1 root nobody 18 5月 29 18:51 2012 test
4. -rw-r--r-. 1 nobody test 18 5月 29 18:51 2012 test

(2)chownfile のファイルモードを 755 にするためのコマンドを記述しなさい。

(3)chownfile のファイルモードを 644 にするためのコマンドを記述しなさい。

(4)umask 070 にした後 umask ファイルを作成した場合に設定されるファイルモードは以下のどれでしょうか？

(5)umask 022 にした後 umask ファイルを作成した場合に設定されるファイルモードは以下のどれでしょうか？

1. -rw--rw-. 1 user user 0 5月 29 19:22 2012 umask
2. -rw--r-. 1 user user 0 5月 29 19:22 2012 umask
3. -rw-----. 1 user user 0 5月 29 19:23 2012 umask
4. -rw-r--r-. 1 user user 0 5月 29 19:24 2012 umask

第9章

シェルスクリプト

シェルにより入力するコマンドは、シェルスクリプトにより順次実行可能な形式になります。それを応用することにより、コマンド入力による作業を自動化することが可能です。その際に必要な技術である、シェルスクリプトを学習しましょう。

この章の内容

- 9.1 シェルとシェルスクリプト
- 9.2 プログラミング
 - 9.2.1 プログラムの例
 - 9.2.2 プログラムの要素
- 9.3 シェルスクリプトの作成
 - 9.3.1 シェルスクリプトの作成
 - 9.3.2 変数
 - 9.3.3 echo コマンド
 - 9.3.4 read コマンド
 - 9.3.5 シェル変数
 - 9.3.6 環境変数
 - 9.3.7 コメント
 - 9.3.8 引用符
 - 9.3.9 引数
 - 9.3.10 shift 文
 - 9.3.11 エスケープシーケンス
 - 9.3.12 source コマンド
 - 9.3.13 バックスラッシュ ¥
- 9.4 条件分岐
 - 9.4.1 if 文
 - 9.4.2 ファイルの属性比較
 - 9.4.3 複数の条件を重ねる
 - 9.4.4 一対多の条件分岐

この章の内容

- 9.5 繰り返し
 - 9.5.1 for 文
 - 9.5.2 while/until 文
 - 9.5.3 select 文
 - 9.5.4 繰り返しの制御
- 9.6 サブルーチン
 - 9.6.1 関数
 - 9.6.2 return 文
- 9.7 実際のシェルスクリプト
 - 9.7.1 起動スクリプト
 - 9.7.2 関数のシェルスクリプト
- 9.8 デバッグ
 - 9.8.1 bash コマンド
- 9.9 章末テスト

9.1 シェルとシェルスクリプト

シェルのプログラミングを始める前に、シェルとシェルスクリプトについて学習しましょう。

シェル

第1章でカーネルはOSの基本部分であり、ハードウェアを操作するほかさまざまな機能を司つていると説明しました。

シェルとは、貝殻(=Shell)の事を指しています。カーネルが提供する機能を操作する際に、OSと対話型で操作する必要があります。OSの、特にカーネル部分を包み込んでいる機能ということからシェルと名づけられて、対話機能を提供します。シェルはコマンドの入力を受け付けそのコマンドを実行し、入力したユーザに対しその結果を返す役割があります。

シェルスクリプト

/etcディレクトリと/homeディレクトリを圧縮して外部サーバーにコピーするといった流れを考えてみましょう。実際に実行するコマンドを列挙するとたとえば以下のような方法が考えられると思います。

```
# tar cvzf 120626-etc.tar.gz /etc
# tar cvzf 120626-home.tar.gz /home
# scp 120626-etc.tar.gz root@backup.local.example.com:~/backup
# scp 120626-home.tar.gz root@backup.local.example.com:~/backup
```

これを実行ごとに入力するのは面倒です。さらに1つのコマンドの処理が終わるまで次のコマンドを実行できないため効率がよくありませんし、そもそも実行する際に端末の前に人手が必要です。

このような繰り返し行なう処理を自動化するための手段としてシェルスクリプトを書いて実行させる方法があります。

先ほどの一連処理を system-backup.sh という名前のファイルを作成して書き込み、system-backup.sh を実行することでファイルに書き込んだコマンドを順に実行します。

```
#!/bin/bash

tar cvzf 120626-etc.tar.gz /etc
tar cvzf 120626-home.tar.gz /home
scp 120626-etc.tar.gz root@backup.local.example.com:~/backup
scp 120626-home.tar.gz root@backup.local.example.com:~/backup
```

9.2 プログラミング

コンピュータに対して、指示を与え順番に実行させる機能を、プログラムといいます。そのプログラムを作成することをプログラミングといいます。今まで学習したコマンドを、条件分岐や繰り返しなど制御機能を加え実行することができます。これをシェルスクリプトといいます。

9.2.1 プログラムの例

プログラムという言葉を聞いて、最初に思いつくのは、運動会や演劇会のプログラムかもしれません。また、新聞に掲載されているテレビの番組表もプログラムです。何かが順序立てて進むものを、プログラムといいます。では、実際にプログラムを作成するとなると、どのような例があるでしょうか？簡単な例だと、テレビ番組の録画予約があります。開始時刻から終了時刻とチャンネルを指定し、画質を指定したりもします。少し複雑な例だと、エアコンの設定で「**度に達したら冷房を入れて、**度を下回ったら冷房を止める」などの条件を設定するというのも、一種のプログラムです。

9.2.2 プログラムの要素

プログラミングには、どの種類のプログラムでもいくつかの重要な構成要素があります。プログラムの要素は主に以下の4つです。

- 順次実行
- 条件分岐
- 繰り返し
- サブルーチン

これらをしっかりマスターしてください。

9.3 シェルスクリプトの作成

それでは実際にシェルスクリプトを作成してみましょう。ここではシェルスクリプトを作成して実行する手順を紹介します。

9.3.1 シェルスクリプトの作成

シェルスクリプトは、テキストで記述します。vi等のテキストエディタを利用してください。lsdate.shというファイルに、lsとdateコマンドを実行するシェルスクリプトを記述してみましょう。

実行例

```
$ vi lsdate.sh  
(lsdate.shというファイルを作ります)
```

vi 上で以下のようなスクリプトを記述して、保存終了してください。

```
#!/bin/bash
ls
date
```

シェルの指定

1行目に#!/bin/bashと記述しました。ファイルの1行目には、利用するシェルの種類とそのコマンド位置を記述します。シェルには数種類ありますが、今回はbashを使用します。

1行目に利用するシェルを指定、2行目以降に実行するコマンドを1行ずつ入力していきます。

パーミッションの変更

作成したシェルスクリプトを実行するには、パーミッションを変更してファイルの実行権限をつける必要があります。例のようにシェルスクリプトに実行権限がついているか確認してみましょう。

lsコマンドでファイルのアクセス権限を確認します。

実行例

```
$ ls -l lsdate.sh
-rw-rw-r--. 1 tooyama tooyama 21 Jun 6 09:51 lsdate.sh
```

実行権限を付与するためにchmodコマンドを使います。

実行例

```
$ chmod u+x lsdate.sh
$ ls -l lsdate.sh
-rwxrwxr--. 1 tooyama tooyama 21 Jun 6 09:51 lsdate.sh
```

これで所有者に実行権限が与えられました。実行権限をシェルスクリプトにつけたら早速実行してみましょう。

実行例

```
$ ./lsdate.sh
Desktop diff2           ls-l-output          ls-usr-bin      touched-file
cat-output     hosts.bak      ls-l-output-second    lsdate.sh      uniq-sample
diff1         hosts.new2      ls-output          score          (ls の実行結果)
Fri Jun 6 09:55:05 JST 2008          (date の実行結果)
```

./というのはパス指定です。意味は「カレントディレクトリにある(lsdate.shを実行せよ)」ということです。lsやcpを実行するときは、パスが通っているので、このようなパス指定は必要ありません。今回は、カレントディレクトリにある特定のシェルスクリプトを実行させるために、パス指

9.3 シェルスクリプトの作成

定として用いました。lsdate.sh の中に記述された、ls コマンドと date コマンドが順に実行されたことがわかります。

9.3.2 echo コマンド

echo コマンドは引数で与えた文字列を標準出力に出力するコマンドです。変数は、\$をつけて出力することができます。

書式

```
echo [オプション] 文字列
```

オプション

-n
改行を抑制します。通常の出力は改行されますが、このオプションがあると改行されません。

```
$ echo Message test  
Message test  
(echo コマンドで指定された文字列を表示する)
```

9.3.3 変数

プログラミングをする上で、非常に重要な考え方が、変数です。変数は、簡単に言うと「ハコ」で、中に数値や文字列が入ります。ちょうど、中学生の数学のときに習った x や y がそれにあたります。シェルスクリプトプログラミングでは、変数に数値や文字列を代入し、それを利用することができます。変数の代入は、=を使って行なうことができます。

○実習：シェル変数の作成

シェル関数 abc に値を設定し、echo コマンドで内容を確認してみましょう。

```
$ abc=123  
$ echo $abc  
123  
(abc の内容を表示する)
```

変数 abc に 123 を代入しました。bash では、一次元の配列変数を使用することができます。要素は角括弧 [] で囲みます。配列変数の内容を表示する場合は、\$の後ろに波括弧{}で配列変数を囲みます。

```
$ abc[0]=123
$ abc[1]=456
$ echo ${abc[0]}
123
(abc[0] の内容を表示する)
$ index=1
$ echo ${abc[$index]}
456
(abc[1] の内容を変数を使って表示する)
```

シェル変数と環境変数

シェルには、2種類の変数があります。シェル変数と環境変数です。シェル変数は、実行しているシェルの内部でのみ有効です。環境変数は、そこから実行されたコマンド内でも有効になります。環境変数は、シェル変数から作成できます。

○実習：環境変数の作成

export コマンドを使って、環境変数を作成してみましょう。

```
$ export abc
(シェル変数 abc を環境変数 abc にする)
$ export xyz=234
(環境変数 xyz を作成し 234 を代入する)
```

一つ目のコマンドでは abc という環境変数を作成しています。abc には値を代入していません。二つ目のコマンドでは xyz という環境変数を作成しています。xyz には 234 を代入しています。

次に、2つのスクリプト BBB.sh と CCC.sh を使って、シェル変数と環境変数の動作の違いについて確認してみましょう。

実行例

```
$ cat BBB.sh
#!/bin/bash
xxx=123
export yyy=234
./CCC.sh
(シェル変数 xxx と環境変数 yyy に値を代入して CCC.sh を実行する)

$ cat CCC.sh
#!/bin/bash
echo xxx=$xxx
echo yyy=$yyy
(xxx と yyy の値を表示する)

$ ./BBB.sh
xxx=
yyy=234
```

9.3 シェルスクリプトの作成

```
$
```

(BBB.sh の中に CCC.sh を動作させると、 xxx は引き継がれないが、 yyy は引き継がれた)

このシェルスクリプトを実行した時、シェルスクリプト CCC.sh の中で xxx の値は表示されません。シェル変数は引き継がれないからです。一方で、yyy は環境変数なので CCC.sh まで引き継がれるため、値が表示されます。

9.3.4 read コマンド

read コマンドは、標準入力からデータを読み込みます。すでに変数にデータが入っていた場合、新しいデータに上書きされます。

書式

```
read 変数名
```

○実習: read コマンドの実行

read コマンドを使ってシェル変数の内容を変更してみましょう。

```
$ echo $abc
123
(シェル変数 abc の中身が表示されます)
$ read abc
aaabbccccc
(何か入力します)
$ echo $abc
aaabbccccc
(シェル変数 abc の中身が入れ替わっています)
```

9.3.5 シェル変数

シェル変数の一覧を表示する場合は、set コマンドを利用します。また削除する場合は、unset を利用します。

○実習: シェル変数一覧の表示と削除

シェル変数の一覧表示と削除を行なってみましょう。

```
$ set
BASH=/bin/bash
```

```
BASHOPTS=checkwinsize:cmdhist:expand_aliases:extquote:force_fignore:hostcomplete:intera
BASH_ALIASES=()
abc=aaabbbccc
$ unset abc
(シェル変数 abc が削除されます)
```

9.3.6 環境変数

現在の環境変数の一覧を表示する場合は、env コマンドを利用します。また登録済みの環境変数を削除するときは、unset コマンドを利用します。

○実習: 環境変数一覧の表示と削除

環境変数の一覧表示・削除を行なってみます。

```
$ env
HOSTNAME=host1.alpha.jp
TERM=xterm
SHELL=/bin/bash
HISTSIZE=1000
abc=aaabbbcc
(略)
$ unset abc
(環境変数 abc の内容を削除する)
```

9.3.7 コメント

コメントとは、プログラム上に書く注釈のことです。シェルの場合は#で始まる行がコメントとして認識され、プログラムの実行時はコメントは無視されます。コメントは多くの場合、プログラマーが記述したプログラムがどういった処理をするのかを記述したり、一時的に特定の処理を無効化（コメントアウト）させる場合に利用します。

書式

```
#!/bin/bash
# assign xyz to variable abc and output it
abc=xyz
echo $abc
```

ソース中に記述されたコメントである「# assign....」の行は無視されます。

○実習: コメントがあるスクリプトの実行

9.3 シェルスクリプトの作成

コメントがあるシェルスクリプトを記述して、動作を確認してみましょう。

シェルスクリプトファイルの作成

```
# vi comment.sh
```

シェルスクリプトを記述

```
#!/bin/bash

abc=xyz
#abc=123
echo $abc
```

シェルスクリプトを実行

```
# chmod u+x comment.sh
# ./comment.sh
xyz
```

9.3.8 引用符

コマンドを実行する場合、変数を引用符で囲みます。利用できる変数は'(シングルクオート)、"(ダブルクオート)、'(バッククオート)があります。シングルクオートで囲った場合、中の文字列は「文字列」として認識されるため、引用符内に変数があったとしても文字として出力されます。ダブルクオートで囲った場合、引用符内に変数があった場合、変数が展開されて出力されます。バッククオートで囲った場合、引用符内にコマンドがあるとそのコマンドを実行した結果が展開されます。

○実習：引用符の動作確認

引用符の違いについて実際に確認してみましょう。

```
$ abc=xyz
$ echo 'value abc is $abc'
value abc is $abc
($abc が文字として認識され展開された)
$ echo "value abc is $abc"
value abc is xyz
($abc が変数として認識され、内容である xyz が展開された)
$ abc=`date`;
$ echo $abc
Thu Mar 20 06:08:14 JST 2012
($abc に date コマンドの実行結果が入っている)
```

9.3.9 引数

シェルスクリプトは、実行時にオプションを引数として参照することができます。引数は\$1, \$2…など\$の後に引数の番号を指定することで参照できます。

○実習: 引数出力の確認

次のように args.sh を作成して、引数呼び出しの動作を確認してみましょう。

```
$ cat args.sh
#!/bin/bash

echo '$1:' $1;
echo '$2:' $2;
echo '$3:' $3;
echo '$0:' $0;
echo '$#:!' $#;
```

```
$ ./args.sh aaa bbb ccc
$1: aaa
$2: bbb
$3: ccc
$0: ./args.sh
$#: 3
($1-$3 は引数、$0 は実行コマンド名、$#は、引数の数を示す)
```

9.3.10 shift 文

shift コマンドは、引数の順序をずらします。shift を実行すると、\$2 が\$1 に、\$3 が\$2 に…になります。

○実習: shift コマンドの確認

実際にスクリプトを作成して shift コマンドの動作を確認してみましょう。

```
$ cat argsshift.sh
#!/bin/bash

echo '$1:' $1;
echo '$2:' $2;
echo '$3:' $3;
shift
echo '$1:' $1;
```

9.3 シェルスクリプトの作成

```
echo '$2:' $2;
```

作成したスクリプトに対して、変数を渡してみます。

```
$ ./argshift.sh aaa bbb ccc
$1: aaa
$2: bbb
$3: ccc
$1: bbb
($1 が bbb に変わった)
$2: ccc
($2 が ccc に変わった)
```

9.3.11 エスケープシーケンス

プログラミングでは、特別な文字があります。例えば、echo コマンドを利用する場合を考えてみましょう。echo コマンドで、"(ダブルクオート)" を出力する場合、次のように入力すれば良いと考えるかもしれません。

```
echo """" (ダブルクオートをダブルクオートではさんだ)
```

しかし、これは、ダブルクオートが3つあるという、対応関係がとれていない状態と判断されるため、このコマンドを実行するとエラーになります。このコマンドのエラーを回避するには「2つのダブルクオートは、区切り文字のダブルクオートではなく、文字として出力する」ということを伝える必要があります。そのときに用いる記号をエスケープシーケンスといい、シェルスクリプトでは\$(バックスラッシュ)を使います。

```
$ echo "¥"""
"
(ダブルクオートが標準出力される)
```

この例で出力されたのは、2個目のダブルクオートです。

また、エスケープシーケンスは改行コードにも有効です。\$(バックスラッシュ)を行末に付与することで、文字列の途中で折り返すことができます。適切に改行を入れることでコマンドの視認性を向上できます。

実行例

```
$ echo "I am Cat.My name is nothing."
I am Cat.My name is nothing.
$ echo "I am Cat.¥
> My name is nothing."
I am Cat.My name is nothing.
(途中でバックスラッシュを入れても改行は無視され同じ結果となる)
```

バックスラッシュによる改行は、シェルスクリプト内でも使うことができます。バックスラッシュは本来\ですが、多くの日本語環境では¥と表示されるので、本教科書でも¥で表示しています。環境によっては\と表示されます。

実行例

```
$ vi escape.sh
#!/bin/bash

echo "I am Cat.¥
My name is nothing."

$ ./escape.sh
I am Cat.My name is nothing.
```

9.3.12 source コマンド

source コマンドは、bashなどのシェルの内部コマンドで、指定されたファイルを読み込んでシェル環境を設定します。ファイル内容はシェルコマンドと解釈して実行します。

一般的な用途としては、シェルの環境設定ファイルである".bashrc"や".bash_profile"などを設定変更後、ログインしなおさずに設定を現在のシェル上で有効にする場合に使われます。

実行例のように setsh というシェルスクリプトを作成、\$abc という変数に xyz を定義して、source コマンドで読み込んでみましょう。

○実行例

1. シェルスクリプト「set.sh」を用意

```
$ cat set.sh
(set.sh の内容確認)

#!/bin/sh
abc=xyz
echo $abc
```

2. \$abc を echo コマンドで出力

9.4 条件分岐

```
$ echo $abc  
($abc には何も格納されてないため、何も表示されない)
```

3. set.sh スクリプトを実行

```
$ ./set.sh  
xyz  
(スクリプト内でのみ$abc に"xyz"が格納されている)
```

set.sh には echo 文が記述されているため、その値が出力される。

4. \$abc を echo コマンドで出力

```
$ echo $abc  
($abc には何も格納されてないため、何も表示されない)
```

5. source コマンドで set.sh を読み込む

```
$ source set.sh  
xyz  
($abc に"xyz"が格納されたため、echo コマンドにより出力される)
```

6. source コマンドで読み込んだことにより、\$abc に値が格納された

```
$ echo $abc  
xyz
```

9.4 条件分岐

プログラミングでは、条件にそって挙動を切り替える「条件分岐」を多用します。条件分岐はすべてのプログラミング言語に存在する機能です。もちろんシェルスクリプトでも利用することができます。

9.4.1 if 文

比較などによる条件分岐を行なう場合は if 文を利用します。文法は次の通りです。

書式

```
if 条件式1 then ... elif 条件式2 ... else ... fi
```

elif... の部分と *else...* の部分は省略可能です。*elif* は、別の条件 (*条件式2*) で判断したい場合に利用します。*else* は条件が全てあてはまらなかつた場合、実行されます。*if* 文は *fi* で終了します。

条件式 *if* 文で利用される条件式には、次のような文法があります。

表 9.1 文字列比較を行なう演算子

演算子	比較内容
<i>a == b</i>	<i>a</i> と <i>b</i> が等しければ真
<i>a != b</i>	<i>a</i> と <i>b</i> が等しくなければ真

表 9.2 数値比較を行なう演算子

演算子	比較内容
<i>a -eq b</i>	<i>a</i> と <i>b</i> が等しい (equal) ければ真
<i>a -ne b</i>	<i>a</i> と <i>b</i> が等しくなければ (not equal) 真
<i>a -ge b</i>	<i>a</i> が <i>b</i> 以上 (grater equal) であれば真
<i>a -le b</i>	<i>a</i> が <i>b</i> 以下 (less equal) であれば真
<i>a -gt b</i>	<i>a</i> が <i>b</i> より大きい (grater than) 値であれば真
<i>a -lt b</i>	<i>a</i> が <i>b</i> 未満 (less than) であれば真

9.4.2 ファイル属性の確認

ファイル属性の確認は次のように行います。

書式

```
if test -d パス ; then.....
```

-d の部分がファイル属性確認の演算子にあたります。*-d* はディレクトリであるかの判定を行うので、この *if* 文全体で、「パスがディレクトリであれば真の値を返す」という条件式になります。ファイルの属性確認演算子は次の以下のようなものが利用できます。

なお、*test* コマンドは [] を使って記述することもできます。

書式

```
if [ 条件節 ]; then ...
if test 条件節 ; then ...
```

9.4 条件分岐

表 9.3 ファイル属性の確認を行なう演算子

演算子	内容
-f ファイル名	通常ファイルなら真
-d ファイル名	ディレクトリなら真
-e ファイル名	ファイルが存在すれば真
-L ファイル名	シンボリックリンクなら真
-r ファイル名	読み取り可能ファイルなら真
-w ファイル名	書き込み可能ファイルなら真
-x ファイル名	ファイルが存在して、実行権限があれば真
-s ファイル名	サイズが 0 より大きければ真

ファイルに関する比較演算子は属性以外にもあります。詳細については man コマンドで test コマンドの項目を参照してください。

```
man test
```

9.4.3 複数の条件を重ねる

条件分岐の場合、複数の条件を重ねることができます。条件 A と条件 B が同時に成立している必要があるときは、「条件 A”かつ” 条件 B が成立」ということで、論理積 (=AND) が用いられます。同じく「条件 A”もしくは” 条件 B が成立」の場合は、論理和 (=OR) が用いられます。シェルスクリプトにおいて、論理積・論理和の書き方は 2 通り存在します。

論理積

論理積は-a を用いる場合と、&&を用いる場合があります。それぞれの利用形式は次の通りです。

書式

```
[条件 A -a 条件 B -a 条件 C ] ....  
[条件 A] && [条件 B] && [条件 C] ...
```

-a と&&は、それぞれ [] の内側か外側かで、使われ方が変わることに気をつけてください。

論理和

論理和を表す記号は、やはり 2 通り存在します。-o と||です。利用形式は次の通りです。

書式

```
[条件 A -o 条件 B -o 条件 C ] ....
```

```
[条件 A] || [条件 B] || [条件 C] ...
```

9.4.4 一対多の条件分岐

一対多の分岐を行なうことを考えてみましょう。if 文を用いた場合、一対多の分岐は以下のようになります。シェルスクリプトでは、case 文が用意されており、一対多の分岐が記述できるようになりました。

```
if 条件式 then
:
elif 条件式 then
:
elif 条件式 then
:
:
fi
```

書式

```
case 変数 in
    値 A)
        処理 1;;
    値 B)
        処理 2;;
esac
```

変数の値が値 A のとき、処理 1 が実行されます。最後は case 文の逆からの記述である esac で終わっていることが重要です。値はパイプ記号 (|) で区切って複数指定することができます。

9.4 条件分岐

実行例

```
$ cat case.sh  
(次のように case.sh を作成して実行します)  
  
#!/bin/bash  
  
case $1 in  
    a|A)  
        echo "引数に a または A が入力されました";;  
    b|B)  
        echo "引数に b または B が入力されました";;  
esac
```

これを実行すると、以下のような実行結果になります。

```
$ ./case.sh a  
引数に a または A が入力されました  
$ ./case.sh B  
引数に b または B が入力されました
```

また、どの値にもマッチしなかった場合の処理を記述するには、値にアスタリスク (*) を用います。

実行例

```
$ cat defaultcase.sh          次のように defaultcase.sh を作成し、実行します。  
#!/bin/bash  
  
case $1 in  
    1)  
        echo "引数に 1 が入力されました";;  
    2)  
        echo "引数に 2 が入力されました";;  
    *)  
        echo "1,2 以外が入力されました";;  
esac
```

これを実行すると、以下のような実行結果になります。

```
$ ./defaultcase.sh 1  
引数に 1 が入力されました  
$ ./defaultcase.sh 2  
引数に 2 が入力されました  
$ ./defaultcase.sh 0  
1,2 以外が入力されました  
(どの条件にもマッチしない場合、値*の処理が実行されます)
```

9.5 繰り返し

プログラミングにおいて、条件分岐と同じくらい重要な機構が、この繰り返しです。同じ処理を繰り返し行い、ある条件が成立したときに終了する、という形式が用いられています。シェルスクリプトで用いられている繰り返しは、以下の3通りです。しっかりと学習しましょう。

9.5.1 for文

for文は値を列挙し、それを対象に処理を繰り返します。

書式

```
for 変数 in 値のリスト
do
    処理
done
```

値のリストとは、文字を羅列したものもあれば、実行結果を使うこともできます。

実行例

```
$ for i in a b c d
> do
> echo $i
> done
a
b
c
d
```

これは、値のリストとして'a b c d'が渡され、それを `i=a` を行い `echo $i`、次に `i=b` を行い `echo $i` を・・・と繰り返した結果です。実行結果としてリストを使うと以下のように `i` に `ls` の実行結果を代入し、ループが実行されます。

```
for i in `ls`
```

9.5.2 while/until文

while文は、条件が成立している間ループを繰り返す（条件が成立しなくなったら終了）という処理で利用されます。until文はそれの反対で、条件が成立していない間ループを繰り返す（条件が成立したら処理を終了）、という用途に用いられます。

9.5 繰り返し

書式

```
while 条件式
do
    処理
done
until 条件式
do
    処理
done
```

C 言語の for のようなループ文をシェルスクリプトで実現するには、expr コマンドを用いてループカウンタ用の変数をインクリメント（またはデクリメント）しながら、while/until 文で処理を行ないます。

次のような内容の loop.sh を作成し、実行してみましょう。

実行例

```
$ cat loop.sh

#!/bin/bash

count=1
while [ $count -le 10 ]
do
    echo "この処理は$count 回実行されました"
    count=`expr $count + 1`
done
```

count=1 で、カウンタ用の変数 count に初期値 1 を設定します。while [\$count -le 10] で、シェル変数 count が 10 以下の間、処理を繰り返します。

これを実行すると、以下のような実行結果になります。

```
$ ./loop.sh
この処理は 1 回実行されました
この処理は 2 回実行されました
この処理は 3 回実行されました
この処理は 4 回実行されました
この処理は 5 回実行されました
この処理は 6 回実行されました
この処理は 7 回実行されました
この処理は 8 回実行されました
この処理は 9 回実行されました
この処理は 10 回実行されました
```

9.5.3 select 文

select 文は、ユーザに対し数値による入力を促します。

書式

```
select 変数 in リスト
do
    処理
done
```

実際に動作例を示してみます。

```
select name in "apple" "banana" "orange"
do
    echo "You selected $name";
done
```

これを実行すると、以下のような実行結果になります。

```
1) apple
2) banana
3) orange
$? 1
You selected apple
(「Ctrl」 Cで中止)
```

1-3 を入力すると、do～done 内部が実行されます。

9.5.4 繰り返しの制御

break や continue を用いることで、繰り返しを制御することができます。break は繰り返しを終了して、continue は繰り返しの先頭に戻る役割があります。

実行例

```
$ cat ./sample.sh
while true
do
    echo "Continue? (y/n)"
    read input
    case $input in
        n) break
            ;;
        y) continue
            ;;
        *) echo "Please input y or n."
            ;;
    esac
done
```

これを実行すると、以下のような実行結果になります。

```
$ ./sample.sh
Continue? (y/n)
y
(y を入力)
Continue? (y/n)
(繰り返しの先頭に戻る)
a
(y,n 以外を入力)
Please input y or n.
Continue? (y/n)
n
(n を入力)
$
(繰り返しを終了する)
```

9.6 サブルーチン

プログラミングをする上で、一連の処理をまとめて、再利用できるようにしたものを、サブルーチンといいます。サブルーチンは言語体系によりさまざまな呼ばれ方をしていて、シェルスクリプトでは関数と呼ばれています。

9.6.1 関数

関数は、引数とよばれるデータを与え、処理をして結果を返すという機能の集まりです。書式は次のようになっています。

書式

```
function 関数名
{
    処理
}
```

```
関数名()
{
    処理
}
```

両者とも働きは同じです。引数は、実行時に関数名に続けて記述します。引数は、関数の内部で \$1, \$2 で参照することができます。

9.6.2 return 文

シェルスクリプトの関数で、結果を返すときは return 文を実行します。

書式

```
return 変数名
```

return 文を実行すると、関数内での処理はそこで終了し、変数が関数の呼び出し元に返されます。

9.7 実際のシェルスクリプト

それでは、実際に稼働しているシェルスクリプトを見てみましょう。（今回は CentOS 6.3 の起動スクリプトを例にあげています。他のディストリビュージョンおよび CentOS のバージョンによっては若干内容が異なる場合があります）

9.7.1 起動スクリプト

今回は/etc/rc を参考にします。/etc/rc のシェルスクリプトは、起動シーケンスの際に実行するスクリプトで、起動時に必要な処理が記述されています。

28 行目

```
. /etc/init.d/functions
```

/etc/init.d/functions というファイルを読み込みます。/etc/init.d/functions は、/etc/の中に入っているさまざまなシェルスクリプトが、共通して利用できる便利な関数が入っています。それを有効にするために、28 行目が実行されます。

50 行目

```
[ -d /etc/rc$runlevel.d ] || exit 0
```

\$runlevel には、ランレベルの数値が入っています。ランレベルが 3 であったとしたとき、-d により /etc/rc3.d の存在をチェックしています。|| は論理和の||です。||の左側で /etc/rc3.d のディレクトリの存在をチェックし、もし存在していたら、そのまま処理が次にわたります。言い換えると、||の左側で真が返った場合、||の右には処理が移りません。exit 0 は、処理をここで終了させることを意味しています。従って、50 行目では「/etc/rc3.d の存在を確認。もし無かつたらここで終了」という処理をしています。exit の後に続く数値は、シェルスクリプトの終了コードを表します。通常はスクリプトが正常に終了する場合に 0 を、それ以外の場合にエラーコードを示す値を設定します。省略した場合には 0 が設定されます。設定された値は特殊変数 \$? に格納され、\$? にて参照することができます。

できます。

60 行目

```
for i in /etc/rc$runlevel.d/K* ; do
```

ここでもランレベル (\$runlevel) が 3 だとして説明します。/etc/rc3.d には、K で始まるストップさせるサービスの一覧と、S で始まるスタートさせるサービスの一覧が存在します。60 行目を実行すると、i に K で始まるシェルスクリプト一覧が入ります。その他さまざまな処理がありますが、69 行目で処理を停止させる \$i stop の処理を実行します。K で始まるシェルスクリプトは、実体は /etc/init.d/ にスクリプトが入っています。K の後に始まる 2 桁の数値は、停止させる順番になっています。これは、60 行目でマッチさせるための順番になっています。

9.7.2 関数のシェルスクリプト

/etc/init.d/functions というファイルがあります。/etc/rc の 28 行目でも説明したとおり、シェルスクリプトの便利な関数を集めたシェルスクリプトです。その中で記述されている関数を 1 つ説明します。

578 行目

```
is_ignored_file() {
    case "$1" in
        *~ | *.bak | *.orig | *.rpmnew | *.rpmod | *.rpmsave)
            return 0
        ;;
    esac
    return 1
}
```

is_ignored_file() という関数は、/etc/rc の中に無視したいファイル名に該当するかどうかのチェックを行っており、引数の 1 つめ (\$1) にファイル名が代入されます。case 文で引数に格納されたファイル名が *~, *.bak, *.orig, *.rpmnew, *.rpmod, *.rpmsave (*は 0 文字以上の任意の文字列を意味します) のいずれかの条件に該当していれば 0 (偽) を返し、該当していなければ 1 (正) を返します。

無視したいファイル名が増えた場合、例えば .abc で終わる場合が出たとします。そのときは *.abc のエントリをここで加えると、/etc/rc のみならず、is_ignored_file() を呼んでいる関数全部で同等の機能が反映されます。関数の利用は、呼ばれているプログラムの機能を一括で更新できる、という利点があることを理解してください。

この関数は /etc/rc 以外でも利用できます。

9.8 デバッグ

作成したプログラムが思ったように動作しない場合、どこに問題があるのか調べる必要があります。これをデバッグといいます。プログラムの所々に、変数を表示させるコードや、プログラムの走行ルートを表示させるコードを埋め込む必要があります。非常に労力がかかります。よって多くのプログラム言語には、デバッグを支援するツールが用意されています。シェルスクリプトにも、スクリプトをデバッグモードで動かすためのコマンドが用意されています。

9.8.1 sh コマンド

sh 自身はシェルを起動するコマンドですが、-x オプションを付けて引数にシェルスクリプトを指定すると、コマンドや変数の中身を表示しながらスクリプトを実行します。繰り返しの項で使用した sample.sh を sh コマンドにて実行してみましょう。

実行例

```
$ sh -x ./sample.sh
+ true
+ echo 'Continue? (y/n)'
Continue? (y/n)
+ read input
y
(y を入力)
+ case $input in
+ continue
+ true
+ echo 'Continue? (y/n)'
Continue? (y/n)
+ read input
n
(n を入力)
+ case $input in
+ break
```

9.9 9章 章末テスト

(1) 実行結果が解になるように、変数"LPi"に入力した値を出力しなさい。

```
$ LPi = linux
$ echo (      )
linux
```

(2) 以下のスクリプトの空欄を埋め、カレントディレクトリにファイル lpi.txt が存在しているか確認するスクリプトを完成させなさい。

```
#!/bin/bash
if [ -f lpi.txt ]; then
    echo file exists.
(
)
echo file does not exists.
(
)
```

(3) 以下のスクリプトの空欄を埋め、カレントディレクトリ内のすべてのファイル（サブディレクトリを除く）の種類を表示するスクリプトを完成させなさい。

```
#!/bin/bash
for i in `ls`
do
    if [ ! -d $i ]; then
        file $i
    fi
(
)
```

(4) 次の動作をするシェルスクリプトを書きなさい。

1. 実行すると、age:として年齢の入力を促される。
2. 20 以上の値を入れると、'you can drink.'と出力される。
3. 20 未満の値を入れると、'you cannot drink.'と出力される。

(5) bash で、シェルスクリプトの中身を表示しながら実行（デバッグ）するにはどのようにしたらよいか答えなさい。

第10章

ネットワークの設定と管理

Linux がネットワークにつながれたとき、必要とされる基礎知識と確認コマンドと設定を見ていきます。確認コマンドとして ping コマンド、traceroute コマンド、ifconfig コマンド、nslookup コマンド、/etc/rc.d/init.d/network スクリプトを説明します。

この章の内容

10.1 TCP/IP とは

10.1.1 ローカルエリアネットワークとワイドエリアネットワーク

10.1.2 IP とは

10.1.3 TCP と UDP とは

10.1.4 IP アドレス

10.1.5 IP アドレスのクラス

10.1.6 プライベート IP アドレス

10.1.7 サブネットマスク

10.1.8 経路の確認

10.1.9 ネットワークインターフェース

10.1.10 IP アドレスを確認

10.1.11 IP アドレスの設定ファイル

10.1.12 インターフェースの設定

10.1.13 ルーティングの確認

10.1.14 ルーティングの変更

10.1.15 DNS を使う設定

10.1.16 名前 (FQDN)

10.1.17 ポート番号

10.1.18 サービスの確認

10.1.19 ネットワークセキュリティの設定

10.1.20 ファイアウォールの設定

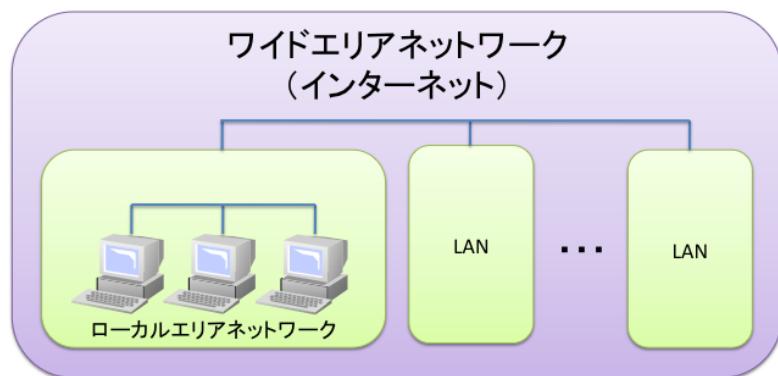
10.2 章末テスト

10.1 TCP/IP とは

コンピュータ間をケーブルや無線機能で接続したシステムをネットワークと呼びます。Linux を OS に利用したコンピュータのほとんどはネットワークに接続して利用します。物理的なネットワークはコンピュータ間をツイストペアケーブルでつなぎます。最近では有線のネットワークに加えて無線機能を使って接続したり、有線と無線の混在したネットワークもあります。

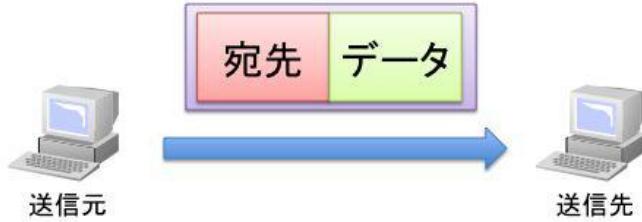
10.1.1 ローカルエリアネットワークとワイドエリアネットワーク

学校や自宅などの閉じられた複数のコンピュータを相互接続したネットワークをネットワークをローカルエリアネットワーク (LAN) と呼びます。離れた場所にある LAN と LAN を結んだネットワークのことをワイドエリアネットワーク (WAN) と呼びます。LAN はコンピュータ同士を LAN ケーブルで結び、WAN は LAN 同士を通信回線で結びます。



10.1.2 IP とは

ネットワークでつながったコンピュータ同士の間では、決まった手順（プロトコル）に従ってデータを送受信しています。今日一般的に使われているイーサネット規格のネットワークはプロトコルとして TCP/IP を利用しています。TCP/IP の IP(Internet Protocol : インターネット プロトコル) は手順の基本です。IP は送り先とデータからなるパケットを送るだけの簡単な仕組みです。ping コマンドでデータが送信され、データを受けた機器 (サーバやルータなど) から確認メッセージかが返ってくるかを確認できます。



書式

```
ping ターゲット
ターゲット(ホスト名や IP アドレス)にデータを送り、返答が戻るまでの時間を表示します。
-c オプションを付けて、ping を発行する回数を指定することもできます。
```

○実習: ローカルなマシンの IP アドレスの確認

まずは、LAN 上にあるホストに ping コマンドを実行してみましょう。

```
$ ping 192.168.1.1
(192.168.1.1 へ ping を送信)
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=30 time=1.01 ms
(1.01m 秒で返答があった)
64 bytes from 192.168.1.1: icmp_seq=2 ttl=30 time=0.914 ms
(0.914m 秒で返答があった)

(「Ctrl」 C で ping コマンドを中止)

--- 192.168.1.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 0.914/0.966/1.018/0.052 ms
```

○実習: インターネットにつながるマシンで確認

ping コマンドに対して lpi のサーバからの返答があるか(返答時間も)を確認してください。ping を 3 回実施するため、-c オプションを付けてコマンドを実行します。

```
$ ping lpi.jp -c 3
PING lpi.jp (203.174.74.34) 56(84) bytes of data.
64 bytes from sv1.lpi.jp (203.174.74.34): icmp_seq=1 ttl=52 time=3.14 ms
64 bytes from sv1.lpi.jp (203.174.74.34): icmp_seq=2 ttl=52 time=6.87 ms
64 bytes from sv1.lpi.jp (203.174.74.34): icmp_seq=3 ttl=52 time=5.67 ms

--- lpi.jp ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2009ms
rtt min/avg/max/mdev = 3.142/5.229/6.873/1.556 ms
```

ping コマンドは「Ctrl」C を押すと止まります。コンピュータによってはセキュリティ対策のため、ping コマンドに応答しない場合があります。

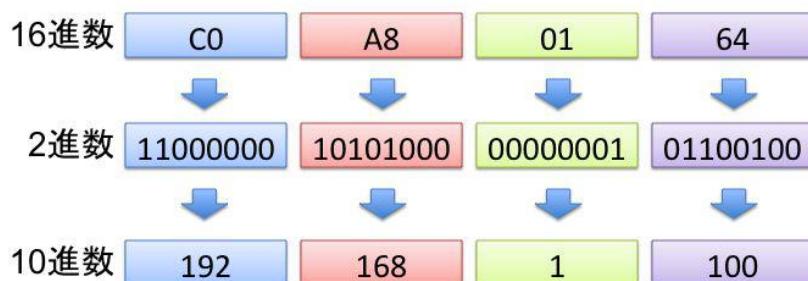
10.1.3 TCP と UDP とは

TCP(Transmission Control Protocol) は IP(Internet Protocol) の仕組みを使ってデータを送る手順です。IP は読んで字のごとく、インターネットにおいて情報の伝達を行なう手順(プロトコル)です。TCP では、データのまとまりを 1 つ受け取ると、すぐにデータの破損などのエラーを確認し、間違っている場合はデータの再送を依頼します。

UDP(User Datagram Protocol) はエラー確認やデータの再送制御は行いませんが(必要であれば上位のアプリケーションでそのような処理を実装する必要があります)、その分データを高速に送ることができるので、小さなデータや映像のような多少欠損しても問題なく、信頼性よりも速度を求められる通信に適しています。

10.1.4 IP アドレス

TCP/IP でデータを送受信するには発送元と発送先の場所を表すアドレスが必要です。インターネットでは IPv4(Internet Protocol version 4) と IPv6 という 2 つのプロトコルが使われています。IPv4 では 4 バイト(16 進数で 8 桁)で表せるアドレスで、4 バイトのアドレスは 1 バイト(16 進数で 2 桁)ごとで 10 進数に変換してでつないで表記します。プライベートな IP アドレスを除いた IP アドレスをグローバルな IP アドレスと呼びます。グローバルな IP アドレスは NIC(日本では JPNIC) に管理されており、許可なく利用(設定)できません。



10.1.5 IP アドレスのクラス

IPv4 で使われる IP アドレスはネットワークアドレスとホストアドレスをつなげた形となります。ネットワークアドレスは次の表のサブネットマスクのビットが立っている (255 の様な) 部分で、ホストアドレスはサブネットマスクのビットが立っていない (0) の部分です (サブネットマスクに関しては後述します)。IP アドレスは学校や会社などの組織に対して割り当てられるので、割り当てられた組織で使われる範囲は連続したアドレスです。ネットワークアドレスの長さにより、A から C のクラスがあり、クラス D やクラス E などの特殊なクラスも予約されています (現状では A から C のクラスが実用されます)。

ホストアドレスにおいて、ホスト部の全てのビットが 0 のアドレスはそのネットワーク自身のアドレスを示し、ホスト部の全てのビットが 1 のアドレスは、そのネットワーク内の全てのホストに届くプロードキャストアドレスという特殊なアドレスを示します。これらのアドレスは実際にコンピュータに割り当てるとはできません。

表 10.1 IP アドレスのクラス

アドレスの範囲	サブネットマスク	クラス	IP アドレスの数
0.0.0.0～127.255.255.255	255.0.0.0	クラス A	16,777,216 個
128.0.0.0～191.255.255.255	255.255.0.0	クラス B	65,536 個
192.0.0.0～223.255.255.255	255.255.255.0	クラス C	256 個
224.0.0.0～239.255.255.255		クラス D	
240.0.0.0～255.255.255.255		クラス E	

10.1.6 プライベート IP アドレス

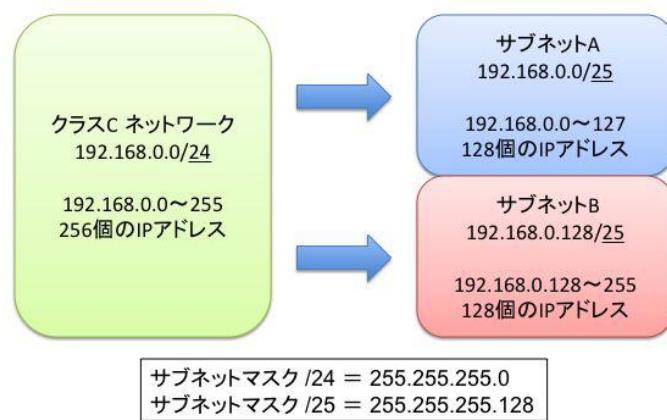
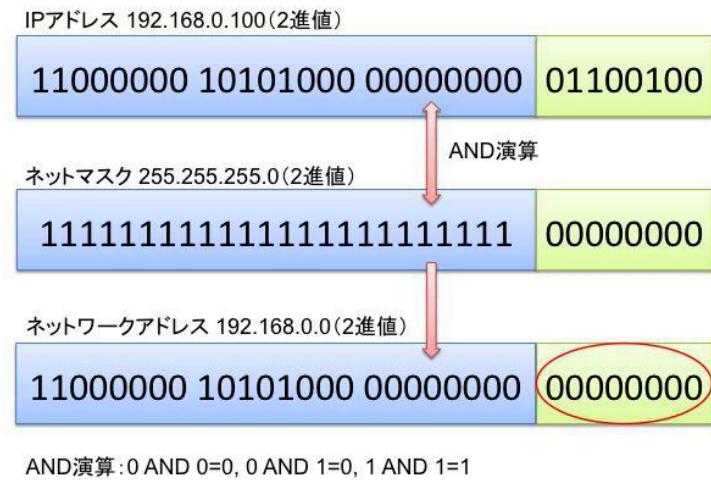
ローカルなネットワークで自由に利用可能な IP アドレスが用意されており、プライベート IP アドレスと呼びます。プライベート IP アドレスは次のようにになります。

表 10.2 プライベート IP アドレスとクラス

アドレスの範囲	サブネットマスク	クラス
10.0.0.0～10.255.255.255	255.0.0.0	クラス A
172.16.0.0～172.31.255.255	255.255.0.0	クラス B
192.168.0.0～192.168.255.255	255.255.255.0	クラス C

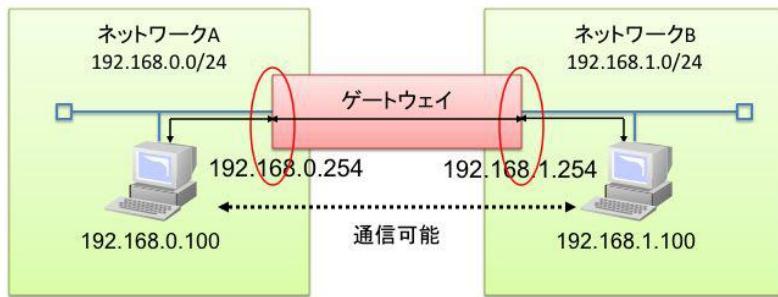
10.1.7 サブネットマスク

IP アドレスのうち、ネットワークアドレスとホストアドレスを識別するための数値をサブネットマスクといいます。サブネットマスクは通信先ホストが同一ネットワークにいるかいないかの判断に使われます。ネットワーク (IP アドレスの集まり) はサブネットマスクを変えることで、複数のネットワークに分けて、効率的に IP アドレスを利用できます。



10.1.8 経路の確認

LAN と LAN をつなげる場合、もしくは LAN と WAN をつなげる場合は、ゲートウェイ（ルーター）が間に接続されています。ゲートウェイは次の図の様にネットワークのデータを接続された隣のネットワークへ転送します。



インターネットは多くのネットワークがつながって構成されていて、いくつものネットワークをデータが中継されるので、いくつものゲートウェイを通ってデータが送受信されます。自分が利用しているコンピュータから通信先のコンピュータまでに仲介する複数のゲートウェイを調べるには traceroute コマンドがあります。

書式

```
traceroute ターゲット
```

ターゲットや途中のゲートウェイへデータを送り、ゲートウェイから返答が戻ってくる時間を表示します。

○実習: traceroute コマンドでゲートウェイを確認

traceroute コマンドで、lpi.jp サーバまでの間に複数のゲートウェイから返答があるかを確認してください。

```
# traceroute lpi.jp
(1pi.jp をトレースルート)
traceroute to lpi.jp (203.174.74.34), 30 hops max, 60 byte packets
(最大 30 台までの経由ルータを表示)
1 172.16.x.x (172.16.x.x)  1.924 ms  1.772 ms  2.136 ms
2 172.17.x.x (172.17.x.x)  3.447 ms  4.645 ms  5.917 ms
(略)
14 sv1.lpi.jp (203.174.74.34)  31.723 ms  30.773 ms  30.823 ms
```

172.16.x.x まで 2.136 m 秒、172.17.x.x まで 5.917 m 秒、sv1.lpi.jp まで 30.823 m 秒かかっています。ゲートウェイによってはセキュリティ対策のため、traceroute コマンドに応答しない場合があります。

10.1.9 ネットワークインターフェース

ネットワークへアクセスするためにネットワークインターフェースが必要です。物理的なネットワークインターフェースとしてネットワークインターフェースカード (NIC) があります。現在は、多くの場合、ネットワークインターフェースがコントロールチップの中に統合されています。プログラムが内部的に使う仮想のネットワークインターフェースとしてループバックインターフェースがあります。

10.1.10 IP アドレスを確認

ifconfig コマンドで現在の設定を確認できます。

書式

```
ifconfig [インターフェース]  
ネットワークインターフェースに設定された IP アドレスとサブネットマスクを表示します。
```

インターフェース (lo や eth0 など) を省略すると、全インターフェース情報を表示します。

○実習: ネットワークインターフェースの IP アドレスを確認

ifconfig コマンドを使ってホストの IP アドレスを確認してみましょう。

```
$ ifconfig  
(ネットワークインターフェースの情報を表示)  
  
eth0      Link encap:Ethernet  HWaddr 00:0C:29:EB:5D:C6  
          inet addr:192.168.1.60  Bcast:192.168.1.255  Mask:255.255.255.0  
          inet6 addr: fe80::20c:29ff:feeb:5dc6/64 Scope:Link  
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1  
            RX packets:11432 errors:0 dropped:0 overruns:0 frame:0  
            TX packets:8705 errors:0 dropped:0 overruns:0 carrier:0  
            collisions:0 txqueuelen:1000  
            RX bytes:2169853 (2.0 MiB)  TX bytes:1097164 (1.0 MiB)  
            Interrupt:185 Base address:0x1400  
  
lo        Link encap:Local Loopback  
          inet addr:127.0.0.1  Mask:255.0.0.0  
          inet6 addr: ::1/128 Scope:Host  
            UP LOOPBACK RUNNING  MTU:16436  Metric:1  
            RX packets:2385 errors:0 dropped:0 overruns:0 frame:0  
            TX packets:2385 errors:0 dropped:0 overruns:0 carrier:0  
            collisions:0 txqueuelen:0  
            RX bytes:4014756 (3.8 MiB)  TX bytes:4014756 (3.8 MiB)
```

ifconfig コマンドで表示された lo がループバックインターフェース (機器が正常に稼動しているか確認するためにデータを送って試すループバックテストなどに利用) で、eth0 が物理的に存在する

ネットワークインターフェースです。

マシンに物理的に存在するネットワークインターフェイスが複数ある場合、CentOS の場合は一番最初に認識したネットワークインターフェイスを eth0 として、eth1、eth2... とインターフェイスが増えていきます。

10.1.11 IP アドレスの設定ファイル

RedHat Linux 系のディストリビューションは IP アドレスの設定を 2 つの設定ファイルに記述しておき、ブート時に 2 つの設定ファイルを使って設定されます。/etc/sysconfig/network-scripts/ifcfg-XXX 設定ファイルは XXX インターフェースの IP アドレスとサブネットマスクなどを記述します。XXX はループバックインターフェースの場合は lo、ネットワークインターフェースカードの場合は eth0, eth1, … と順番に振られます。

/etc/sysconfig/network-scripts/ifcfg-XXX 設定ファイルの主な内容は次の様になります。

表 10.3 ifcfg 設定ファイルの項目

項目	内容
ONBOOT	起動時にアドレスを自動設定する (yes) か設定しない (no)
DEVICE	ネットワークインターフェイス
HWADDR	MAC アドレス

BOOTPROTO 以降の設定を利用 (static) か自動取得 (dhcp)

表 10.4 固定アドレス (static) を設定する場合の設定項目

項目	内容
IPADDR	IP アドレス
NETMASK	サブネットマスク
NETWORK	ネットワーク

起動時にアドレスを自動設定するには DHCP 機能が提供された環境が必要です。/etc/sysconfig/network 設定ファイルはデータが他のネットワークへ転送されるゲートウェイのアドレスとホスト名を記述します。/etc/sysconfig/network 設定ファイルの主な内容は次の様になります。

表 10.5 ネットワークの項目

項目	内容
NETWORKING	ネットワークを有効にする (yes) か無効にする (no)
HOSTNAME	ホスト名
GATEWAY	ゲートウェイアドレス

10.1 TCP/IP とは

2つの設定ファイルの内容はディストリビューションによって少しずつ違います。

○実習: ネットワークインターフェースの確認

cat コマンドでファイルを表示してみます。

```
$ cat /etc/sysconfig/network-scripts/ifcfg-lo
(設定ファイルを表示)
DEVICE=lo
IPADDR=127.0.0.1
(IP アドレス)
NETMASK=255.0.0.0
(サブネットマスク)
NETWORK=127.0.0.0
(ネットワーク)
# If you're having problems with gated making 127.0.0.0/8 a martian,
# you can change this to something else (255.255.255.255, for example)
BROADCAST=127.255.255.255
ONBOOT=yes
(ブート時に有効にする)
NAME=loopback
(インターフェース名)

$ cat /etc/sysconfig/network-scripts/ifcfg-eth0
(設定ファイルを表示)
# Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE]
DEVICE=eth0
BOOTPROTO=static
BROADCAST=192.168.1.255
HWADDR=00:0C:29:FB:5F:23
IPADDR=192.168.1.60
NETMASK=255.255.255.0
NETWORK=192.168.1.0
ONBOOT=yes

$ cat /etc/sysconfig/network
(設定ファイルを表示)
NETWORKING=yes
(ネットワークを有効)
NETWORKING_IPV6=no
HOSTNAME=localhost.localdomain
(ホスト名)
GATEWAY=192.168.1.1
(デフォルトゲートウェイ)
```

MAC アドレスはハードウェアに設定されているユニークな番号であり、正しい MAC アドレスが設定されていない場合通信することができなくなるため、自動的に認識された番号を書き換えないようにしましょう。ブロードキャストは 1 対多で通信をする場合に使われるアドレスで、ネットワークのアドレスの一番最後にします。

10.1.12 インターフェースの設定

ネットワークのインターフェースは設定ファイルを書き換えて、Linux がブートされるときに実行される /etc/init.d/network スクリプトに restart オプションを付けて変更できます。

○実習：ネットワークインターフェースの再設定

ifconfig コマンドで設定を確認後、IP アドレスを変更して network サービスを再起動してみましょう。network サービスを再起動したあとは、ifconfig コマンドで IP アドレスの変更が適用されているか確認してください。

```
# ifconfig eth0
(eth0 の設定を表示)

eth0      Link encap:Ethernet HWaddr 00:0C:29:FB:5F:23
          inet addr:192.168.1.60 Bcast:192.168.1.255 Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:febf:5f23/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:73136 errors:0 dropped:0 overruns:0 frame:0
            TX packets:46269 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:15664826 (14.9 MiB) TX bytes:7723381 (7.3 MiB)
            Interrupt:185 Base address:0x1400

# vi /etc/sysconfig/network-scripts/ifcfg-eth0
(IP アドレスを変更)

# /etc/init.d/network restart
(ネットワークサービスの再起動)

インターフェース eth0 を終了中:                                [ OK ]
ループバックインターフェースを終了中:                            [ OK ]
ループバックインターフェイスを呼び込み中:                          [ OK ]
インターフェース eth0 を活性化中:                                [ OK ]

# ifconfig eth0
(eth0 の設定を表示)

eth0      Link encap:Ethernet HWaddr 00:0C:29:FB:5F:23
          inet addr:192.168.1.70 Bcast:192.168.1.255 Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:febf:5f23/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:73136 errors:0 dropped:0 overruns:0 frame:0
            TX packets:46269 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:15664826 (14.9 MiB) TX bytes:7723381 (7.3 MiB)
            Interrupt:185 Base address:0x1400
```

/etc/init.d/network スクリプトに stop オプションを付けるとネットワークインターフェースを無効にし、start オプションを付けると無効なネットワークインターフェースを有効にできます。

10.1.13 ルーティングの確認

ネットワークの状況を表示するための netstat コマンドがあり、netstat コマンドでデータが流れる方向(ルーティング)を調べられます。

書式

```
netstat
```

ネットワークの状況を表示します。

オプション

```
-r
```

コンピュータのルーティング(ゲートウェイへの流れ)を表示します。

```
-n
```

IP アドレスを数値で表示します。
コンピュータのルーティングを表示します。

○実習: ルーティングとインターフェースの確認

netstat コマンドを利用しルーティングテーブルを表示してみましょう。

```
$ netstat -nr
```

(ルーティングを表示)

Kernel IP routing table

Destination	Gateway	Genmask	Flags	MSS	Window	irtt	Iface
192.168.1.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
169.254.0.0	0.0.0.0	255.255.0.0	U	0	0	0	eth0
127.0.0.0	0.0.0.0	255.0.0.0	U	0	0	0	lo
0.0.0.0	192.168.1.1	0.0.0.0	UG	0	0	0	eth0

結果は、全てのデータは 0.0.0.0 へ行き、0.0.0.0 のデータは 192.168.1.1 へ行くので、192.168.1.1 がゲートウェイであれば、ネットワークの外へデータが転送されます。

10.1.14 ルーティングの変更

ルーティングの変更を行なうには、route コマンドを使用します。route コマンドの実行には、管理者権限が必要となります。

書式

```
route
```

オプションを付けずに route コマンドを実行した場合、netstat -nr を実行した結果とほぼ同様の形式で、コンピュータのルーティングを表示します。

書式

```
route add [-net|-host] ターゲット [netmask サブネットマスク] [インターフェース]
```

add を指定すると、ルーティングに経路を追加します。ターゲットには、-net を指定した場合にはネットワークアドレスを、-host を指定した場合にはホストアドレスを宛先として設定します。サブネットマスクの指定を省略した場合は、ターゲットに設定したアドレスから判別されたクラスに該当するマスク値が設定されます。インターフェースを省略すると、カーネルが自動的に最適なデバイスを設定します。

○実習: ルーティングの追加

クラス C のネットワーク 192.168.2.0 への経路を追加してみます。

```
# route add -net 192.168.2.0 netmask 255.255.255.0 eth0
(経路を追加)
# route
(ルーティングを表示)
Kernel IP routing table
Destination      Gateway          Genmask        Flags Metric Ref  Use Iface
192.168.2.0      *               255.255.255.0    U     0      0      0 eth0
192.168.1.0      *               255.255.255.0    U     0      0      0 eth0
169.254.0.0      *               255.255.0.0     U     0      0      0 eth0
127.0.0.1        *               255.0.0.0       U     0      0      0 eth0
```

書式

```
route add default gw ターゲット [インターフェース]
```

default gw を指定すると、デフォルトの経路（適切な経路が他にない場合の宛先となるゲートウェイ）をルーティングに追加します。

10.1 TCP/IP とは

○実習: デフォルトルートの追加

192.168.1.1 をデフォルトルートとして追加してみます。

```
# route add default gw 192.168.1.1 eth0
(デフォルトルートを追加)
# route
(ルーティングを表示)
Kernel IP routing table
Destination     Gateway         Genmask        Flags Metric Ref    Use Iface
192.168.2.0     *              255.255.255.0 U     0      0        0 eth0
192.168.1.0     *              255.255.255.0 U     0      0        0 eth0
169.254.0.0     *              255.255.0.0   U     0      0        0 eth0
127.0.0.1       *              255.0.0.0    U     0      0        0 eth0
default         192.168.1.1   0.0.0.0     UG    0      0        0 eth0
```

上記の例ではインターフェースが一つしかないため、結果的には全ての通信が 192.168.1.1 を経由してネットワークの外へ転送されます。

書式

```
route del [-net|-host] ターゲット [netmask サブネットマスク] [インターフェース]
```

del を指定すると、ルーティングから経路を削除します。

○実習: ルーティングの追加

クラス C のネットワーク 192.168.2.0 への経路を削除してみます。

```
# route del -net 192.168.2.0 netmask 255.255.255.0 eth0
(経路を削除)
# route
(ルーティングを表示)
Kernel IP routing table
Destination     Gateway         Genmask        Flags Metric Ref    Use Iface
192.168.1.0     *              255.255.255.0 U     0      0        0 eth0
169.254.0.0     *              255.255.0.0   U     0      0        0 eth0
127.0.0.1       *              255.0.0.0    U     0      0        0 eth0
default         192.168.1.1   0.0.0.0     UG    0      0        0 eth0
```

10.1.15 DNS を使う設定

名前の機能を使うには/etc/nsswitch.conf ファイルで何を使うか指定します。ファイルを使う指定があれば/etc/hosts ファイルを参照し、DNS(Domain Name System: ドメイン・ネーム・システム)を使う指定があれば、/etc/resolv.conf ファイルを見て DNS サーバを使います。/etc/nsswitch.conf

ファイルでは hosts: の項目でファイル (files) や DNS(dns) を指定します。hosts: の項目のファイルや DNS の優先順位は記述された順番となります。

```
hosts:      files dns
```

/etc/resolv.conf ファイルでは nameserver で DNS サーバを 3 つまで指定でき、応答が返ってこない場合は順番に次に指定しているサーバに問い合わせします。

```
nameserver 192.168.1.1
nameserver 192.168.1.2
nameserver 192.168.1.3
```

/etc/hosts ファイルはホストの名前と IP アドレスの関係を静的に定義しておくファイルです。定義は次のように 1 行で 1 つの定義を書きます。

IP アドレス	名前 (FQDN)	ホスト名
例		
127.0.0.1	localhost.localdomain	localhost
192.168.1.50	test.lpi.or.jp	test

/etc/hosts ファイルは DNS サーバによる名前解決よりも優先されます。

10.1.16 名前 (FQDN)

インターネットでは IP アドレスだけを使うと人間にはわかり辛いので、FQDN(Fully Qualified Domain Name) と呼ばれる名前 (例 : www.lpi.or.jp) を使えます。IP アドレスと名前の対応は nslookup コマンドで調べられます。

書式

```
nslookup ターゲット
ターゲットの名前と IP アドレスを返します。
```

○実習: IP アドレスに対応するホスト名の問い合わせ

nslookup コマンドを利用して DNS へホスト名の問い合わせを行なってみましょう

```
$ nslookup 203.174.74.34
(203.174.74.34 のホスト名の問い合わせ)
Server:          192.168.1.1
```

```

Address:      192.168.1.1#53
Non-authoritative answer:
34.74.174.203.in-addr.arpa      name = sv1.lpi.jp.
(IP アドレスに対して FQDN)
Authoritative answers can be found from:
(略)

$ nslookup lpi.jp
(lpi.jp の IP アドレス問い合わせ)
Server:      192.168.1.1
Address:      192.168.1.1#53
Non-authoritative answer:
Name:        lpi.jp
Address:    203.174.74.34

```

10.1.17 ポート番号

TCP/IP で通信をする場合は、IP アドレスに加えてサービスごとにポート番号を使います。ポート番号がどのサービスに対応するかは規格として取り決められており、一般的なサービスはサービス番号との対応が/etc/services ファイルに書かれています。

表 10.6 ポート番号と対応サービス

項目	ポート番号範囲	内容
WELL KNOWN PORT NUMBERS	0～1023	一般的なポート番号
REGISTERED PORT NUMBERS	1024～49151	登録済みポート番号
DYNAMIC AND/OR PRIVATE PORTS	49152～65535	自由に利用できるポート番号

表 10.7 サーバで使われる主なポート番号と対応するサービス

ポート番号	サービス名
20	FTP (データ)
21	FTP (制御)
22	SSH
23	Telnet
25	SMTP
53	DNS
80	HTTP
443	HTTPS

10.1.18 サービスの確認

サーバではホームページを見せるなどのサービスを提供するプログラムを主に動かします。Linux のサービスやネットワークの状況を表示する netstat コマンドは、提供されているサービスを調べて表示することができます。

書式

```
netstat ターゲット
ネットワークの状況を表示します。ここではサービスの表示に使います。
```

オプション

```
-a
全てのサービスを表示します。

-t
TCP サービスを表示します。

-u
UDP サービスを表示します。
```

-t オプションか-u オプションだけでは現在接続しているサービスだけを表示し、-a オプションを付けてコマンドを実行すると待機状態のサービスも表示できます。

○実習: 提供されている TCP サービスの表示

netstat コマンドに-a オプションと-t オプションを付けて実行してください。

```
$ netstat -at
(TCP サービス一覧を表示)
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address          Foreign Address        State
tcp      1      0 192.168.1.60:39671      ns1.centos.org:http    CLOSE_WAIT
機中
tcp      1      0 192.168.1.60:39673      ns1.centos.org:http    CLOSE_WAIT
機中
tcp      1      0 192.168.1.60:39675      ns1.centos.org:http    CLOSE_WAIT
機中
tcp      1      0 192.168.1.60:58609      centos.at.multacom.com:http  CLOSE_WAIT
信中
tcp      0      0 ::ffff:192.168.1.70:ssh    ::ffff:192.168:nicetec-mgmt ESTABLISHED
信中
```

10.1 TCP/IP とは

○実習：提供されている UDP サービスの表示

netstat コマンドに-a オプションと-u オプションを付けて実行してください。

```
$ netstat -au
(UDP サービス一覧を表示)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address          Foreign Address      State
udp    0      0 *:filenet-tms            *:*                # 待機中
udp    0      0 *:722                  *:*                # 待機中
udp    0      0 *:725                  *:*                # 待機中
udp    0      0 *:mdns                 *:*                # 待機中
udp    0      0 *:sunrpc               *:*                # 待機中
udp    0      0 *:ipp                  *:*                # 待機中
udp    0      0 *:filenet-rpc           *:*                # 待機中
udp    0      0 *:mdns                 *:*                # 待機中
```

10.1.19 ネットワークセキュリティの設定

ネットワークセキュリティのために TCP ラッパー (TCP Wrapper) という機能が提供されています。TCP ラッパーでセキュリティを強化する場合は、サービスが TCP ラッパー機能を提供するライブラリ (プログラム) を利用している必要があります。アクセスを制限する /etc/hosts.deny ファイルとアクセスを許可する /etc/hosts.allow ファイルを用意する必要があります。2つの制御ファイルはサービスを提供するプログラムのリストに対してコロン (:) で区切り、適応するネットワークや IP アドレスやホスト名やドメイン名などのクライアントのリストを書き、ファイルによって許可または拒否します。クライアントのリストの後には:で区切り、制限に引っかかる場合に実行するシェルスクリプトを書けます。全てを表す時はリストの代わりに ALL と記述します。制御ファイルは次のように記述します。

プログラムのリスト : クライアントのリスト [: シェルスクリプト]

例

```
/etc/hosts.allow
sshd : 192.168.1.60
/etc/hosts.deny
ALL : ALL
```

2つのファイルに制御を記述して初めて設定が有効になります。/etc/hosts.deny ファイルで全てのアクセスを不可にしてから、/etc/hosts.allow ファイルで必要なサービスを解放すると、安全な設定になるでしょう。

10.1.20 ファイアウォールの設定

TCP/IP で使用されるポート番号（サービス）単位でセキュリティの設定ができる、ファイアウォールという機能が提供されています。ファイアウォールは `iptables` というコマンドを使用して設定します。`iptables` の実行には、管理者権限が必要となります。

○実習: ファイアウォールの表示

-L オプションを付けると、現在のファイアウォールの設定を確認できます。

```
# iptables -L
(ファイアウォールの設定を表示)
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
RH-Firewall-1-INPUT  all  --  anywhere             anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
RH-Firewall-1-INPUT  all  --  anywhere             anywhere

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain RH-Firewall-1-INPUT (2 references)
target     prot opt source               destination
ACCEPT    all  --  anywhere             anywhere
          :
          :
ACCEPT    tcp  --  anywhere             anywhere           state NEW tcp dpt:ssh
REJECT    all  --  anywhere             anywhere           reject-with icmp-host-prob
```

ディストリビューションによってはデフォルトではほとんどのサービスが抑制されており、サービスを使用するために設定を追加する必要があります。`iptables` コマンドにて個別に追加することも可能ですが、リブートすると追加した設定が消えてしまいます。RedHat Linux 系のディストリビューションは `iptables` の設定を `/etc/sysconfig/iptables` に記述しておき、サービスの起動時に設定されます。

○実習: ファイアウォールの設定変更

CentOS 6 を例に、ファイアウォールに Apache サービスを追加してみます。Apache サービスはデフォルトのポート番号（80）を使用しているものとします。以下の 1 行を `/etc/sysconfig/iptables` に追加します。

```
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 80 -j ACCEPT
```

10.1 TCP/IP とは

デフォルトの設定では、初めに許可するサービスを記述し、最後に拒否するルールを記述することで、ルールに記述されてないサービスは全て拒否されます。よって、追加するサービスは REJECT の行より上に記述する必要があります。

```
# vi /etc/sysconfig/iptables
:
:
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 80 -j ACCEPT
-A RH-Firewall-1-INPUT -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

追加後、iptables を再起動します。

```
# /etc/init.d/iptables restart
ファイアウォールルールを適用中: [ OK ]
チェインポリシーを ACCEPT に設定中 filter [ OK ]
iptables モジュールを取り外し中 [ OK ]
iptables ファイアウォールルールを適用中: [ OK ]
iptables モジュールを読み込み中 ip_conntrack_netbios_ns [ OK ]
# iptables -L 再起動後、ファイアウォールの設定を表示
:
:
ACCEPT      tcp  --  anywhere           anywhere          state NEW tcp dpt:ssh
ACCEPT      tcp  --  anywhere           anywhere          state NEW tcp dpt:http
(Apache サービスが追加されたことを確認してください)

REJECT      all   --  anywhere           anywhere          reject-with icmp-host-proh
```

10.2 10 章 章末テスト

(1) 以下のうち、一般的に使われるサーバーが使うポートについて、正しい組み合わせを選びなさい。

1. 22:SSH 80:HTTP 443:HTTPS
2. 23:SSH 80:HTTP 443:HTTPS
3. 22:SSH 443:HTTP 80:HTTPS
4. 23:SSH 443:HTTP 80:HTTPS

(2) traceroute コマンドの動作と利用方法を説明しなさい。

(3) マシンに設定されている IP アドレスをコマンドで表示しなさい。

(4) マシンの DNS サーバー参照先として 8.8.8.8 を設定しなさい。

(5) 192.168.100.20 からのアクセスを拒否するよう設定しなさい。

第11章

ファイル管理

ファイルシステムの説明を読みながら、ファイルシステムの管理コマンドに慣れることで、ファイルシステムの概念と管理操作を身に付けます。ファイルシステムの構造を把握するために説明するのは ext3 ファイルシステムにおけるファイルとディレクトリ、i ノード、リンクという用語です。用語を理解するために、df コマンド、du コマンド、fdisk コマンド、fsck コマンド、mount コマンド、umount コマンドに加え、ln コマンドを実行します。

この章の内容

- 11.1 Linux のファイル管理
 - 11.1.1 ファイルシステムとは
 - 11.1.2 パーティションとは
 - 11.1.3 Linux のディレクトリ構造
 - 11.1.4 ファイルシステムの作成方法
- 11.2 ディスクのパーティション
 - 11.2.1 パーティション分割する理由
 - 11.2.2 パーティションの分割
 - 11.2.3 ハードディスクを増設して利用可能にする
- 11.3 ファイルシステム
 - 11.3.1 ジャーナリング機能
 - 11.3.2 マウント状態の表示
 - 11.3.3 ファイルシステムの作成
 - 11.3.4 ラベル
- 11.4 マウント
 - 11.4.1 マウントポイント
 - 11.4.2 マウント (mount コマンド)
 - 11.4.3 アンマウント (umount コマンド)
- 11.5 スワップ領域の作成
 - 11.5.1 スワップファイルシステムの作成
 - 11.5.2 スワップの領域作成と利用

この章の内容

11.6 自動マウント

11.7 CD/DVD/USB メモリ (リムーバブルメディア) の利用

11.7.1 CD/DVD のマウント

11.7.2 CD/DVD のアンマウントと取り出し

11.7.3 アンマウントできない場合には

11.8 i ノード

11.8.1 i ノード情報の確認

11.8.2 i ノード番号の確認

11.9 ハードリンクとシンボリックリンク

11.10 ディスクを管理するコマンド

11.10.1 ファイルシステムのチェックと修復

11.10.2 fsck とジャーナリング機能

11.11 章末テスト

11.1 Linux のファイル管理

Linux のファイル管理にはハードディスクとファイルシステムとディレクトリ構造の知識が必要となります。ハードディスクを Linux で利用するために、ファイルシステムについて知ることが望ましく、コマンドで準備作業が必要です。利用目的に応じてファイルシステムを選び、コマンドでファイルシステムを準備する作業が必要となります。ディレクトリの階層標準を初めとする、ディレクトリに関する知識と状態を調べるコマンドを、習得しましょう。

11.1.1 ファイルシステムとは

ファイルシステムとは、ファイル名・更新日付などの属性データ・ファイルデータ本体を効率よく管理するためのしくみです。ファイルシステムにはいくつか種類があり、利用するファイルシステムにより、ファイルをアクセスした時に得られる効率や安全性が変わります。ファイルシステムの概要をつかむと、今までに出てきたコマンドの理解度も上がるでしょう。

11.1.2 パーティションとは

ハードディスクを利用するため、ハードディスクを区切った単位です。ディスクの内部を複数のパーティション（領域）に区切る作業が必要となります。パーティションを分割する作業をパーティショニングと呼びます。

11.1.3 Linux のディレクトリ構造

ファイル単位でデータを管理すると段々とファイル数が多くなり、ファイル名の管理が難しくなります。ファイル管理の効率化を図るために、多くのファイルシステムと同様に ext3 ファイルシステムでも、ディレクトリという入れ物が用意されています。ディレクトリは複数のファイルをまとめて管理できる上に、他のディレクトリもまとめられます。ディレクトリを使うと、利用目的ごとにファイルをまとめたり、幾つかのディレクトリをディレクトリに入れて管理できるでしょう。Linux を始めとする Linux のディレクトリ構造は / ディレクトリが大元になるツリー状となっています。

/ディレクトリ

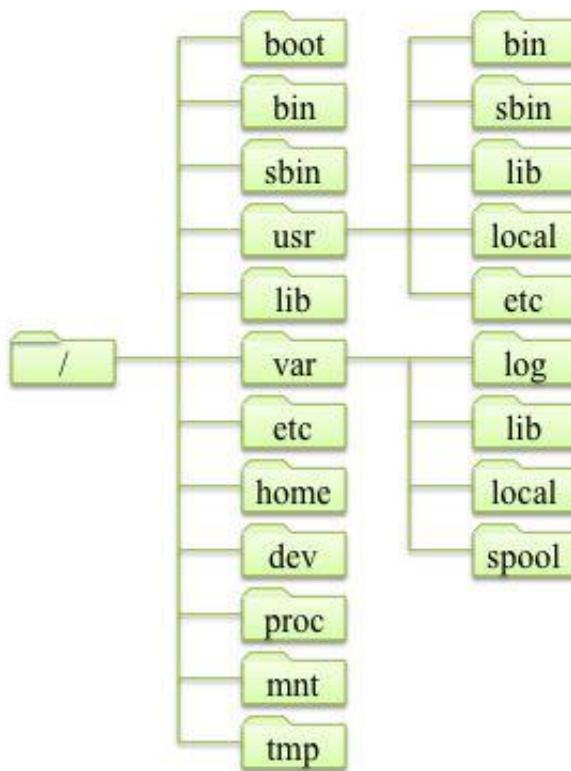
大元になるディレクトリで” / ” をルートと読み、/ ディレクトリをルートディレクトリと読みます。/ ディレクトリにディレクトリやファイルが入ります。

マウント

ディスク（ハードディスクを始めとして DVD-ROM ドライブや RAM ディスクなど）を利用するためには 1 つ目を / ディレクトリに必ずマウントします。複数のディスクを利用する場合は、2 つ目以降のディスクを既にマウントされているディスクの中のディレクトリにマウントします（既に中身があるディレクトリにディスクをマウントすると、マウントされたディレクトリが / ディレクトリであっても中身は置き換わり、今まであったディレクトリの中身は読み書きできなくなります）。

FHS(Filesystem Hierarchy Standard)

ファイルは必ずディレクトリに入るので、必ず/ディレクトリから始まる記述ができます。Linux システムは FHS と呼ばれる規格に基づいてディレクトリの構成が定められています。一般ユーザ用プログラムは bin ディレクトリ、システム管理用プログラムは sbin ディレクトリ、設定ファイルは etc ディレクトリ、複数のプログラムで共通に使われるライブラリは lib ディレクトリ、可変的なデータ（ログ・データベース・ウェブサイトなど）は var ディレクトリという緩やかな決まりがあります。/usr ディレクトリの下にも同じ構成のディレクトリがあり、ユーザーが独自にインストールするファイルは/usr/local ディレクトリの下に置く場合もあるでしょう。上記のディレクトリのほか、起動関連のファイルが格納される boot ディレクトリ、デバイスファイルが格納される dev ディレクトリ、ユーザの作業領域が格納される home ディレクトリ、一時ファイルが格納される tmp ディレクトリなどが FHS で定められているディレクトリとしてあげられます。



11.1.4 ファイルシステムの作成方法

ハードディスクを利用するには以下の手順で作成して利用できます。

- パーティションの作成・・・fdisk コマンド
- ファイルシステムの作成・・・mkfs コマンド
- マウント・・・mount コマンド

以降に出てくるコマンドは/sbin ディレクトリにあるシステム管理用のコマンドです。多くのシステム管理用のコマンドは root ユーザで(管理者になってから)実行する必要があります。

一般ユーザはシステム管理の権限がないため、システム管理用のコマンドを実行できなかったり、システム管理用のコマンドを実行できても必要な資源(ハードディスクなど)にアクセスできなかったりします。また、一般ユーザの初期設定では、システム管理用のコマンドが置いてある/sbin ディレクトリが PATH の設定に含まれていないため、/sbin/fdisk の様に絶対パス(/ディレクトリからの指定)でコマンドを実行する必要があります。本教科書の実行例で、行頭が#になっているコマンドは root ユーザでの作業が必要です。一般ユーザのときは su - コマンドで root ユーザになってからコマンドを実行してください。コマンドの詳しい内容は後ほど説明します。

```
$ su -
パスワード: xxxxxxxx
(root のパスワードを入力)
#
(root ユーザに切り替わる)
```

11.2 ディスクのパーティション

fdisk コマンドで、分割されたパーティションの情報を調べたり、パーティションを作成、削除したりできます。

11.2.1 パーティション分割する理由

パーティションを分割するのは以下の理由があります。

- システムとデータのバックアップ頻度を差別化する
- 障害発生時の影響範囲を狭める
- ファイルアクセスの速度向上

11.2 ディスクのパーティション

書式

```
fdisk [オプション] [デバイスファイル]
ディスクの構造を表示します。
ディスクをパーティション分割します。
```

オプション

```
-l
デバイスのパーティション情報を表示します。
```

○実習: ハードディスクのパーティション情報の表示

fdisk コマンドを実行して現在認識されているハードディスクのパーティション情報を表示しましょう。以下の例のようにコマンドを実行してみてください。

```
# fdisk -l
(ハードディスクのパーティションを表示)
ディスク /dev/sda: 6442 MB, 6442450944 バイト
ヘッド 255 , セクタ 63 , シリンダ 783
Units = シリンダ数 of 16065 * 512 = 8225280 バイト
(略)
デバイス ブート      始点      終点      ブロック  Id システム
/dev/sda1    *          1        587     4715046  83 Linux
/dev/sda2            588       783    1574370  82 Linux swap / Solaris
(2つのデバイス情報が表示される)
```

fdisk コマンドで表示された情報から、ここでは接続されているハードディスクのデバイス名は/dev/sda1 と/dev/sda2 の 2 つです。1 つ目のハードディスク (sda) が 2 つに分割されているのがわかります。1 つめのパーティションは 4715046 ブロック (4.5G バイト) で ID が 83(Linux 用)、2 つ目のハードディスクは 1574370 ブロック (1.5G バイト) で ID が 82(スワップ用) です。



11.2.2 パーティションの分割

コンピュータのハードディスクとして主に SATA と SAS の 2 つのインターフェース (周辺機器を接続するためのハードウェア) 規格が広く使われています。ハードディスクは、1 つのパーティションとして使うか、2~4 つまでのパーティションに分割できます。各パーティションは、基本パーティションまたは拡張パーティションとなります。拡張パーティションは 1 台のハードディスクに 1 つだけ作ることができ、拡張パーティションの中には、さらに論理パーティションを複数作ることができます。

パーティション分割の例

- 基本パーティションを 4 つ
- 基本パーティションを 3 つ + 拡張パーティションを 1 つ (論理パーティションを 2 つ)
- 論理パーティションは、基本パーティションと同様にファイルシステムを作成して、ファイルやディレクトリを保管できます。拡張パーティションは、論理パーティションを格納する役割となっており、ファイルシステムを作成することはできません。

パーティションの最大数

- IDE : 63 (基本 3 + 論理 60)
- SATA : 15 (基本 3 + 論理 12)
- SCSI : 15 (基本 3 + 論理 12)

Linux では 4 つ以上のパーティションが必要となるケースが多いので、拡張パーティション 1 つと他の 3 つを基本パーティションという構成で使う場合が多くなります。また、拡張されたハードディスクは基本パーティション 1 つや、LVM(Logical Volume Manager) 機能を使って複数のハードディスクを統合した 1 つのハードディスクとして大きく使うことが多いでしょう。

11.2.3 ハードディスクを増設して利用可能にする

Linux マシンにハードディスクを増設して利用可能な状態にします。新しいハードディスクには、データ用の領域と、スワップ領域 (Linux がメモリの代わりにプログラムを置いておく領域) の 2 つを作成します。利用可能にするには、fdisk コマンドでパーティションを 2 つに分割します。パーティションは分割した後からサイズ等の変更が難しいので、あらかじめどのようにパーティションを分割するのか計画してから行ないます。また、既に使用しているハードディスクのパーティションを操作すると、大切なデータが消えてしましますので、細心の注意を払って進めてください。パーティションを分割する作業をしているときに、指定サイズや分割数の設定ミスに気づいたら、パーティションの変更を保存しないで直ちに終了してください。

パーティションの操作はシステム管理の仕事なので、root ユーザで fdisk コマンドを実行する必要があります。可能であれば、他のユーザーがシステムを利用できないランレベル 1(ハードディスクの管理をするときに指定するモードで、管理者のみがコマンドを実行。通常はランレベル 3 やランレベル 5 で動作) の状態で作業をしてください。

ハードディスクを利用するまでの手順

1. 増設前のデバイス構成の確認
2. ハードディスクを増設
3. 増設後のデバイス構成の確認
4. パーティション情報の確認
5. パーティションの作成
6. パーティションの種類の変更
7. パーティション情報の保存
8. 再起動 (警告が表示されて、必要な場合)
9. ファイルシステムの作成
10. マウント
11. スワップパーティションの作成
12. スワップの追加
13. 自動マウントの設定

○実習: 増設前のデバイス構成の確認

まず、ディスクを増設する前のデバイスの構成を確認します。以下のコマンドを実行して確認してみましょう。

```
$ ls -l /dev/sd*
```

brw-rw----. 1 root disk 8, 0 6月 20 11:09 2012 sda
brw-rw----. 1 root disk 8, 1 6月 20 11:09 2012 sda1
brw-rw----. 1 root disk 8, 2 6月 20 11:09 2012 sda2
brw-rw----. 1 root disk 8, 16 6月 20 11:09 2012 sdb

```
brw-rw----. 1 root disk      8,  17 6月 20 11:09 2012 sdb1
```

/dev ディレクトリは Linux が認識するすべてのデバイスが置かれています。このうち、hd*もしくは sd*のような名前がついているデバイスがハードディスクドライブになります。

数字無しのデバイスはディスクドライブ自体を示しています。このことから、このシステムには 2 つのハードディスクドライブが認識されていることがわかります。数字付きのデバイスはハードディスクドライブが認識する、ファイルシステムを示します。sda には 2 つのパーティションが存在し、sdb には 1 つのパーティションが存在することがわかります。

○実習: ハードディスクを増設

CentOS をシャットダウンして、新しいハードディスクを増設します。増設したハードディスクは新しいデバイスとして認識されます。

ハードディスクは増設しただけでは利用できません。このあと以下の作業を行ないます。

- fdisk でパーティションを設定
- Physical Volume の作成
- mkfs コマンドを実行してディスク作成
- マウントおよび自動マウントの設定

○実習: 増設後のデバイス構成の確認

ハードディスクを増設したらデバイスとして認識されているか、以下のコマンドを実行して確認してみましょう。

```
$ ls -l /dev/sd*
brw-rw----. 1 root disk      8,  0 6月 20 11:09 2012 sda
brw-rw----. 1 root disk      8,  1 6月 20 11:09 2012 sda1
brw-rw----. 1 root disk      8,  2 6月 20 11:09 2012 sda2
brw-rw----. 1 root disk      8, 16 6月 20 11:09 2012 sdb
brw-rw----. 1 root disk      8, 17 6月 20 11:09 2012 sdb1
brw-rw----. 1 root disk      8, 32 6月 20 11:09 2012 sdc
```

増設したハードディスクが、デバイス/dev/sdc として追加・認識されました。

○実習: パーティション情報の確認

fdisk コマンドに /dev/sdc デバイスを指定して起動します。p コマンドで現在のパーティション情報を表示できます。パーティションが分割されていないハードディスクであれば何も表示されません。パーティションの分割がされていてもパーティションとして確保していない場所があれば、新たなパーティションを作成することができます。

```
# fdisk /dev/sdc
コマンド (m でヘルプ): p 現在のパーティション情報を表示

ディスク /dev/sdc: 6442 MB, 6442450944 バイト
ヘッド 255 , セクタ 63 , シリンダ 783
Units = シリンダ数 of 16065 * 512 = 8225280 バイト
(略)
デバイス ブート 始点 終点 ブロック Id システム

分割されていない状態で何も表示されない
```

○実習: パーティションの作成

n コマンドで新しいパーティションを確保します。コマンドアクションでは p を指定して、基本パーティションを 2 つ作成します。パーティションの先頭シリンド番号と、パーティションのサイズを指定します。パーティションを確保したら、p コマンドで確保されたパーティションを確認してください。

```
# fdisk /dev/sdc

コマンド (m でヘルプ): m
(ヘルプを表示)

コマンドの動作
a ブート可能フラグを付ける
b bsd ディスクラベルを編集する
c dos 互換フラグを付ける
d 領域を削除する
l 既知の領域タイプをリスト表示する
m このメニューを表示する
n 新たに領域を作成する
o 新たに空の DOS 領域テーブルを作成する
p 領域テーブルを表示する
q 変更を保存せずに終了する
s 空の Sun ディスクラベルを作成する
t 領域のシステム ID を変更する
u 表示/項目ユニットを変更する
v 領域テーブルを照合する
w テーブルをディスクに書き込み、終了する
x 特別な機能 (エクスパート専用)

コマンド (m でヘルプ): n
(新しいパーティション領域の確保)

コマンドアクション
e 拡張
p 基本パーティション (1-4)
p
(p を入力して基本パーティションを作成)

パーティション番号 (1-4): 1
```

(初めてパーティションを作成するので 1)

最初 シリンダ (1-783, default 1): 1
 (デフォルトの先頭シリンドを指定)

Last シリンダ,+ シリンダ数 or +size(K,M,G) (1-783, 初期値 783): +1024M
 (確保するサイズ)

コマンド (m でヘルプ): n
 (新しいパーティションの確保)

コマンドアクション
 e 拡張
 p 基本パーティション (1-4)
p
 (基本パーティションを作成)

パーティション番号 (1-4): 2
 (2つ目のパーティションを作成するので 2)

最初 シリンダ (126-783, default 126): 126
 (先頭シリンドを指定)

Last シリンダ,+ シリンダ数 or +size(K,M,G) (126-783, 初期値 783): +1024M
 (確保するサイズ)

コマンド (m でヘルプ): p
 (現在のパーティション情報を表示)
 ディスク /dev/sdc: 6442 MB, 6442450944 バイト
 ヘッド 255, セクタ 63, シリンダ 783
 Units = シリンダ数 of 16065 * 512 = 8225280 バイト
 (略)
 デバイス ブート 始点 終点 ブロック Id システム
 /dev/sdc1 1 125 1004031 83 Linux
 /dev/sdc2 126 250 1004062+ 83 Linux
 (確保したパーティションがリスト表示される)

シリンドを指定する箇所が 2 箇所あります。パーティションの分割は途中を空けないで連続して指定するので、最初シリンドの数はデフォルトのシリンド数をそのまま使えます。終点シリンドの数の代わりに + を付けたサイズを指定できます。シリンド数を計算して指定するなどの面倒な作業は必要ないので、ここではシリンド数に関しては考慮する必要がありません。

○実習: パーティションの種類の変更

パーティションは、その使用目的によって種類が指定されています。確保したばかりのパーティションは、Linux のファイルシステム用である Id の 83(Linux 用) が指定されています。スワップとして利用するためには、t コマンドで種類を変更してください。スワップパーティションとして利用するには、ID に 82(スワップ用) を指定します。指定できる種類は、種類を設定する途中で、L コマンドを使って一覧表示できます。

```

コマンド (m でヘルプ): t          (確保したパーティションの情報を変更)
パーティション番号 (1-4): 2
16進数コード (L コマンドでコードリスト表示): L  (パーティションの種類のリスト表示)
 0 空           1e Hidden W95 FAT1 80 古い Minix      be Solaris boot
 1 FAT12        24 NEC DOS       81 Minix / 古い    bf Solaris
 2 XENIX root   39 Plan 9       82 Linux swap / So c1 DRDOS/sec (FAT-
 3 XENIX usr    3c PartitionMagic 83 Linux          c4 DRDOS/sec (FAT-
 4 FAT16 <32M   40 Venix 80286  84 OS/2 隠し c:  c6 DRDOS/sec (FAT-
 5 拡張パーティション 41 PPC PReP Boot 85 Linux 拡張頒    c7 Syrinx
 6 FAT16        42 SFS          86 NTFS ポリュ      da 非 FS デー
 7 HPFS/NTFS   4d QNX4.x      87 NTFS ポリュ      db CP/M / CTOS /
 8 AIX          4e QNX4.x 2nd part 88 Linux plaintext de Dell ユーテ
 9 AIX ブート   4f QNX4.x 3rd part 8e Linux LVM      df BootIt
 a OS/2 ブート  50 OnTrack DM   93 Amoeba         e1 DOS access
 b W95 FAT32   51 OnTrack DM6 Aux 94 Amoeba BBT     e3 DOS R/O
 c W95 FAT32 (LBA) 52 CP/M      9f BSD/OS        e4 SpeedStor
 e W95 FAT16 (LBA) 53 OnTrack DM6 Aux a0 IBM Thinkpad eb BeOS fs
 f W95 Ext'd (LBA) 54 OnTrackDM6  a5 FreeBSD        ee EFI GPT
10 OPUS         55 EZ-Drive     a6 OpenBSD        ef EFI (FAT-12/16/
11 隠し FAT12   56 Golden Bow   a7 NeXTSTEP      f0 Linux/PA-RISC
12 Compaq 診断 5c Priam Edisk  a8 Darwin UFS     f1 SpeedStor
14 隠し FAT16 <3 61 SpeedStor   a9 NetBSD        f4 SpeedStor
16 隠し FAT16   63 GNU HURD また ab Darwin boot    f2 DOS セカン
17 隱し HPFS/NTF 64 Novell Netware b7 BSDI fs      fd Linux raid 自
18 AST SmartSleep 65 Novell Netware b8 BSDI スワッ fe LANstep
1b Hidden W95 FAT3 70 DiskSecure Mult bb 隠し Boot Wiz ff BBT
1c Hidden W95 FAT3 75 PC/IX

```

16進数コード (L コマンドでコードリスト表示): 82
(スワップ領域を指定)

Changed system type of partition 2 to 82 (Linux swap / Solaris)

○実習: パーティション情報の保存

パーティション情報は、w コマンドでハードディスクに書き込むと変更されます。パーティション情報の再読み込みが行えなかった場合には、警告が表示されるので、システムを再起動してください。

パーティション情報をハードディスクへ書き込まないで、パーティションの変更を破棄したい場合には、fdisk を q コマンドで終わらせます。

```

コマンド (m でヘルプ): w
(ハードディスクにパーティション情報を書き込む)

The partition table has been altered!
Calling ioctl() to re-read partition table.
WARNING: Re-reading the partition table failed with error 16: Device or resource busy.
The kernel still uses the old table.
The new table will be used at the next reboot.
Syncing disks.
(ワーニングメッセージは出ますが、書き込みは終了できます)
コマンド (m でヘルプ): q

```

コマンド 意味

- w =書き込み、保存 (write)
- q=変更を書き込まない、破棄 (quit)

11.3 ファイルシステム

ファイルシステムとは、ディレクトリやファイルの情報やデータをディスクの何処に保存してあるか管理するためのシステム(仕組み)です。ファイルシステムはディレクトリやファイルの、ファイル名やファイルの作成時間やファイルの所有権限などの情報を管理しています。また、ディスクのデータが書いてある場所とデータが書いていない場所なども管理しています。

ファイルシステムはオペレーティングシステムによって利用されるファイルシステムが異なります。Linux の場合はそれぞれのディストリビューションにより採用するファイルシステムは異なりますが、多くの場合は ext3 もしくは ext4 が利用されています。Windows の場合はバージョンは異なるものの多くの領域を管理できる NTFS が利用されている事がほとんどでしょう。CD や DVD、BD では ISO9660 という規格のファイルシステムが使われます。

各オペレーティングシステムで利用できるファイルシステムは以下の通りです^{*1}。

表 11.1 Linux で利用できるファイルシステムの一例

FS 名	内容	最大サイズ	最大ファイルサイズ
ext2	Berkeley Fast File System を参考にした 16 ビット構造	16TB	2TB
ext3	ext2 にジャーナル機能付加と機能拡張	2~32TB	16GB~2TB
ext4	ジャーナリングファイルシステム。ファイルの断片化防止などさまざまな機能拡張によりパフォーマンスが向上	1EB	16TB
ReiserFS	小さなファイルの扱いに向いたジャーナリングファイルシステム	8TB	16TB
XFS	SGI 社が提供したジャーナリングファイルシステム	8EB	8EB
JFS	IBM 社が提供したジャーナリングファイルシステム	512TB	4PB~8EB

以降の解説では ext3/4 の主要機能を解説します。

*1 容量の単位:

1M(メガ) バイト=1024K バイト 1G(ギガ) バイト=1024M バイト
1T(テラ) バイト=1024G バイト 1P(ペタ) バイト=1024T バイト
1E(エクサ) バイト=1024P バイト 1Z(ゼタ) バイト=1024E バイト

表 11.2 Windows で利用できるファイルシステムの一例

FS名	内容	最大サイズ	最大ファイルサイズ
FAT16	セクタという領域を 1 次元配列で管理するファイルシステム。現在は使われない	2GB(NT:4GB)	2GB(NT:4GB)
FAT32	セクタという領域を 1 次元配列で管理するファイルシステム。外付ディスクなどで使われており、Windows 以外でも利用できる	2TB(2KB セクタ:8TB)	4GB
NTFS	NT/XP/Vista 用ジャーナリングファイルシステム	256TB	16TB(16EB)
exFAT	フラッシュメディア向けファイルシステム	8ZB(512B セクタ時)	16TB(16EB)

表 11.3 CD/DVD のファイルシステム

FS名	内容
ISO9660	ISO(国際標準化機構)で標準化されたファイルシステム、Joliet や RockRidge や Apple の規格拡張が存在、CD や DVD に利用される

11.3.1 ジャーナリング機能

ジャーナリングとは、ファイルシステムに対する書き換え処理のコマンドをファイルシステムに逐一記録する機能です。ジャーナリング機能はコンピュータが急停止するなどの障害時に有効となるでしょう。コンピュータが急停止し、ファイルのデータがファイルシステムに書き込み途中である場合、処理されていないコマンドを対処します。ファイルの書き込み途中の場合、書き込み終わっていないデータが無くて書き込みを終わないので、途中までディスクに保存されているファイルの管理情報を消し、書き込み途中のファイルが作られなかった状態にします。ext3 ファイルシステム以降のファイルシステムの多くはこのジャーナリング機能を取り入れています。ext3 ファイルシステムは、ext2 ファイルシステムにジャーナリング機能 + α が追加されたファイルシステムです。ext3 のほかに、ReiserFS、XFS、JFS もジャーナリング機能を持ったファイルシステムです。

11.3.2 マウント状態の表示

df コマンドで現在マウントされているファイルシステムのリストを表示できます。

○実習: ハードディスクの確認

それでは実際に df コマンドを実行し、動きを見てみましょう。

```
# df
(マウントされているファイルシステムの一覧表示)
Filesystem      1K-ブロック    使用    使用可 使用 % マウント位置
/dev/sda1        4567236     3395908    935576  79% /
tmpfs           517660       0       517660   0% /dev/shm
/dev/hdc         3630332     3630332       0 100% /media/cdrom
```

sda ハードディスクの 1 つ目のパーティションである /dev/sda1 が / ディレクトリに、 tmpfs という特殊デバイスが /dev/shm ディレクトリに、 DVD-ROM の /dev/hdc が /media/cdrom にマウントされています。

11.3.3 ファイルシステムの作成

ハードディスクを利用するためには、パーティションに分割した後ファイルシステムを作成します。ファイルシステムの作成には mkfs コマンドを使用します。

書式

mkfs オプション

ファイルシステムを作成します。

オプション

-t
作成するファイルシステムの種類を指定する。

-c
ハードディスクの壊れている箇所を検出して、利用しない。

○実習: ハードディスクの作成

mkfs コマンドを実行してハードディスクを作成してみましょう。以下の例のようにコマンドを実行してみてください。

sd_c1(3 つ目のインターフェースの 1 つ目のパーティション) に ext3 形式のファイルシステムを作成

```
# mkfs -t ext3 -c /dev/sdc1
mke2fs 1.39 (29-May-2006)
Filesystem label=opt
(ハードディスクのラベル)
OS type: Linux
Block size=4096 (log=2)
```

```
(ハードディスクのパーティションのサイズ)
Fragment size=4096 (log=2)
70400 inodes, 140568 blocks
7028 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=146800640
5 block groups
32768 blocks per group, 32768 fragments per group
14080 inodes per group
Superblock backups stored on blocks:
      32768, 98304
Checking for bad blocks (read-only test): done
Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done
This filesystem will be automatically checked every 22 mounts or
180 days, whichever comes first. Use tune2fs -c or -i to override.
```

11.3.4 ラベル

ラベルとは、ハードディスクなどのパーティションを認識するための名前です。基本パーティションや論理パーティションにファイルシステムを作るときにラベルを指定するか、または e2label コマンドでラベルを付けられます。

書式

```
e2label デバイス [ラベル]
パーティションのラベルを表示します。
パーティションのラベルを付けます。
```

○実習: e2label コマンドでハードのラベル名を変更

e2label コマンドにデバイスを指定するとラベルを確認できます。e2label コマンドにデバイスとラベルを指定するとデバイスのラベルを変更できます。

```
# e2label /dev/sda1
/1
(/dev/sda1 のラベルを表示)

# e2label /dev/sdc3
opt
(/dev/sdc3 のラベルを表示)

# e2label /dev/sdc3 opt2
(/dev/sdc3 のラベルを opt2 に変更)

# e2label /dev/sdc3
opt2
```

```
(/dev/sdc3 のラベルを表示)
```

11.4 マウント

ハードディスクはパーティション分割してファイルシステムを作成してからマウントすると読み書きできます。CD-ROM や DVD-ROM などのリムーバブルメディアもマウントすると読めるようになります。

11.4.1 マウントポイント

ハードディスクや CD-ROM などを利用する場合は、存在しているディレクトリにマウントする必要があります。マウントするために利用するディレクトリをマウントポイントと呼びます。/として利用されるファイルシステムは/にマウントされており、後から利用するファイルシステムも、必ずどこかのディレクトリにマウントして初めて利用できます。

11.4.2 マウント (mount コマンド)

ハードディスクやリムーバブルメディアなどをマウントポイントにマウントするために mount コマンドを使います。

書式

```
mount -t タイプ -o オプション デバイスファイル マウントポイント  
ハードディスクやリムーバブルメディアをマウントします。
```

オプション

-t タイプ
ファイルシステムの ext3 や Windows の msdos、CD と DVD の iso9660 など

-o オプション
読み書きの rw や読み取り専用の ro など

デバイスファイル
ファイルシステムをアクセスするためのデバイスファイル

マウントポイント
マウントするディレクトリ

11.4.3 アンマウント (umount コマンド)

マウントされたハードディスクやリムーバブルメディアを利用しなくなると、アンマウントします。アンマウントするために umount コマンドを使います。

書式

```
umount マウントポイント  
ハードディスクやリムーバブルメディアをアンマウントします。
```

○実習: ハードディスクのパーティションのマウント

mount コマンドで sdc2(3 つ目のドライブの 2 つ目のパーティション) を /opt ディレクトリへマウントしてください。ファイルシステムが、読み書きできるように rw オプションを付けました。

マウントしているファイルシステムは ls コマンドおよび df コマンドで確認できます。

```
# mount -t ext3 -o rw /dev/sdc2 /opt  
(/dev/sdc2 を /opt ディレクトリへ ext3 形式でマウント)  
# ls /opt  
lost+found  
  
# df  
Filesystem      1K- ブロック    使用    使用可  使用 % マウント位置  
/dev/sda1        4567236    3444596    886888   80%  /  
tmpfs           517660        0    517660   0%  /dev/shm  
/dev/sdc2        553408    16840    508456   4%  /opt
```

○実習：ハードディスクのパーティションのアンマウント

umount コマンドで /opt にマウントされているファイルシステムをアンマウントします。先の実習でマウントした /opt をアンマウントしてみましょう。umount コマンドを実行したら、ls コマンドおよび df コマンドでアンマウントされたことを確認してみてください。

```
# df
Filesystem      1K-ブロック    使用    使用可  使用 % マウント位置
/dev/sda1        4567236     3444596   886888  80% /
tmpfs           517660       0       517660   0% /dev/shm
/dev/sdc2        553408     16840     508456   4% /opt

# umount /opt                         マウント位置 /opt をアンマウント
# ls /opt

# df
Filesystem      1K-ブロック    使用    使用可  使用 % マウント位置
/dev/sda1        4567236     3444596   886888  80% /
tmpfs           517660       0       517660   0% /dev/shm
```

11.5 スワップ領域の作成

Linux のカーネルと呼ばれる根幹のプログラムは、ハードディスクからプログラムやデータをメモリ領域へ読み込んで実行します。プログラムやデータを新たに読み込むための空きメモリ領域がなくなると、Linux のカーネルは今利用していないメモリ上のプログラムやデータをスワップ領域へ一時的に退避します。スワップパーティションとは Linux のカーネルシステムがメモリの代わりとして一時的に使うハードディスクのパーティション領域です。ハードディスクのパーティションをスワップ領域として使うのであれば、メモリサイズに対応したパーティションを確保し、スワップファイルシステムをパーティションに作成する必要があります。

11.5.1 スワップファイルシステムの作成

スワップするためのパーティション確保と、ファイルシステムの作成は、インストールするときや新しくハードディスクを増設するときに実行します。スワップ領域（スワップのパーティション）を作るには `mkswap` コマンドを使います。

書式

```
mkswap デバイスファイル  
スワップファイルシステムを作ります。
```

オプション

```
-c  
不良な部分を探して利用しない  
  
デバイスファイル  
ファイルシステムをアクセスするためのデバイスファイル
```

11.5.2 スワップの領域作成と利用

スワップ領域は作成した後に `swapon` コマンドを実行すると有効化できるようになります。

書式

```
swapon [デバイスファイル]
```

オプション

-s
現在利用しているスワップ領域を表示

利用中のスワップ領域は swapoff コマンドで無効化できます。

書式

```
swapoff [デバイスファイル]
```

オプション

-v
指定したスワップを無効化
-a
すべてのスワップを無効化

○実習: スワップ領域を作成して有効化

早速以下の操作を行ない、スワップ領域を作成して利用できるようにしましょう。

1. mkswap コマンドでスワップ領域を作ります。
2. mkswap コマンドで Id が 82(スワップ用) であるパーティションに、スワップファイルシステムを作ります。
3. swapon コマンドに-s オプションを付けて、現在のスワップ領域の状況を表示してください。
4. swapon コマンドでスワップ領域を有効化します。

```
# fdisk -l
(ハードディスクのパーティションを表示)
ディスク /dev/sda: 6442 MB, 6442450944 バイト
ヘッド 255 , セクタ 63 , シリンダ 783
Units = シリンダ数 of 16065 * 512 = 8225280 バイト
(略)
デバイス ブート    始点    終点    ブロック  Id システム
/dev/sda1      *        1      587    4715046  83 Linux
/dev/sda2          588      712   1004062+  82 Linux スワップ / Solaris
/dev/sda3          713      783   570307+  83 Linux

# mkswap -c /dev/sda2
(/dev/sda2 をスワップファイルに変更)
Setting up swapspace version 1, size = 1036378 kB
```

11.5 スワップ領域の作成

```
# swapon -s
(利用開始しているスワップ領域を表示)
Filename           Type      Size     Used Priority
/dev/dm-1          partition 2064376  0      -1

# swapon /dev/sda2
(/dev/sda2 をスワップファイルシステムとして利用する)
# swapon -s
Filename           Type      Size     Used Priority
/dev/dm-1          partition 2064376  0      -1
/dev/sda2          partition 1012084  0      -4
(/etc/sad2 が追加された)
```

○実習: スワップ領域を無効化

先の手順で追加したスワップ領域を無効化してみましょう。

1. swapon コマンドに-s オプションを付けて、現在のスワップ領域の状況を表示してください。
2. swapoff コマンドでスワップ領域を無効化します。
3. swapon コマンドに-s オプションを付けて、スワップ領域が一つなくなっていることを確認します。

```
# fdisk -l
(ハードディスクのパーティションを表示)
ディスク /dev/sda: 6442 MB, 6442450944 バイト
ヘッド 255 , セクタ 63 , シリンダ 783
Units = シリンダ数 of 16065 * 512 = 8225280 バイト
(略)
デバイス ブート    始点    終点    ブロック  Id システム
/dev/sda1  *        1       587    4715046  83 Linux
/dev/sda2          588     712    1004062+  82 Linux スワップ / Solaris
/dev/sda3          713     783    570307+  83 Linux

# swapon -s
Filename           Type      Size     Used Priority
/dev/dm-1          partition 2064376  0      -1
/dev/sda2          partition 1012084  0      -4

# swapoff -v /dev/sda2
(/dev/sda2 のスワップファイルシステムを無効化)
/dev/sdc2 に swapoff

# swapon -s
(有効化しているスワップ領域を表示)
Filename           Type      Size     Used Priority
/dev/dm-1          partition 2064376  0      -1
```

/dev/sda2 のスワップ領域が表示されなくなったことを確認します。

11.6 自動マウント

インストール時に作られたパーティションは、コンピュータの起動時に自動マウントされます。起動時に自動マウントされる設定を記述するファイルは/etc/fstab というパスにある設定ファイルです。自動マウントには mount コマンドが使われます。増設したハードディスクを自動マウントするためには、/etc/fstab ファイルを変更してください。/etc/fstab ファイルに記述されるのは次の項目です。

- ブロックスペシャルデバイス
- マウントポイント
- ファイルシステムのタイプ (ext3,swap など)
- オプション (カンマで区切る)
- dump コマンドがダンプ (バックアップ) するか否か (0・・・ダンプ不要,1・・・ダンプ)
- ブート時にチェックする順番

書式

```
mount -a
デバイスをマウントします。
```

オプション

```
-a
/etc/fstab ファイルに指定された swap 以外のパーティションをマウント
```

○実習: 新しく作ったパーティションの自動マウント

先の手順で手動マウントした/dev/sda3 を自動で/opt にマウントするように設定しましょう。/etc/fstab に記述しておくことで、CentOS が起動時に自動的にマウントするようになります。

自動マウントする前に現状の構成を確認

```
# df
Filesystem      1K-ブロック   使用   使用可 使用 % マウント位置
/dev/sda1        4567236    3470024    861460  81% /
tmpfs            517660        0    517660  0% /dev/shm
```

fstab に自動マウントしたいパーティションを追記

```
# vi /etc/fstab
/dev/sda3      /opt      ext3      defaults      1 2
```

マウントの実行と確認

```
# mount -a
# df
Filesystem      1K-ブロック    使用    使用可    使用 %  マウント位置
/dev/sda1        4567236     3470024   861460   81% /
tmpfs           517660       0       517660   0% /dev/shm
/dev/sda3        553408     16840     508456   4% /opt
```

mount コマンドに-a オプションを付けた場合、fstab に記載されているファイルシステムをすべてマウントします。

再起動後、自動マウントされる事を確認

```
# reboot
# df
Filesystem      1K-ブロック    使用    使用可    使用 %  マウント位置
/dev/sda1        4567236     3470024   861460   81% /
tmpfs           517660       0       517660   0% /dev/shm
/dev/sda3        553408     16840     508456   4% /opt
```

新しいパーティションが、mount コマンドを実行しなくても自動マウントされる事を確認します。

11.7 CD/DVD/USB メモリ (リムーバブルメディア) の利用

電源を入れたまま取り外したり装着してメディアを取り替えられるリムーバブルな外部記憶装置として、CD/DVD ドライブや USB メモリがあります。リムーバブルメディアを使用する場合、手動でマウントをする必要があります（一部のディストリビューションでは、自動的にマウントされる場合があります）。リムーバブルメディアのマウントポイントとして、/mnt ディレクトリや/media ディレクトリの下に作られたディレクトリが使われます。

11.7.1 CD/DVD のマウント

CD/DVD を Linux で使用するには、mount コマンドでマウントを行ないます。さらにオプションとして、タイプに CD の規格である iso9660 を、モードに読み込み専用の ro (Read Only) を指定します。

○実習: DVD-ROM のマウント

DVD-ROM をマウントしてアクセスしてみます。CentOS のインストールメディアを端末の DVD ドライブに挿入してから実行してください。

```
# mkdir /media/cdrom
(マウントするためのディレクトリを作成)

# mount -t iso9660 -o ro /dev/cdrom /media/cdrom
(DVD-ROM をマウント)
# ls /media/cdrom
(マウントポイントを確認)
CentOS_BuildTag           RPM-GPG-KEY-CentOS-6      images
EULA                     RPM-GPG-KEY-CentOS-Debug-6   isolinux
GPL                      RPM-GPG-KEY-CentOS-Security-6 repodata
Packages                 RPM-GPG-KEY-CentOS-Testing-6
RELEASE-NOTES-en-US.html  TRANS.TBL
```

実習例では、デバイス/dev/cdrom を/media/cdrom ディレクトリにマウントしています。ディストリビューションによっては、/media/cdrom ディレクトリが存在しないので、代わりに/mnt を指定します。正常にマウントできたら、ls コマンドで DVD-ROM の中身を参照してください。

11.7.2 CD/DVD のアンマウントと取り出し

マウントした CD/DVD を取り出すには、umount コマンドでアンマウントを行ないます。

○実習: DVD-ROM のアンマウント

さきほどマウントした DVD-ROM をアンマウントしてみましょう。

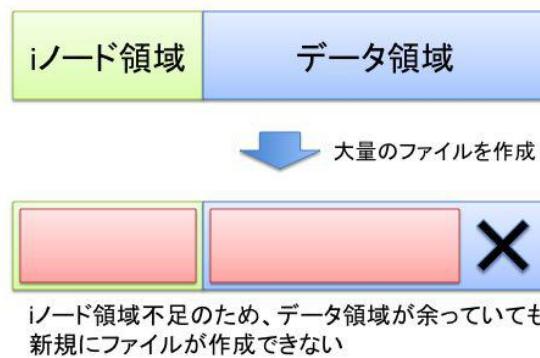
```
# umount /media/cdrom
(DVD-ROM をアンマウント)
# ls /media/cdrom
(マウントポイントを確認)
```

11.7.3 アンマウントできない場合

マウントポイントが使用中であったり参照中の場合はアンマウントすることはできません。具体的に言うと、マウントポイント以下がカレントディレクトリである場合や、リムーバブルメディアに含まれるファイルを参照している場合などです。ディレクトリを使用していないか確認し、カレントディレクトリを移動したり、処理を終了してからアンマウントしてください。特に複数の端末 ウィンドウを開いている場合は要注意です。

11.8 i ノード

ext3(ext2) ファイルシステムは、ファイルやディレクトリに対し、i ノード番号というユニークな番号を割り振って管理しています。ファイルシステムを作成した時に i ノード領域という場所が確保されます。i ノード領域には、ファイルがディスク上にある位置やアクセス権限などの情報が保持されています。ファイルシステムに作れるファイル数は、i ノード領域の大きさに左右されることとなります。もしファイルが多く作られて i ノード領域が足りなくなると、そのファイルシステムにデータを書き込める空き領域があっても、新規のファイルが作成できなくなります。



11.8.1 i ノード情報の確認

ext3 ファイルシステムの i ノード領域の使用状況を確認するには、df コマンドに -i オプションを付けて実行します。i ノード情報として、全体の数、使用している数、残りの数、使用量 (%) が表示されます。

書式

```
df -i
ファイルシステムの情報を表示します。
```

オプション

```
-i
i ノード情報の表示をします
```

○実習: i ノード情報の表示

df -i コマンドを実行して、システムの i ノード情報を表示してみましょう。以下の例のようにコマンドを実行してみてください。

```
# df -i
(i ノード情報を表示)
Filesystem      i ノード  i 使用   i 残り  i 使用 % マウント位置
(i ノードの使用情報が表示される)
/dev/sda1        1179648  150788  1028860    13% /
tmpfs           129415      1  129414     1% /dev/shm
/dev/hdc          0       0       0     -  /media/cdrom
```

表示されているデバイス/dev/sda1 は、Linux のファイルシステムである ext3 ファイルシステムとなっているので i ノード情報が表示されます。一方、DVD-ROM ドライブは ext3 ファイルシステムでない (ISO9660 規格のファイルシステム) ので i ノード情報を持たないことがわかります。

11.8.2 i ノード番号の確認

ファイルに割り振られている i ノード番号を確認するには、ls コマンドに -i オプションを付けて実行します。

○実習: i ノード番号の確認

ls -i コマンドを実行して、システムの i ノード番号を表示してみましょう。以下の例のようにコマンドを実行してみてください。

```
$ ls -il /bin | sort
(i ノード番号を表示)

(略)
750812 -rwxr-xr-x  1 root root  18568  2月 22  2005 setserial
750813 -rwxr-xr-x  1 root root  57488  5月  3  2007 cpio
750814 -rwxr-xr-x  3 root root  56080  7月 15  2007 gunzip
750814 -rwxr-xr-x  3 root root  56080  7月 15  2007 gzip
750814 -rwxr-xr-x  3 root root  56080  7月 15  2007 zcat
750815 -rwxr-xr-x  1 root root   6420 11月 26 22:52 dmesg
750816 -rwxr-xr-x  1 root root 31892 11月 18 02:55 setfont
(略)
```

実行例では、行頭 (1 番目) に i ノード番号が表示され、gunzip コマンド、gzip コマンド、zcat コマンドに同じ i ノード番号が割り振られています。ファイル名が別々でも i ノード番号が同じファイルは、ファイルとしての実体は同じなハードリンクという仕組みを表しています。3 番目に表示されている項目の 3 という数字はリンク数で、1 つのファイル実体を 3 つのファイル名でリンクしていることを表しています。

11.9 ハードリンクとシンボリックリンク

リンク機能は、ファイルをコピーしたり移動したりせずに、別のディレクトリにあるように扱うことができる機能です。たとえば、コマンドの引数としてファイルやディレクトリを指定する場合、パス指定が長くて複雑になると入力を間違えることがあるでしょう。リンク機能を使えば、よく使うファイルを自分のホームディレクトリに置いてあるように扱うことができます。リンクには、ハードリンクとシンボリックリンクの2種類があります。

ハードリンク

ハードリンクは、ファイルの実体を直接指示して共有します。ハードリンクを削除しても、元ファイルは削除されません。ハードリンクはiノード番号を共有することで実現しているので、別ファイルシステム（別パーティション）には作成できません。

シンボリックリンク

シンボリックリンクは、元ファイルが保管されている位置（パス）を示す擬似的なファイルを作ります。シンボリックリンクを消しても元ファイルに影響はありません。元ファイルを消すとシンボリックリンクからのアクセスがエラーとなります。シンボリックリンクは別ファイルシステムの間で作成することができます。Windowsではショートカットと呼ばれるファイルと考え方が同じです。

書式

```
ln 元ファイル名 リンク名
```

オプション

```
-s  
シンボリックリンクを作成します。  
-s オプションを付けない場合はハードリンクを作成します。
```

○実習: ハードリンクの作成

fileコマンドのコピーを用意し、lnコマンドでfileのハードリンクであるfile2を作ります。fileファイルを削除しても、file2が実行できるので、データが削除されないことがわかります。

```
$ cp -p /usr/bin/file .  
$ ln file file2  
(ハードリンクを作成)  
$ ls -il file*
```

```

559793 -rwxr-xr-x 2 root root 12428 5月 31 2007 file
(コピーされたファイル)
559793 -rwxr-xr-x 2 root root 12428 5月 31 2007 file2
(ハードリンクされたファイル)

$ rm file
(元ファイルを削除)
$ ./file2
(ハードリンクされたファイルを実行)
Usage: file2 [-bcikLnNsvz] [-f namefile] [-F separator] [-m magicfiles] file...
      file2 -C -m magicfiles
(ハードリンクされたファイルが実行できる)
Try `file --help' for more information.

# rm file2
(ハードリンクされたファイルを削除)

```

ファイルは必ず i ノード情報にリンクしており、ファイルを作るとハードリンクが 1 つできることと同じです。ファイルのハードリンクを作ると i ノードの領域にあるリンク数が 1 つ増え、ハードリンクを削除するとリンク数が 1 つ減ります。ファイルを消してリンク数が 0 になると、ファイルのデータがファイルシステムから完全に削除されます。

○実習: シンボリックリンクの作成

早速シンボリックリンクを作成してみましょう。ここでは less のマニュアルをファイル出力したものにシンボリックリンクを張って、シンボリックリンクを参照してみましょう。ls コマンドで調べるとシンボリックリンクファイルはパーミッション情報の先頭が l で表示されることも確認します。

```

$ man less > man-less
(man-less というファイルのシンボリックリンクを作成するための準備)

$ ln -s man-less doc-less
(man-less のシンボリックリンク (doc-less) を作成)

$ ls -li *-less
(シンボリックリンクを確認)

668311 lrwxrwxrwx. 1 tooyama tooyama     8 6月 14 11:56 2012 doc-less -> man-less
660339 -rw-rw-r--. 1 tooyama tooyama 89526 6月 14 11:55 2012 man-less

$ cat doc-less
(man-less と同様の内容が表示されることを確認)

(略)
[参考訳]
Copyright (c) 1994-2000 Kazushi (Jam) Marukawa, 日本語化ルーチンのみ。
この部分に関するコメントは jam@pobox.com へ送って下さい。
このパッチは Less ライセンスの下で配布できる。

```

11.10 ディスクを管理するコマンド

ハードディスクを管理するコマンドとして、ファイルシステムをチェックして修復する fsck コマンドや、ファイルやディレクトリが使っているディスク容量を調べる du コマンドがあります。

11.10.1 ファイルシステムのチェックと修復

コンピュータが異常終了するなど正常にシャットダウンが行われないと、ファイルシステムのファイル管理情報とハードディスクに書き込まれたデータとの間で辻褄が合わなくなることがあります。ファイルシステムが不整合に陥った場合、fsck コマンドを使って整合性をチェックし、修復を行なう必要があります。fsck コマンドは、異常終了した後のシステム起動時に自動的に実行されます。

書式

```
fsck デバイス名  
ファイルシステムのチェックと修復をします。
```

11.10.2 fsck とジャーナリング機能

ジャーナリング機能を持つ ext3 ファイルシステムは、ジャーナル情報があるので修復が素早く行えます。一方、ext2 ファイルシステムはジャーナル情報がないので、全ての管理情報が正しいかチェックする必要があるので修復にとても時間がかかります。

○実習: ハードディスクのチェック

fsck コマンドで ext3 ファイルシステムを修復します。fsck コマンドにはハードディスクのデバイスファイルとして /dev/sdc3 を指定しました。

```
# fsck /dev/sda3  
(/dev/sda3 をチェック)  
fsck from util-linux-ng 2.17.2  
e2fsck 1.41.12 (17-May-2010)  
opt: clean, 11/70400 files, 6426/140568 blocks  
(ファイルシステムのチェック)
```

マウントされているハードディスクをチェックして修復できますが、修復中にファイルシステムが書き換えられると、更に問題が増えるかもしれません。修復をするときは、チェックするパーティションがアンマウントされている状態での fsck コマンドの実行をお薦めします。

11.10.3 ディレクトリ使用量の確認

df コマンドでハードディスクなどの全体の使用量を確認できました。細かいディレクトリの使用量を調べるには du コマンドを使います。

書式

```
du [ディレクトリ]
du [ファイル]
ディレクトリの使用量を調べます
ファイルのサイズを調べます
```

オプション

```
-s
指定ファイルや指定ディレクトリの総計を表示します。
```

○実習: ディレクトリ使用量の確認

du コマンドに-s オプションを付けると、/usr ディレクトリにあるディレクトリ毎の全ファイルのサイズの総計を表示します。（/usr ディレクトリにファイルがあれば、ファイルのサイズも表示されます）

```
# du -s /usr/*
(/usr ディレクトリにある各ディレクトリのファイルサイズの総計を表示)

24      /usr/X11R6
9184    /usr/arc
307200   /usr/bin
8       /usr/etc
8       /usr/games
158984   /usr/include
1900    /usr/kerberos
927576   /usr/lib
53228   /usr/libexec
248     /usr/local
34540   /usr/sbin
1936436  /usr/share
82480   /usr/src
4       /usr/tmp
```

11.11 11章 章末テスト

- (1) マシンのハードディスクのパーティションを確認するコマンドを記述しなさい。
- (2) スワップファイルシステムについて説明しなさい。
- (3) ext3 や ext4 が ext2 より優れている点を 1 つあげなさい。
- (4) ハードリンクをシンボリックリンクの違いを説明しなさい。
- (5) ログインしているユーザーのホームディレクトリの使用量を表示するコマンドを記述しなさい。

Linux 標準教科書

●改訂履歴

2008 年 04 月 15 日 第 1 版 (Ver.1.0.0)

2008 年 11 月 13 日 第 2 版 (Ver.1.0.1)

2008 年 11 月 20 日 第 3 版 (Ver.1.0.2)

2009 年 08 月 26 日 Ver.1.1.0

2010 年 10 月 01 日 Ver.1.1.1

2012 年 10 月 01 日 Ver.2.0.0

●制作

特定非営利活動法人エルピー・アイ・ジャパン

●著者

岡田 賢治

川井 義治

佐久間 伸夫

宮原 徹

遠山 洋平

田口 貴久

●校正・協力

松田 神一

木村 真之介

高橋 征義

Linux 標準教科書メーリングリスト参加の皆様

Linux 標準教科書

2012 年 10 月 1 日 v2.0.0 版発行

発行所 LPI-Japan

Copyright(C) 2012 The Linux Professional Institute Japan. All Rights Reserved.