Knowledge Design

Knowledge Design Corporation
http://www.knowd.co.jp

UNIX
の基礎

D19990527
V1.0
Knowledge Design Corporation

「UNIX の基礎」コース 目次

イントロダクション

ようこそ、「UNIX の基礎」コースへ！ 0-3
もし、もっとたくさんのことを知りたくなったら 0-5

第１章 オペレーティング・システムについて

はじめに・・・ 1-3
コンピューターを構成している装置 1-5
OS の機能 1-7

第２章 UNIX のアーキテクチャ １

UNIX の歴史 2-3
オペレーティング・システムの構成要素 2-5
UNIX 環境の利用の仕方 - コマンド・インタプリタ - 2-7

第３章 ログイン・ログアウト

UNIX を使用するために必要な作業 3-3
ログイン・プロンプト 3-5
コマンド・プロンプト 3-7
パスワードの変更 3-9
ログアウト 3-11
主電源のオフ 3-13
第３章 ログイン・ログアウト 演習問題 3-15

第４章 UNIX コマンド

UNIX コマンド、ユーティリティ 4-3
コマンドライン 4-3
いくつかの簡単なコマンドの使用 4-5
date 4-5
man 4-5
cal 4-7
last 4-9
kon 4-9
clear 4-11
echo 4-11
sleep 4-11
bc 4-13
whoami 4-13
コマンド・ラインの便利な使い方 4-15
マルチ・ステートメント 4-15
過去に実行したコマンドの再実行 4-15
第４章 UNIX コマンド 演習問題 4-17

第５章 ウィンドウ・システム

X Window 5-3
X Window の外観と操作 5-5
ウィンドウ・システムの起動 5-7
アプリケーションの起動と終了 5-9
一般的なウィンドウ操作 5-11
コマンドラインからのアプリケーションの起動 5-13
Netscape Communicator を使用してウェブにアクセスする 5-15
Netscape Communicator を利用して電子メールを送受信する 5-17
ウィンドウシステムの終了 5-21
第５章 練習問題 5-23

第６章 ファイルとディレクトリの操作

情報（データ）保存方法 6-3
ファイルの表現方法 6-5
ファイル操作コマンド 6-11
ファイル、ディレクトリのコピー 6-21
第６章 ファイルとディレクトリの操作 演習問題 6-33

第７章 vi エディタ

vi エディタ 7-3
削除 7-7
挿入 7-9
コピー・ペースト 7-11
大きなファイルの編集に便利なコマンド 7-15
編集のために便利な機能 7-17
環境の設定 7-19
vi の環境設定用のファイル .exrc 7-21
第７章 vi エディタ 演習問題 7-23

第８章 ファイルのパーミション

ファイルの保護 8-3
パーミションの読み方 8-5
パーミションの種類 8-7
パーミションの変更 8-9
デフォルトのパーミション 8-11
第８章 ファイルのパーミション 演習問題 8-13

第９章 プロセスの実行と監視

プロセスとは？ 9-3
プロセスの実行 9-5
プロセスの稼動状況の監視 9-7
プロセスの強制終了 9-11
第９章 プロセスの実行と監視 演習問題 9-13

第１０章 ネットワーク・クライアント

ネットワーク機能 10-3
コンピューターの名前 10-5
ネットワークが起動しているかを確認する 10-7
ホスト名と IP アドレスの変換ができているかを確認する 10-7
リモートによるログイン 10-9
リモートによるファイルのコピー、転送 10-11
リモートによるコマンドの実行 10-13
リモートによるログイン状況の確認 10-13
第 10 章 ネットワーク・クライアント 演習問題 10-15

Appendix - コマンド表 -

クレジット

このコース・テキストは有限会社ナレッジ・デザインのトレーニング・コースのマテリアルとして作成され、講師の説明、実習環境と共に使用することを前提としています。

有限会社ナレッジデザイン 〒182-0024 東京都調布市布田 4-6-1 調布丸善ビル TEL.0424-40-7912 FAX.0424-40-7913

イントロダクション

学習目標

「Linux サーバー構築」のコースの概要とコースの紹介。

Knowledge Design

ようこそ！！

「UNIX の基礎」コースへ！

コースの達成目標

UNIX, Linux とは何か？

管理者、オーナーになる前に・・・

UNIX オリエンテーション

UNIX と実習を通して対話する

ようこそ、「UNIX の基礎」コースへ！

本コースは、これから Linux の広大な世界に初めて入ろうとする方のためのガイドを目的としています。

コースは、インストラクターの講義と演習問題、実習から構成されており、受講者の理解と問題を解決する為のヒントを与える事ができるようになっています。疑問に感じた事はインストラクターと共に解決していきます。

本コースの目的

UNIX, Linux とは何かを理解する

UNIX, Linux の関係や、歴史的側面、特徴などを理解します。

管理者、オーナーになる前に知っておくべき最小限の事

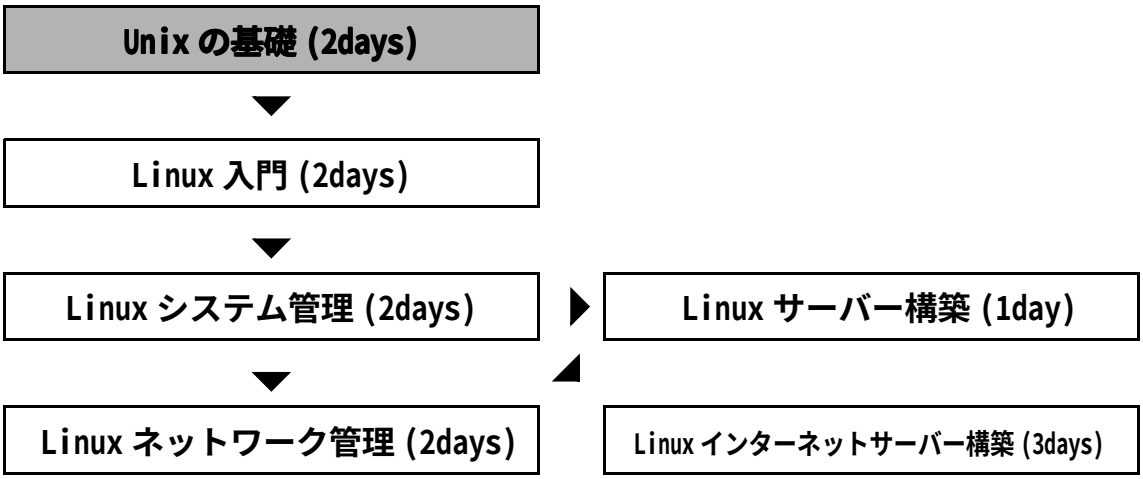
ログイン、ログアウトの方法や、ウィンドウシステムの利用の仕方、ファイルの操作方法、マニュアルの読み方など、UNIX, Linux を利用する際に、あらかじめ知っておくと便利な知識、技法についての説明をします。

UNIX オリエンテーション

実際に UNIX, Linux に触れ、演習問題を試し、実習する事で何かを発見し、理解を深めます。

MEMO インストラクターの電子メールアドレス：

コースフロー



ナレッジデザインのウェブサイト

http://www.knowd.co.jp

もし、もっとたくさんのことを知りたくなったら

ナレッジデザインでは、Linux に関するさらに詳細なコースを有償で提供しています。もし、更に理解を深めたくなったら、他のコースも合わせて受講してみてください。

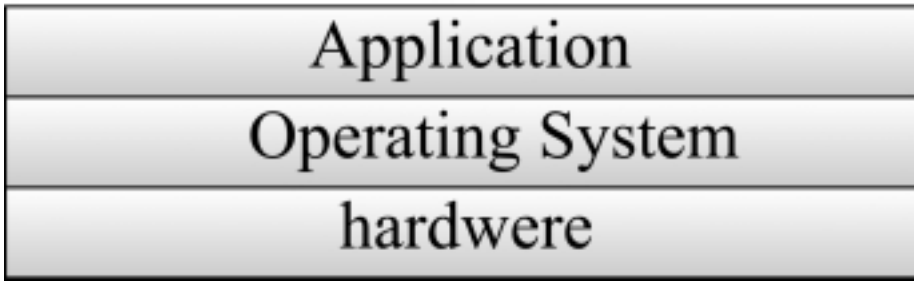
トレーニングに関する最新の情報は、ナレッジデザインのウェブサイトに掲載しています。
http://www.knowd.co.jp をご覧ください。

第１章 オペレーティング・システムについて

学習目標

オペレーティング・システムとは何かを考えます。

オペレーティング・システム



はじめに・・・

現在、私たちの生活の中で、たくさんのコンピューターが様々な目的で利用されています。コンピューターはどんなことに使われているかを考えてみます。

電子メールやウェブを使ったインターネットの利用

たくさんの計算を伴う会計、事務処理

文章や図形を考えたり、作成する

飛行機等のチケットを予約するするシステム

大量の計算処理が出る科学技術計算

ゲームを楽しむ

映像や、音声などの編集、作成作業を行う

電子商取引を行う

スケジュールを管理する

企業において顧客の管理を行う

無人で何かを監視させる

これ以外にも、まだたくさんの利用方法と可能性を秘めています。

コンピューターに上記の処理を行わせるときには、何がしらかのプログラムの存在があります。

インターネットにアクセスする為のブラウザ

顧客のデータを管理する為のデータ・ベース・システム

文章や図形を操作する為のワード・プロセッサやグラフィックス・ツール

表計算が行えるような会計ソフトウェア

学術的な計算をする為の関数を処理するプログラム

ネットワークが使えるようになるような通信プログラム

企業や利用者のニーズに沿って作られた、その他のプログラム

これらのプログラムは、例外なくハードウェア（コンピューターを構成する装置）を必要とし、それ自体、非常に複雑な仕組みを持ちます。又、そのハードウェアはたくさんの供給元を持ち、機能や性能も異なります。

オペレーティング・システムは、ハードウェアの機能をより効率よく利用し、多くの開発者が目的のプログラムを開発しやすいように、また、多くの利用者がコンピューターを簡単に使えるように開発された、基本的な機能を提供するソフト・ウェアです。

優れたオペレーティング・システムは、開発者の開発効率を上げ、利用者は少ない労力で、必要な機能を得ることができます。

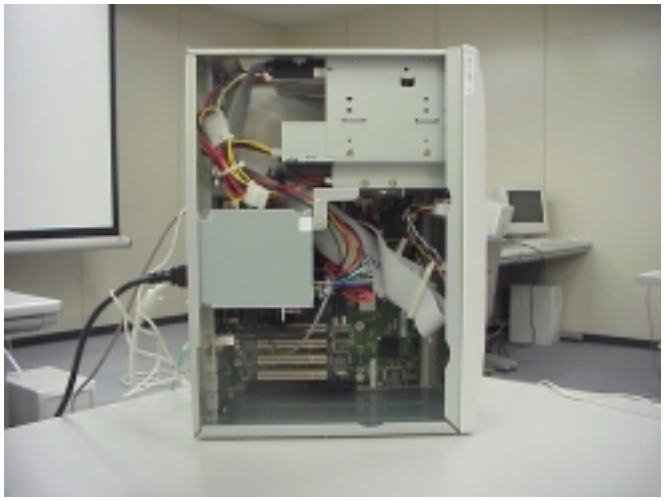
コンピューターを構成している装置

CPU
RAM
HDD
キーボード
ポインティング・デバイス
ディスプレイ
ビデオ・カード
FDD
CD-ROM ドライブ
ネットワーク・カード
モデム
サウンド・カード
マザー・ボード

pìå=jáÅêçëóëíÉãë社=räíê~=N

Sun Ultra 1 Creator 3D
CPU: Ultra Sparc 140MHz
HD: 2.1GB Ultra SCSI
Memory: 128MB
Solaris 7
Network: 100BASE-TX

d~óíï~ó=OMMM社 モデル dmJPSS



Gateway GP-366C
CPU: Intel 366MHz Celeron
HD: 6.4GB UATA
Memory: 128MB
VIDEO CARD: ATI RAGEIIc
Network: SMC EtherPower

コンピューターを構成している装置

汎用的なコンピューターは、ある機能を持った様々な電気的、機械的装置が相当数装備されています。これらはユーザーが利用できるハードウェア的な資源であると考えることができます。一般的なパーソナル・コンピューターや * ワークステーションに備わっている代表的な装置には次のものがあります。

* **ワークステーション** 一般的にパーソナル・コンピューターよりも高性能なエンジニア向けのコンピューターをさしますが、現在、パーソナル・コンピューターの性能の向上が著しく、その違いは徐々になくなりつつあります。

CPU（中央演算処理装置）	データの演算を行う装置。コンピューターで行われるほとんどの処理は、演算を伴うため、コンピューターの全体的な性能に影響します。。Intel 社、MIPS 社、Motorola 社、AMD 社、Cyrix 社等が供給を行っています。
RAM（ランダム・アクセス・メモリ）	データの記憶をする装置です。データの読み書きは高速で、電源の供給がなくなると、記録されていたデータは消去されます。
HDD（ハード・ディスク・ドライブ）	データーを記憶する装置です。RAM に比べると読み書きは低速ですが、電源の供給がなくなっても、データは恒常的に記録されます。Quantum Technology 社、Seagate 社、Maxtor 社、IBM 社、Western Digital 社等が供給を行っています。
キーボード	文字や数字、記号などをキーを押すことで、入力することのできる装置です。
ポインティング・デバイス	画面上のある点の座標を指し示し、入力することのできるような装置で、マウスやトラックボール、タブレット等があります。
ディスプレイ	文字データや、画像を表示する装置です。
ビデオ・カード	データを画像に変換しディスプレイへ信号を送るための装置です。性能の高いビデオカードは、高い解像度、たくさんの色を高速に描画することがでいます。ATI 社、S3 社、Matrox 社、Creative Media 社、Diamond 社等が供給を行っています。
FDD（フロッピー・ディスク・ドライブ）	フロッピー・ディスクに対し、データを読み書きできるような装置です。
CD-ROM ドライブ	CD-ROM からデータを読み込むことのできる装置です。
ネットワーク・カード	他のコンピューターや、他の装置とデータを相互に交換するための装置です。3Com 社、SMC 社、Corega 社等が、供給を行っています。
モデム	電話回線を使用した音声データを、コンピューターが扱えるようなデータに変換する、または、コンピューターのデータを音声に変換する装置です。
サウンド・カード	データを音声に変換し再生、録音するための装置です。
マザー・ボード	上記の装置を物理的に装着できるようにし、それぞれが互いに電気信号ののやり取りが行えるような基盤です。AsusTek 社、GigaByte 社、ABIT 社、Tekram 社等が供給を行っています。

* 選択したメーカーについて・・・ コンピューターシステム全体として市場に提供しているメーカーもたくさんありますし、その中に、上記のメーカの装置を含んでいる場合もあります。上記の会社は、単一の周辺機器として市場に提供している代表的なメーカーとしていくつか選びました。

OS の機能

ユーザー認証
個々のユーザーを識別するしくみ

データの管理
大量のデータの保存と、検索、操作

マルチタスク
複数の処理を同時期に行える機能

仮想記憶
主記憶装置の容量以上のプログラム、データを扱う

ネットワーク
他のコンピューターと通信できる機能

ウィンドウシステム
仮想的な複数のディスプレイ ,GUI による直感的な操作

プログラミング環境
新たなプログラムの作成を補助する環境

マルチメディア
音声、画像、動画などの情報を扱える能力

OS の機能

一般的な OS は次の機能のうちのいくつかを持っています。

ユーザー認証

OS が利用者を何らかの方法で識別します。一般的には、利用する前にユーザー ID とパスワードを入力させ、一致すれば、資源の利用が許可されるという方法を用います。これにより、OS はユーザーが他人のファイルや、その他の情報を不正に利用することを防ぐことが可能です。

データの管理

データを作成しファイルに保存することができます。データは通常 HDD に保存され必要なときに利用することができます。紙で保存するよりもスペースを取らず、また、大量のデータが保管でき、その中から特定のものを高速に検索することができます。リレーショナル・データベース・マネージメント・システム (RDBMS) は OS が持っていない更に強力なデータ管理のしくみを提供するシステムです。

マルチタスク

複数のプログラムを同時に処理できるしくみを提供します。例えば、あるプログラムは終了するまで、長い時間（たとえば１時間）かかる処理を行っていたとも、それが終わるまで待っている必要はなく、他の仕事を始めることができます。

仮想記憶

プログラムの実行や、処理されるデータはコンピューターの主記憶 (RAM) 上で行われます。ですから、通常は、コンピューターに搭載されている主記憶装置の容量以上のプログラム、または、データの処理ができません。近代的な OS は、主記憶の容量以上のプログラムやデータをハードディスクに一時的に待避させ、必要なときに呼び戻すという機能を持っています。これにより、非常にたくさんのプログラムの実行や、データの処理を行うことが可能になります。

ネットワーク

他のコンピューターや、他の通信機器と通信できる機能を提供します。他のコンピューターを遠隔操作、他のコンピューターのハードディスクやプリンタを利用する、また、他のコンピューター上に保存されている情報を転送することが可能になります。電子メールはネットワーク機能の応用例の一つです。一つの高性能のコンピューターを複数人で共有して使用したり、逆に、一つの大きな仕事をたくさんのコンピューターに分散させて処理することも可能になります。

ウィンドウシステム

一つのディスプレイに、仮想的なディスプレイを複数立ち上げ、たくさんの処理を行えるようにするシステムです。例えば、ある資料を表示しながら、別のウィンドウで、その資料をもとに別の資料を作成することができます。ウィンドウ間では、情報をやり取りすることができます。

プログラミング環境

独自のプログラムを作成するための開発の言語や補助プログラム、そしてそれを実行、評価するための環境が用意されている場合があります。

マルチメディア

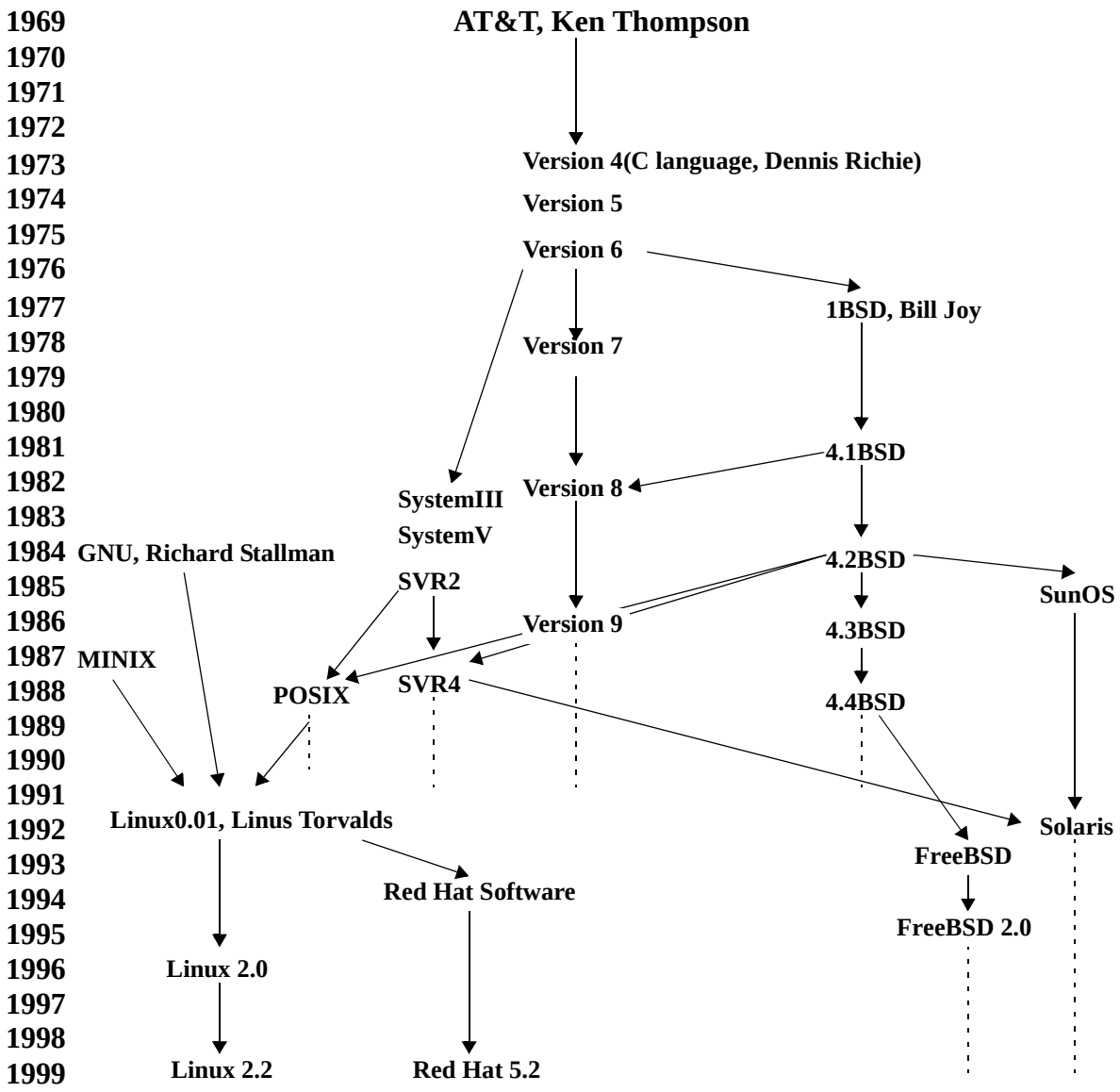
文字情報だけではなく、音声、画像、動画等の情報を扱うことができます。

第２章 UNIX のアーキテクチャ

学習目標

UNIX の構造や、アーキテクチャについて学びます。

UINX の歴史


1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
AT&T, Ken Thompson
Version 4(C language, Dennis Richie)
Version 5
Version 6
1BSD, Bill Joy
Version 7
4.1BSD
Version 8
SystemIII
SystemV
GNU, Richard Stallman
4.2BSD
SunOS
SVR2
Version 9
4.3BSD
MINIX
POSIX
SVR4
4.4BSD
Linux0.01, Linus Torvalds
Solaris
FreeBSD
Red Hat Software
FreeBSD 2.0
Linux 2.0
Linux 2.2
Red Hat 5.2

UNIX の歴史

Ken Thompson, Dennis Richie

1969 年、アメリカ AT&T、ベル研究所の Ken Thompson 氏が DEC 社の PDP-7 上で UNIX の原形に当たる OS を開発したのが始まりといわれています。1973 年、同 Dennis Richie 氏の協力により全面的に C 言語に書き直され移植が容易になったことから、様々なハードウェア上で稼動するようになっていきます。

BSD, Bill Joy

1977 年、カリフォルニア州立バークレー校に UNIX がわたり、Bill Joy 氏らの手で、仮想記憶やプロセス間通信、C シェル、vi 等の革新的な機能が盛り込まれることになり、以後、BSD(Berkeley Software Distribution) として一つの流れを形成し、その他の UNIX 全体に大きな影響を与えました。

SystemV

AT&T は主に UNIX を商用のオペレーティング・システムとして SystemIII を経て , 現在の SystemV 系の流れを形成しています。UNIX は移植性と、そのライセンス料の安さから、Sun Microsystems 社をはじめとする様々なベンダーが開発するハードウェアに移植され、UNIX にはたくさんのクローンが誕生していきます。この頃から AT&T は、UNIX のライセンスに対して規制を強めてきています。

POSIX

1988 年、UNIX の標準化の動きの一環として、IEEE (Institute of Electric and Electronic Engineers) と ANSI(American National Standards Institute) は UNIX 風のインターフェイスを定義する POSIX(Portable Operating System Interface for Computer Environments) を作成します。多くの UNIX ベンダーは UNIX の標準的な定義であるとして、これに同意し準拠した UNIX を作っています。POSIX は一般に公開され入手可能です。(http://www.ieee.org)

GNU, Richard Stallman

1984 年、MIT (マサチューセッツ工科大学 Massachusetts Institute of Technology) に在籍していた Richard Stallman 氏は GNU(GNU ' s not UNIX) というフリーソフトウェアの慈善団体を作り、そこで、ソフトウェアに関する考え方 (GNU Manifesto) と、配付条件 (GNU General Public License) を定め、様々な人がソフトウェアを配付する際のわかりやすい方針を出します。GPL はソフトウェアの複製を配付する際の指針として優れたソフトウェアの間で用いられるようになります。

Linux, Linus Torvalds

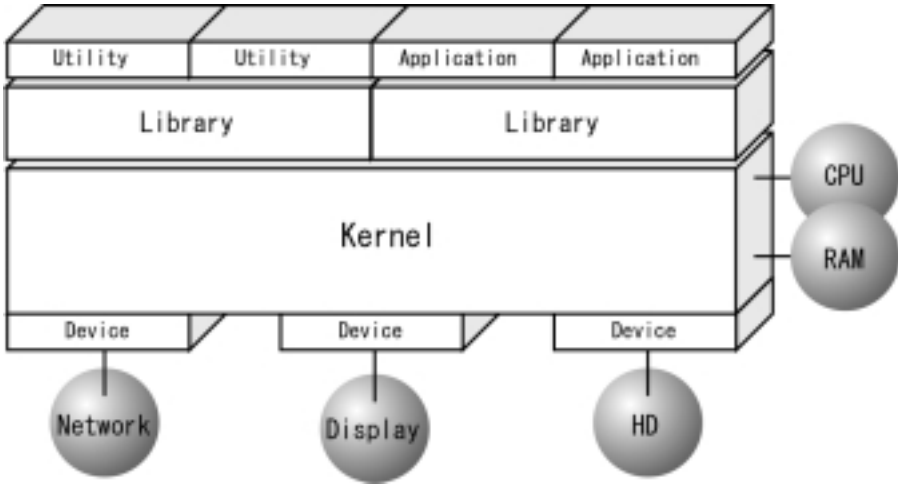
1991 年、フィンランドの Linus Torvalds 氏は、個人として UNIX のクローンを作成し始め、その旨を世界規模のニュースグループである USENET に投稿し、同時にそのソースコードをフィンランドにある FTP サーバーに公開しダウンロード可能な状態にしました。Linus 氏は UNIX ライクな OS を、市場で一般に入手しやすい PC/AT 互換機上で作成し、それを無料で配布しました。世界中の人たちは、それをただ使用するだけでなく、自分に必要なプログラムを追加したり、自分のニーズに合わない部分を変更したりして、時にはそれが、Linux の新しい部品として使われるという循環が起こり、非常に洗練された POSIX 互換の UNIX に仕上がっています。

Distributers, Red Hat Software

1994 年、Red Hat Software 社 (http://www.redhat.com) は、Linux に簡単なインストールの仕組みや、パッケージ管理の機能 (Redhat Package Manager) を加え、ネットワーク上から、または、CD-ROM にパッケージングし、より入手しやすい形態にしています。現在、たくさんのディストリビューターが Linux をサポートしています。

パッケージ化されたディストリビューション

UNIX の構成要素



カーネル

基本的な機能の提供、ハードウェアの管理

デバイス・ドライバ

周辺機器の制御

ライブラリ

プログラム開発のための素材となるルーチンの集合

ユーティリティ

よく使用される基本的なプログラム群

アプリケーション

OS が提供していない最終的なプログラムの形態

オペレーティング・システムの構成要素

オペレーティング・システムは次の役割 * を持ったソフトウェアです。

ハードウェア資源の制御と、効率的な活用

プログラム（アプリケーション）の効率的な開発を助けるための、基礎的なしくみの提供

利用者の作業を円滑に行うための基礎的な作業環境の提供

OS ではいくつかの役割を持ったプログラム群で全体が構成されるのが一般的です。

カーネル

OS のもっとも基本的な機能を提供するプログラム。つまり、CPU の利用、メモリの管理、周辺機器の管理、ファイルシステムの管理、ハードウェアの割り込み等、ハードウェアに関するあらゆることを管理しています。OS の起動は、まず、このカーネルがコンピューターシステムに読みこまれることから始まります。プログラマや利用者は、カーネルが提供しているサービスを利用することで、ハードウェア資源を活用することができるようになります。

デバイス・ドライバ

デバイス・ドライバは、周辺機器をカーネルが制御できるようにするためのプログラムです。周辺機器は様々なメーカーから提供されます。カーネルは、コンピューターに接続されうる全てのハードウェアに対応する仕組みを持っているわけではありません。ですから、新たな周辺機器をコンピューターに追加した場合、カーネルが制御できるようにデバイス・ドライバを OS に追加する必要があります。

ライブラリ

プログラムを開発する際の素材となるルーチンを集めたプログラムです。このことで、プログラマは、カーネル自身の提供する機能を一つ一つ呼び出すのではく、ライブラリによって提供されるある目的を持った処理を組み合わせて利用することができるようになります。同一のライブラリを使用して作成されたプログラムは、同一のライブラリを持つシステム間で同じように働くことになり、開発者間でのプログラミングの手間の重複を避け、作成するプログラムのサイズを小さくすることができます。優れたライブラリ群は、開発の効率を上げることができます。

お互い共通のライブラリを持つことのメリット

ユーティリティ

プログラムを開発したり、コンピューターを管理、操作するために、多くの人が一般的によく行うであろう基本的な作業を行うためのプログラムは、あらかじめ OS に付属してきます。たとえば、ファイルの内容をディスプレイに表示する、文章を作成する、システムの状態を監視する、プログラムを開発するといった作業は、コンピューターを使用する多くの人が行う作業です。このような作業を簡単に行えるためのプログラムがユーティリティです。ユーティリティにはシェルが含まれます。

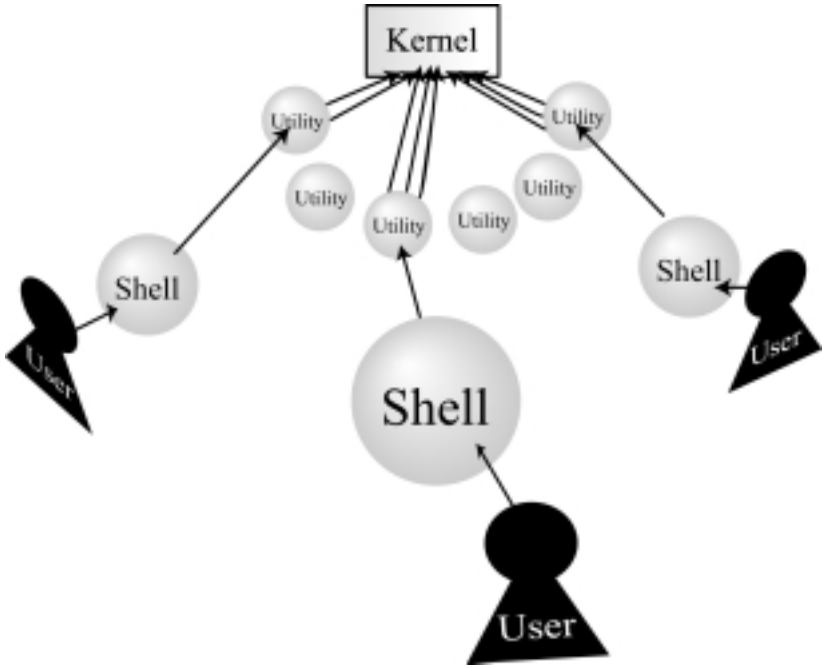
アプリケーション

OS の機能をもとに、利用者が最終的に望む分野に対し適応した OS が提供していないプログラム。アプリケーションは、一般的には、プログラムベンダーによりパッケージや販売されたり、受託開発、または自社開発などによって適切なプログラムとして利用されます。

* OS の役割について OS が提供する機能は、コンピューター・テクノロジーの成熟と共に変化（基本的には拡大方向に）しています。ユーティリティとアプリケーションの違いも同様で、状況により変化し、厳密に定義することは難しいと考えます。

シェル

ユーザーはシェルにより OS を容易に使用できる



UNIX 環境の利用の仕方 - コマンド・インタプリタ -

利用者が UNIX 提供する機能や、環境、ユーティリティを使用するときに、それらを簡単にする為の仕組みは次のようになります。シェルは UNIX システムに多数用意されていますし、開発することも可能です。

シェル （コマンド・インタプリタ）

ユーザーは、OS に用意されている、または、開発者が開発したプログラムを、シェルから名前を入力するだけで利用できます。シェルは、特別な機能を持つプログラム可能な、ユーティリティの１つであり、プログラムを実行しやすいよう、利用者毎に選択、カスタマイズ可能です。

第３章 ログイン・ログアウト

学習目標

UNIX システムを使用するためにはログインという作業を行わなくてはなりません。このことで、システムは利用者を識別できるようになります。又、UNIX システムの使用を終了する際は、ログアウトという作業を行います。シングル・ユーザー向けのオペレーティング・システムにはないこの作業の方法を学習します。

UNIX を使用するために必要な作業

ユーザー認証

ユーザー名をシステムに告げる
システムは利用者を識別する

システム管理者、スーパーユーザー、root

管理者、全権限を持つ特別なユーザー
システムの破壊も可能
システムの管理作業を行うユーザーが用いる

一般ユーザー

システム管理者によって作成される
限定された操作のみが行える
通常一般的な作業を行うユーザーが用いる

UNIX を使用するために必要な作業

システムを使用する際の身分証明

システムがユーザーが誰であるかを確認することを一般的に**「ユーザー認証」**といいます。

UNIX のユーザー認証は、**ユーザー名**と**パスワード**で行われ、利用者がパスワードを知っているかどうかで、本人であるか否かを判定します。

UNIX システムに対しユーザー名とパスワードを入力し、作業を開始しようとすることを、**「ログイン」**と呼び、ログインに成功しなければ、UNIX を使用することができません。ログインに成功した後、実際に作業している状態は、**「ログイン中」**であるといわれます。

このように、ユーザー名とパスワードを入力しなければ使用できないしくみは、UNIX システムの**機密保護機能（セキュリティ）**のもっとも基本的な仕組みです。

ユーザー名は誰が設定するのか？ システム管理者と一般ユーザーについて

UNIX システムを使う人の立場は「システム管理者」と「一般ユーザー」の２種類あり、それぞれの役割が異なります。**一般ユーザーはシステム管理者によってあらかじめ登録されていなければなりません**。

システム管理者、スーパーユーザー、root

UNIX システムの設定に関する全責任を持ち、かつ、あらゆる操作の全権限を持つユーザーです。ユーザー名は、慣例的に **root（ルート）**という名前が使われます。このシステム管理者が、UNIX システムが使用できるユーザーを決めます。全権限を持つがゆえに、システムを動作不能の状態することも可能です。

一般ユーザー（システム管理者以外のユーザー）

システム管理者によって事前に登録された、システム管理者以外のユーザーを指します。一般ユーザーはシステムにおいて、**限定された操作**しか行えません。システム管理以外の作業を行う際、一般ユーザーで作業するのは安全な方法です。

UNIX システムを管理する人間が、使用する人間と異なっている場合は、システム管理を担当している人に、ユーザーを作成してもらわなければ、ログインしてシステムを使うことはできません。

ログイン・プロンプト

login: ユーザー名
Password: パスワード

！ユーザー名とパスワードは事前にシステム管理者から得ておく
！パスワードは表示されない
！大文字と小文字は区別される
！テンキーで数字を入力するときは [NumLock] キーを押す

ログインの失敗

Login incorrect

login:

再挑戦！

ログインの成功

Last login: Mon Apr 12 17:40:44 from kdcgw2
[yyamazaki@linux11 yyamazaki]$

コマンド・プロンプトが表示される

ログイン・プロンプト

UNIX システムは利用者の利用を待機している際、ログイン・プロンプトを画面に表示しています。ログイン・プロンプトからは、ユーザー名に続いてパスワードを入力します。

ユーザー名とパスワードを入力する

最も初期の状態において、ユーザー名はシステム管理者が設定したものになっています。自分のユーザー名とパスワードがなんであるかは、システム管理者に尋ねます。

"login: " の表示の前に、表示されるのはホスト名（システムの名前）です。

```
login: yyamazaki
Password:
```

ユーザー名を入力すると、パスワードの入力を促されます。正しいパスワード入力してください。入力された**パスワードはエコーバック（画面への表示）されません**。

不正なユーザー名、パスワード

入力されたパスワードが正しくなければ、"**Login incorrect**" という表示が出て、数秒経過した後、再度、ログインプロンプトが現れますので、**再度、正しいユーザー名とパスワードの入力を試みます。**

```
login: yyamazaki
Password:
Login incorrect

login:
```

ユーザー名、パスワードを入力する際、以下のことに気を付けます。

アルファベットの大文字、小文字は区別されます。

数字をテンキーを使用して入力する際は、[NumLock] キーを有効にしておきます。

ログインの成功

ユーザー名とパスワードが正しいものであれば、システムはログインを許可します。

```
login: yyamazaki
Password:
Last login: Mon Apr 12 17:40:44 from kdcgw2
[yyamazaki@linux11 yyamazaki]$
```

画面の表示は、あなたが以前ログインした日時と場所（コンピュータ名）を表示し、コマンド・プロンプトを表示します。これで UNIX システムを使うことができるようになりました。

コマンド・プロンプト

システムに対しコマンドを入力するための行を表す

プロンプトの形

#	システム管理者
$ （又は %）	一般ユーザ

一般ユーザーのデフォルトのプロンプト

[ユーザー名@ホスト名 カレント・ディレクトリ] $

ex.
[yyamazaki@linux11 yyamazaki] $

ユーザー yyamazaki が、linux11 というシステム上で、yyamazaki というディレクトリで作業している

コマンド・プロンプト

ログインに成功すると、UNIX システムを使用することができるようになります。UNIX システムの使用方法は２種類あります。これらは、完全にお互いを補完し合うものではありません。

CUI(Character User Interface)　文字を用いた表現方法で操作する。キーボードからコマンドを入力する。UNIX の操作は CUI だけでもほとんどの操作を網羅できる。ユーザーは、前もって、入力できるコマンド群のボキャブラリを持つ必要がある。

GUI(Graphical User Interface)　グラフィックスを用いて操作する。画面のグラフィックスをマウスで操作しながら、作業を行う。ウィンドウ・システム（後述）を用いて実現されている。一般的には、メニュー形式や操作を導くようなシンボルが用いられ、あらかじめ知っておく操作方法が少なくてすむ。

コマンド・プロンプトは CUI に関するコマンド入力を促しているものです。
コマンド・プロンプトからは、UNIX コマンドが入力できます。

コマンド・プロンプトの形

コマンド・プロンプトは利用者が自由に変更（後述）することができますが、一般的な慣例から次のようなシンボルが用いられます。

特徴	内容	サンプル
最後の文字が #	システム管理者	linux11#
最後の文字が $	B シェル系のシェルを使用している一般ユーザー	[linux11@yyamazai home]$
最後の文字が %	C シェル系のシェルを使用している一般ユーザー	linux11%

* これらは制約ではなく、よく用いられている慣例です。

デフォルトのコマンド・プロンプトの形

コマンド・プロンプトには様々な情報を持たせることができます。デフォルト（利用者が、何も変更をしなかったときの状態、初期状態）でのコマンド・プロンプトの意味は次のようになっています。

* この限りではない UNIX システムも多数存在します

[yyamazaki@linux11 yyamazaki]$

[ユーザー名 @ ホスト名 カレントディレクトリ名] $

ユーザー名　現在ログインしているユーザーの名前、つまり、自分自身の名前です。
ホスト名　現在使用しているコンピューター・システムの名前です。
カレント・ディレクトリ　現在作業しているファイルシステム上（後述）でのディレクトリ名です。

パスワードの変更

```
$ passwd
```

パスワードのデフォルトの規則

６文字以上でなくてはいけない

５種類以上の文字でなくてはいけない

大文字、小文字は区別される

ユーザー名を元に構成しない

辞書に出てくるような一単語だけで構成しない

以前のパスワードと類似させない

システム（定型）的すぎない、又は単純すぎない

その他・・・

```
$ passwd
Changing password for yyamazaki
(current) UNIX password: 現在のパスワード
New UNIX password: 新しいパスワード
BAD PASSWORD: it’s WAY too short エラー！
New UNIX password: 再設定
```

パスワードの変更

passwd コマンド

パスワードを変更するには、passwd コマンドを使用します。コマンド・プロンプトから次のように入力します。

```
$ passwd
```

パスワードを変更できるのは、**システム管理者と自分だけ**です。

＊コマンドを入力するときは小文字で入力してください。

passwd コマンドを使用してパスワードを変更する

passwd コマンドを実行すると、古いパスワードを聞いてきます。古いパスワードを入力した後、新しいパスワードを**2回タイプ**します。

```
$ passwd
Changing password for yyamazaki
(current) UNIX password:
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully
$
```

パスワードの規則、及び設定時のエラー

パスワードを設定する際、システムによっては、安易なパスワードをユーザーが設定することで、セキュリティ・レベルの低下を防ぐために、次のような規則がある場合があります。規則に違反した場合は、次のエラー・メッセージが出てきますので、**再度設定し直してください**。

メッセージ	内容	エラーのサンプル
BAD PASSWORD: it's WAY too short BAD PASSWORD: it is too short	５文字以下で短すぎる	unix linux
BAD PASSWORD: it does not contain enough DIFFERENT characters	４種類以下の文字しか使っていない	aaaaaa abcdabcd
BAD PASSWORD: it is based on a dictionary word	辞書の一単語だけをベースに構成されている	99summer student00
BAD PASSWORD: it is based on your username	ユーザー名だけをベースに構成されている	yamazaki71 (yamazaki がユーザー名の場合)
BAD PASSWORD: is too similiar to the old one	以前のパスワードと類似している	旧：hellounix 新 :hellolinux
BAD PASSWORD: it is too simplis-tic/systematic	単純、またはシステム的である	123456789 ABCDEFGHIJKLMN
BAD PASSWORD: is too simple	単純すぎる	ASDFGH（キーボードの下の配列を左から入力）
Sorry, passwords do not match	１回目と２回目の入力が異なった	1st: unixisfun 2nd: unixisfan

＊その他のメッセージが出てきたら、指示を読みそれにしたがって、設定し直して下さい。

ログアウト

logout コマンド
exit コマンド
[Ctrl] + d

UNIX システム使用の終了
ログインする前の状態（ログイン・プロンプトの表示）に戻る
セキュリティ

主電源のオフ

必ずシャットダウン作業を行ってから

ログアウト

UNIX システムの使用終了は、ログアウトによって行われます。ログアウトは logout コマンド、exit コマンド、または、^D （Ctrl キーと d を同時に押す）ことで行えます。ログアウトせず、ログ・インしたままの状態は、自分以外のユーザーが自分のユーザー名で操作することを許してしまうことになります。誤操作、セキュリティのことを考慮し、**使用がすんだらログアウトするようにします**。

$ logout

又は

$ exit

又は

$ ^D

ログアウトすると、ログイン・プロンプトが現われ、他のユーザーで UNIX システムを使用することができるようになります。

電源のオフについて

UNIX システムでは、コンピューターの**主電源をオフにする前に、システムの保護のためにある手順を行う必要があります**。また、UNIX では、ネットワークを使用して遠隔（リモート）で、同じ UNIX システムを使用している場合もあります。

主電源をオフにする前に UNIX システムを停止させる手順をシャットダウン作業が必要です。もし、この作業が行われないうちに、電源を切ると、システムが破壊される危険があります。**シャットダウン作業なしに、電源は切らないようにしましょう。**

主電源のオフ

方法１

システム管理者としてログインする

shutdown -h now　を実行する

その後、電源をオフにする

ログアウト

方法２

[Ctrl] + [Alt] + [Delete] を同時に押す
システム終了後、再起動前に、電源をオフにする

主電源のオフ

主電源をオフにするためには、次の２つの手順のどちらかを用いることができます。

シャットダウン作業を行う その１ - shutdown コマンド -

STEP 1) システム管理者としてログインする

システム管理者のユーザー名は、慣例的に root（ルート）が使われます。root はそのシステムの全権限を持ったユーザーです。root のパスワードを知っているユーザーはシステムにおける全権限を持つことになります。

root の初期のパスワードは、UNIX システムをインストールする過程で決定されます。その後、インストールした人が他人にパスワードを教えると、その人はシステム中で、全権限を持ったユーザーになることができます。root のパスワードは、必要最低限の人にだけ伝えましょう。

```
TurboLinux release 3.0 (Karatu)
Kernel 2.0.35 on an i686 (linux11.sr2.knowd.co.jp)
TTY: ttyp2

login: root
Password:
Last login: Mon Apr 12 17:32:54 on tty1
You have new mail.
[root@linux11 /root]#
```

STEP 2) shutdown コマンドを実行する

シャットダウン作業は、shutdown コマンドを用います。このコマンドは、**root しか実行できません**。

以下のコマンドを実行すると、待ち時間なしに即座にシャットダウンを行います。

```
# shutdown -h now
...
...
system halted
```

シャットダウン手順のためのたくさんのメッセージが出た後、**"system halted" というメッセージが出たら、コンピューターの電源をオフにします**。

シャットダウン作業を行う その２ - [Ctrl] + [Alt] + [Delete] キー -

どのようなユーザーであっても、また、たとえログインしていなくても、**[Ctrl] キーと [Alt] キーと [Delete] キーの３つのキーを同時に押すと**、いつでもシステムをシャットダウンできます。

システムがシャットダウンした後、自動的に再起動されます。このような状態を **reboot（リブート）**するといいます。もし、電源をオフにしたければ、UNIX システムが立ちあがる前に電源を落としてください。

この方法は、ネットワークを通じたリモート操作（遠隔操作）では行えません。その UNIX システムのキーボードからのみ行えます。

＊一般的な PC Linux は上記の方法が使えますが、他の UNIX システムでは種類によって異なります

NOTE:
シャットダウン作業は、現在行われている作業を全て強制的に中断し、コンピューターの電源を落としてもよい状態にします。ネットワークを通じ複数人でシステムを共有している際は、**他の利用者が使用していないか、又は、シャットダウンすることを事前に通知するような配慮が必要になります**。

第３章 ログイン・ログアウト 演習問題

1） システムにログインしてください。（すでにログインしている場合は次へ）[実習]

2） ユーザーが UNIX システムを使用する際に、あらかじめ知っておかなければいけないことは何ですか？

_____

3） UNIX システムにおいて、全特権を持つユーザーはなんというユーザー名ですか？

_____

4） 自分のユーザー名に設定されたパスワードを別のものに変更してください。[実習]

_____

5） システムからログアウトしてください [実習]

6） システムをシャットダウンし、コンピューターの電源をオフにしてください。電源のスウィッチの場所はインストラクターから聞いてください。[実習]

7） 電源をオンにしてください。[実習]

8） ログイン・プロンプトが表示されたのを確認し、再度、一般ユーザーでログインしてください。[実習]

第４章 UNIX コマンド

学習目標

UNIX で使用でできる、簡単なコマンドのいくつかを紹介し、実際に使用してみます。ここで紹介するコマンドは、一般的な UNIX システムでほぼ同様に使える基本的なものばかりです。

コマンドライン

$ コマンド 引数１ 引数２・・・

引数とはコマンドに対するパラメーター

引数の指定の仕方はコマンドごとに異なる

引数を指定しなくてもよいコマンドもある

コマンド、各引数間の区切りは１つ以上のスペース

UNIX コマンド、ユーティリティ

UNIX では、多くの人がよく使用するであろう、便利な数百個のコマンドが標準的に用意されています。ここでは、簡単に使えるコマンドを紹介します。

コマンドライン

UNIX システムには、文字で**コマンド名を入力して実行**するタイプ (CUI) のオペレーションと、**ウィンドウシステムを用い、マウスで操作するタイプ** (GUI) のオペレーションの２つの操作が提供されます。基本的な操作のほとんどが、CUI で行えるというのは、UNIX の特徴の一つです。

コマンドはコマンド・プロンプトで始まる行に入力し、改行することで実行されます。この行のことをコマンド・ラインと呼びます。コマンド・ラインには次のような入力をします。

$ コマンド 引数１ 引数２ ・・・・ (この行全体がコマンドライン)

引数というのは、コマンドが要求するパラメーターのことで、どのような引数が必要かは、**全てのコマンドごとに異なります**。又、引数をとらないコマンドも多数存在します。**コマンドと各引数は１つ以上のスペースによって区切ります。**

引数を指定しなくても働くコマンドの例

```
passwd, logout, exit
```

引数を指定するコマンドの例

```
shutdown -h now ( -h, now という２つの引数が指定されている)
```

コマンドの例

date
現在の時刻の表示

```
$ date
Tue Apr 13 19:14:55 JST 1999

$ date -u
Tue Apr 13 10:15:33 UTC 1999
```

man
オンラインマニュアルの表示と検索

```
$ man date
DATE(1)                                                    DATE(1)

NAME
       date - print or set the system date and time

SYNOPSIS
       date  [-u] [-d datestr] [-s datestr] [--utc] [--universal]
       [--date=datestr]  [--set=datestr]   [--help]   [--version]
       [+FORMAT] [MMDDhhmm[[CC]YY][.ss]]
....

$ man -k linux
xgammon -- an X11 backgammon for linux
bootparam (7)         - Introduction to boot time parameters of the
Linux kernel
cdplay -- an interactive text-mode program for controlling and play-
ing audio CD
Roms under Linux.
cfdisk (8)          - Curses based disk partition table manipulator
```

マニュアルのページ操作

[スペース]	一画面進める
[b]	一画面もどる
[q]	終了
/ キーワド	検索
[n][N]	検索の繰り返し（順方向、逆方向）

いくつかの簡単なコマンドの使用

date

概略　現在の時間を表示します。システム管理者だけはシステムの時間を変更することもできます。

```
$ date
Tue Apr 13 17:15:52 JST 1999
```

引数　-u　　協定世界時間 (UTC)、世界標準時で表示します。

例）世界標準時を調べる

```
$ date -u
Tue Apr 13 08:22:41 UTC 1999
$
```

man

概略　オンライン・マニュアルを表示します。

引数　コマンド名　　調べたいコマンドの名前を指定します。

例）date コマンドについてのオンライン・マニュアルを調べる

```
$ man date

DATE(1)
DATE(1)

NAME
       date - print or set the system date and time

SYNOPSIS
       date  [-u] [-d datestr] [-s datestr] [--utc] [--universal]
       [--date=datestr]  [--set=datestr]   [--help]   [--version]
       [+FORMAT] [MMDDhhmm[[CC]YY][.ss]]

DESCRIPTION
       This  documentation  is no longer being maintained and may
       ...
```

オンライン・マニュアルのページの操作方法

マニュアルを表示したときに、ページが一画面に収まりきれないものがあると、1 ページごとに区切って表示されます。ページを操作するには次のようにします。

[スペース]	一画面進みます。
[b]	一画面戻ります。
[q]	マニュアルの表示を終了します。
/ キーワード	ページ内から指定したキーワードを検索し、そこに表示を切り替えます。続けて [n」をタイプすると、同じキーワードを繰り返し検索します。[N] は逆方向（ページの先頭方向）に向かって検索をします。

-k キーワード　キーワードによるマニュアル・データベースの検索を行います。キーワードは大文字小文字は関係ありません。コマンド名とその概要が表示されます。

例）linux というキーワードで、マニュアル・データベースを調べる

```
$ man -k linux

xgammon -- an X11 backgammon for linux
bootparam (7) - Introduction to boot time parameters of the Linux kernel
cfdisk (8) - Curses based disk partition table manipulator for Linux
....
```

コマンドの例

cal
カレンダーの表示

$ cal

```
     April 1999
Su Mo Tu We Th Fr Sa
                1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30
```

$ cal 2001

```
                              2001

       January               February                March
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6                  1  2  3                  1  2  3
 7  8  9 10 11 12 13    4  5  6  7  8  9 10    4  5  6  7  8  9 10
14 15 16 17 18 19 20   11 12 13 14 15 16 17   11 12 13 14 15 16 17
21 22 23 24 25 26 27   18 19 20 21 22 23 24   18 19 20 21 22 23 24
28 29 30 31            25 26 27 28            25 26 27 28 29 30 31

        April                  May                   June
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7           1  2  3  4  5                  1  2
 8  9 10 11 12 13 14    6  7  8  9 10 11 12    3  4  5  6  7  8  9
15 16 17 18 19 20 21   13 14 15 16 17 18 19   10 11 12 13 14 15 16
22 23 24 25 26 27 28   20 21 22 23 24 25 26   17 18 19 20 21 22 23
29 30                  27 28 29 30 31         24 25 26 27 28 29 30
```

$ cal 1 2001

```
    January 2001
Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
```

画面に収まらない表示の制御

コマンドの末尾に | less を指定する

$ cal 2001 | less

操作方法は man と同じ

cal

概要　**カレンダーを表示します。引数が指定されないと、今月のカレンダーが表示されます。**

```
$ cal
     April 1999
Su Mo Tu We Th Fr Sa
              1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30
```

引数　**西暦**　**西暦を指定すると、その年の１月から１２月までのカレンダーが表示されます。**

例）西暦２０００年のカレンダーを表示します。

```
$ cal 2000
                                       2000

       January               February                March
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
                    1          1  2  3  4  5             1  2  3  4
 2  3  4  5  6  7  8    6  7  8  9 10 11 12    5  6  7  8  9 10 11
 9 10 11 12 13 14 15   13 14 15 16 17 18 19   12 13 14 15 16 17 18
16 17 18 19 20 21 22   20 21 22 23 24 25 26   19 20 21 22 23 24 25
23 24 25 26 27 28 29   27 28 29               26 27 28 29 30 31
30 31
...
```

月 西暦　**指定した、年、月のカレンダーを表示します。**

例）西暦２００１年の１月のカレンダーを表示します。

```
$ cal 1 2001

    January 2001
Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
```

一画面に収まりきれない表示を操作する

cal で年単位での出力をしたときのように、１画面に表示が収まりきれないときは、less というコマンドが便利です。次のように cal コマンドと組み合わせて使います。操作方法は、man のページ操作とまったく同じです。| は**パイプ**と呼ばれる記号です。cal コマンドに限らず、様々なコマンドにも応用できます。

```
$ cal 2001 | less
```

[スペース]　一画面進みます。

[b]　一画面戻ります。

[q]　マニュアルの表示を終了します。

/ キーワード　ページ内から指定したキーワードを検索し、そこに表示を切り替えます。続けて [n」をタイプすると、同じキーワードを繰り返し検索します。[N] は逆方向（ページの先頭方向）に向かって検索をします。

コマンドの例

last
過去にログインした人の履歴

$ last

```
yyamazak tty1      Mon Apr 12 17:36 - 01:52  (08:15)
```

名前、使用端末、リモートホスト名、ログイン時刻、ログアウト時刻、利用合計時間

kon
日本語の表示

VGA ディスプレイで日本語を表示する
終了は exit

$ kon

```
KON2 Kanji On Console ver.0.3.8 (98/03/09) using VT number 1
                                        Copyright (C) 1993-1996  Takashi MANABE
                                                        1993, 1994 MAEDA Atusi
```

$ man kon

```
KON(1)                                                  KON(1)

【名前】
      KON - 漢字コンソールエミュレータ

【書式】
      kon [ video ] [ -capability value ...  ]

【概要】
      KON は Linux, FreeBSD のコンソール画面上で漢字を表示するた
      めのプログラムです。KON では、pty(4) を使ってコンソール入
      出力をフックし、VGA あるいは DCGA に展開する事によって漢字
      表示を実現しています。また、KON が動作しているコンソールか
      ら起動された KON は新しい仮想コンソール上で動作します。
```

$ exit

last

概要 UNIX システムに過去にログインした人の履歴を表示します。

```
$ last

yyamazak tty1                        Tue Apr 13 17:11   still logged in
yyamazak ttyp0       kdcgw2          Tue Apr 13 04:02   still logged in
root     ttyp0       kdcgw2          Tue Apr 13 04:01 - 04:01  (00:00)
yyamazak ttyp0       kdcgw2          Mon Apr 12 17:40 - 17:40  (00:00)
yyamazak ttyp0       kdcgw2          Mon Apr 12 17:37 - 17:40  (00:02)
....
```

出力内容

ユーザー名、使用端末名、リモート端末名、ログイン時刻、ログアウト時刻、合計ログイン時間

引数 -n 表示件数　　最新、何件の履歴を表示するかを指定できます。

例）最新３件の履歴を表示します。

```
$ last -n 3
root     tty1              Tue Apr 13 20:46   still logged in
yyamazak ttyp1  kdcgw2     Tue Apr 13 17:11   still logged in
yyamazak ttyp0  kdcgw2     Tue Apr 13 04:02   still logged in
```

kon

概要 VGA のディスプレイで、日本語表示できるようにします。このコマンドは Linux や FreeBSD できます。kon を終了してもとの表示に戻したいときは、コマンドラインから、exit と入力してください。

```
$ kon
KON2 Kanji On Console ver.0.3.8 (98/03/09) using VT number 1
                                    Copyright (C) 1993-1996  Takashi MANABE
                                                   1993, 1994 MAEDA Atusi
$ man kon

KON(1)                                                  KON(1)

【名前】
      KON - 漢字コンソールエミュレータ

【書式】
      kon [ video ] [ -capability value ...  ]

【概要】
      KON は Linux, FreeBSD のコンソール画面上で漢字を表示するた
      めのプログラムです。KON では、pty(4) を使ってコンソール入
      出力をフックし、VGA あるいは DCGA に展開する事によって漢字
      表示を実現しています。また、KON が動作しているコンソールか
      ら起動された KON は新しい仮想コンソール上で動作します。
...

$ exit
$
```

kon は以下の方々によって作成されました。（オンラインマニュアルに掲載）
MANABE Takashi 氏、MAEDA Atusi 氏、KOMEDA Shinji 氏、YOSHIDA Kensyu 氏

また、日本語のオンライン・マニュアルも多くのボランティアの方々が作成しています。

コマンドの例

clear
画面をクリアする

```
$ clear
```

echo
引数を画面に表示する

```
$ echo Knowledge Design Corporation
Knowledge Design Corporation
```

sleep
指定時間処理を休止する

```
$ sleelp 10
10 秒間待機する
```

[Ctrl] + c でコマンドの強制中止

clear

概要　**画面をクリアします。引数はありません。**

```
$ clear
```

echo

概要　**引数で指定した文字を画面に表示します。**

```
$ echo Hello World!
Hello World!
$
```

引数　**表示したい文字**　**指定した文字が、次の行に表示されます。**

```
$ echo Knowledge Design Corporation
Knowledge Design Corporation
```

-e　次の特殊文字が使えるようになります**。指定した文字は '(シングルクォーテーション）で囲みます**。

```
\a      ベルを鳴らします
\b      バックスペース
\n      改行
\t      タブ
\\      バックスラッシュ
```

\ は日本語環境では、\ で表示されますが、表示も入力も意味は同一です。

例）文字を表示した後にベルを鳴らす

```
$ echo -e 'ring ! \nring !\nring \a'
ring !
ring !
ring !
ピ！
```

sleep

概要　**引数で指定した時間、待機状態になります。**

例）１０秒間何もしません。

```
$ sleep 10
```

引数　**待機時間**　**数値を指定します。通常は秒数ですが以下の単位を指定することができます。**

h　**時間を表わします**

d　**日にちを表わします。**

例）２日間待機

```
$ sleep 2d
^C
$
```

コマンドを強制的に中止する方法

実行したプログラムの終了を待たず、強制的に終了させたいときは **[Ctrl] + c** を押します。

コマンドの例

bc
対話的な計算機

```
$ bc
25 * 5
125
( 8 + 13 ) * 2
42
^d
$
```

小数を含む計算

```
$ bc -l
10 / 3
3.33333333333333333333
```

whoami
自分が誰か調べる

```
$ whoami
yyamazaki
```

bc

概要　対話型のプログラム可能な計算機

例）簡単な計算を行います。

```
$ bc
25+24              éÆÇšì¸óÕ
49                 ìöÇ¶
( 5 + 7 ) * 3      åJÇËï‘ÇµéÆÇšì¸óÕ
36                 ìöÇ¶
quit               終了
$                  ÉRÉ}ÉìÉhÉvÉçÉìÉvÉgÇ…ïúãA
```

使用できる演算子

+, -, *, /	加減乗除
%	剰余算（余りを求める）

例）10 を 3 で割ったときの余りを求める

```
10 % 3
1
```

^	べき乗（累乗）

例）２の２５乗を求める

```
2 ^ 25
33554432
```

引数　-l　小数を扱うときに指定します。小数点以下２０桁まで表示します。小数点以下の表示桁数を変えたいときは、**scale=n** で指定できます。

例）小数点を含む計算を行います

```
$ bc -l
13 / 7
1.85714285714285714285
scale=99
19 / 7
2.714285714285714285714285714285714285714285714285714285714285714285714285714285
71
```

whoami

概要　自分のユーザー名を表示します。自分がなんというユーザーでログインしているか調べることが出来ます。

```
$ whoami
yyamazaki
```

コマンド・ラインの便利な使い方

マルチ・ステートメント

；で区切り、複数のコマンドを１行に書くことができる

```
$ command1; command2; command3
```

コマンドの再実行と編集

↑（又は [Ctrl]+p)	一つ前
↓（又は [Ctrl]+n)	一つ後
←（又は [Ctrl]+b)	左に移動
→（又は [Ctrl]+f)	右に移動
[Backspace][Delete]	一文字削除
[Ctrl] + r キーワード	コマンドラインを検索

コマンド・ラインの便利な使い方

マルチ・ステートメント

コマンドを ;（セミコロン）で区切ると、**１行のコマンド・ラインに複数のコマンドを記述**することができます。

例）date コマンドと cal コマンドを１行に記述する。

```
$ date; cal
Wed Apr 14 16:42:58 JST 1999
     April 1999
Su Mo Tu We Th Fr Sa
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30
```

例）画面をクリアした後、WELCOME という文字を表示する。

```
$ clear; echo WELCOME
```

例）１０秒後にログアウトする

```
$ sleep 10; logout
```

過去に実行したコマンドの再実行

以前実行したことあるコマンドを再度実行するときは、再び同じ文字をタイプしなくても、カーソル・キーの上下を押すと、古いコマンドラインを呼び出すことができます。また、コマンドラインの一部を編集して再実行したり、古いコマンドを検索することもでき、コマンド入力の手間を軽減してくれます。

↑（又は [Ctrl]+p)一つ前に実行されたコマンドラインの呼び出し

↓（又は [Ctrl]+n)一つ後に実行されたコマンドラインの呼び出し

←（又は [Ctrl]+b)コマンド・ラインのカーソルを左に移動させます。

→（又は [Ctrl]+f)コマンド・ラインのカーソルを右に移動させます。

[Backspace][Delete]一文字削除します。

[Ctrl] + r キーワードキーワードで指定した文字を含む最新のコマンドラインを検索し表示します。

history n過去に実行したコマンドの履歴を表示します。n は最近実行したコマンド、n 個だけを表示するようにします。n は省略可能です。

例）過去に実行した who という文字を含むコマンド・ラインを再現します。

```
$[Ctrl] + r
(reverse-i-search)‘’:who
(reverse-i-search)‘who’:man whoami
```

この機能は、bash というシェルのヒストリー機能とファイル名補完機能を用いて実現されています。
この機能がないシェルもあります。Linux ではこの bash がよく用いられているようです。

第 4 章 UNIX コマンド 演習問題

1） コマンドに細かな指示を与える記述をなんといいますか？ 又、それは必須ですか？

2） 今何時ですか？ [実習]

3） あなたは何曜日に生まれましたか？ [実習]

4） 実行したコマンドが、一画面に収まりません。どのように対処できますか？

5） kon を使って、日本語を表示できるようにし、date マンドのマニュアルを日本語で表示してください。[実習]

6） あなた以外の人が過去にそのシステムを使っていますか？ [実習]

7） （ 12 + 8 ）* （ 97 - 43 ） / 7 の答えを小数点以下２０桁の精度で求めてください。[実習]

8） あなたは誰ですか？（何という名前でシステムにログインしていますか？）確認してください。[実習]

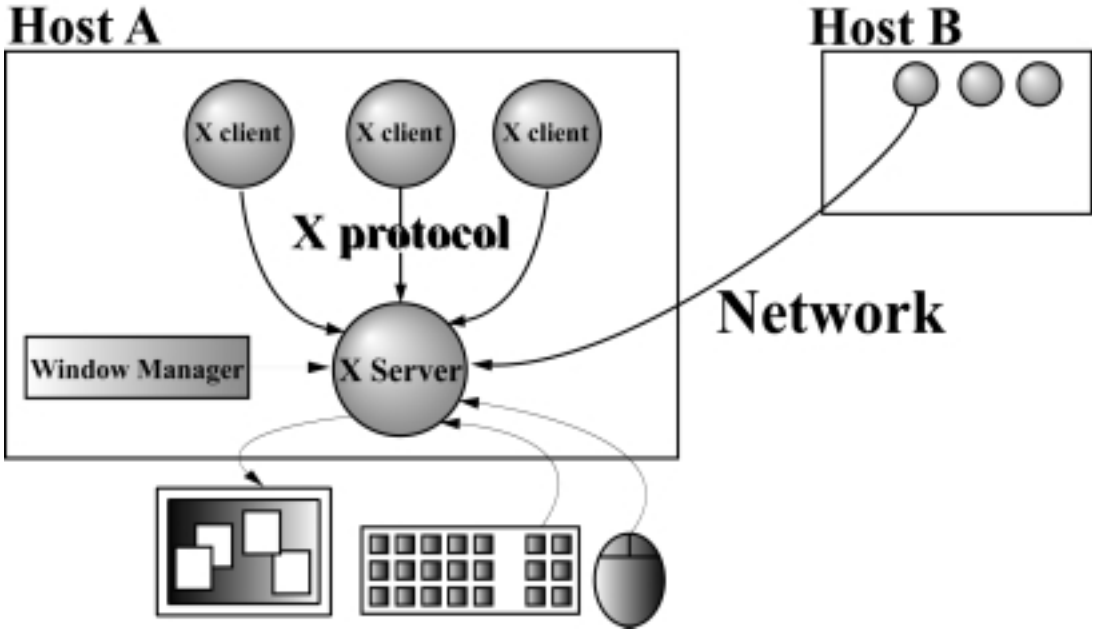
１０） ３秒後に画面の文字を一掃し、ベルを鳴らし、ログアウトするためのコマンドラインを実行してください。[実習]

第5章 ウィンドウ・システム

学習目標

ウィンドウ・システムとコマンド・ラインのオペレーションはまったく無関係ではなく、双方の機能を使用しながら処理を進めることで便利性が増します。UNIX で使用できるウィンドウ・システムと一般的なウィンドウ・アプリケーションの使用方法、そして日本語の入力の方法を紹介します。

X Window システムの構成



X Window

ウィンドウ・システムは１つのディスプレイに、柔軟性のある複数の仮想的なウィンドウをグラフィックスで表示することができ、マウスなどのポインティングデバイスを用い、直感的にオペレーションができるようなシステムです。**Apple 社の Macintosh** や、**Microsoft 社の Windows95/98, WindowsNT** などは、このウィンドウ・システムが前提になって動作する OS です。UNIX はウィンドウシステムなしでも、ほとんどの作業を行うことができますが、ウィンドウ・システムを用いると、別次元のすばらしいユーザー環境を得ることができます。

UNIX で標準的に用いられているウィンドウ・システムは 1984 年にアメリカの**マサチューセッツ工科大学** (Massachusetts Institute of Technology -MIT-) によって開発された **X Window システム**が用いられています。この X Windows システムを Linux や、FreeBSD 等の PC UNIX に移植する非営利目的のプロジェクトが **XFree86 プロジェクト** (http://www.xfree86.org) で、ここからプログラムが配布されています。

1999 年現在で使用できる X Window のバージョンは 11、リリースは６で、これを **X11R6** と呼びます。

X Window の基本的構成

X Window の構成は次の３つからなります。

X サーバー

キーボード、マウス、ディスプレイの制御を行うプログラム。全てのアプリケーションは、X サーバーに対し画面への描画や、マウスから送られたイベントの処理を依頼します。X サーバーが稼動していなければ、X Window のアプリケーションは実行できません。

ウィンドウ・マネージャ

X Window の見た目 (Look & Feel) や操作方法を定義するためのプログラムです。同じ X サーバーでも、ウィンドウ・マネージャの種類を変えると、異なる見た目で表示されます。基本的な操作方法が大きく変わることはありませんが、アイコン、メニュー、ウィンドウの操作などに影響します。利用者は好みのウィンドウマネージャを選択できます。

X クライアント（アプリケーション）

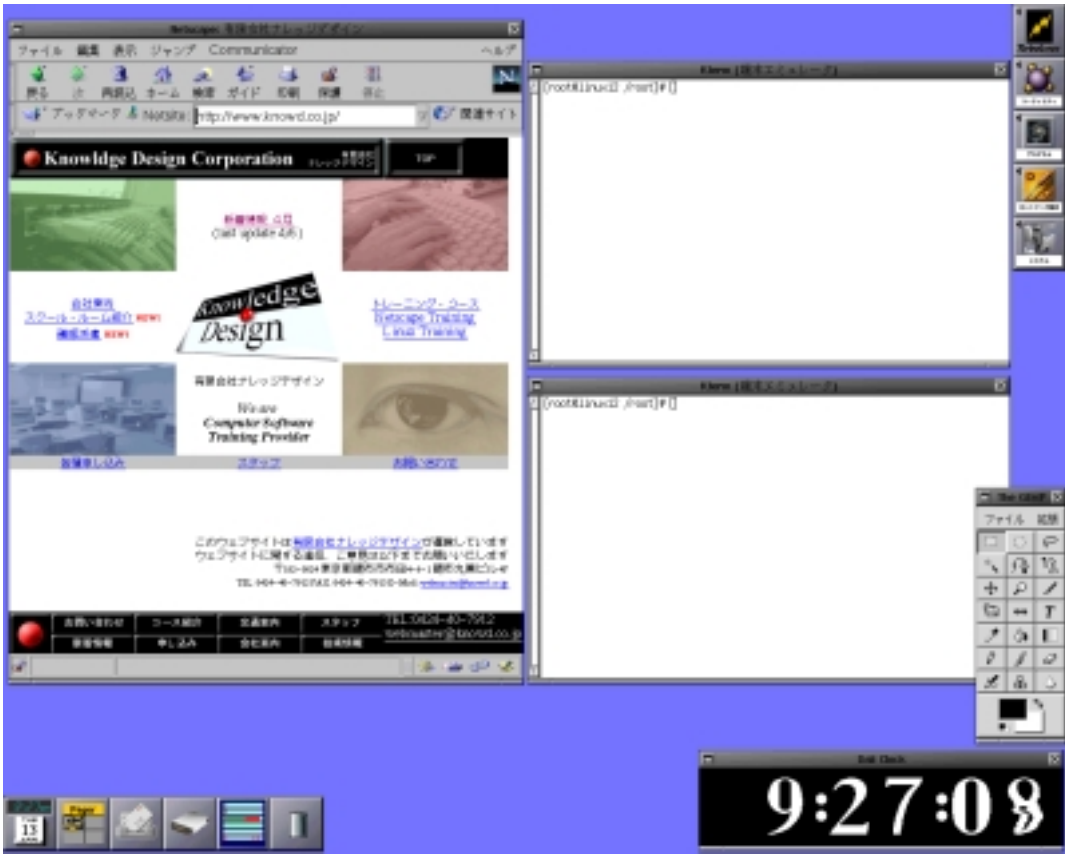
利用者が使うアプリケーションです。たとえば、エディタや、仮想端末、時計、Netscape Communicator 等、たくさんのアプリケーションが使えます。X サーバーが稼動していなければ、稼動させることはできません。

クライアント・サーバーによるネットワーク機能

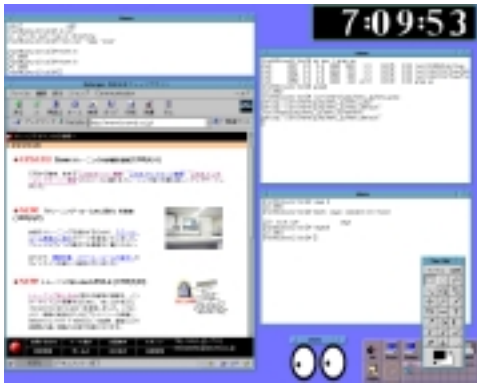
X Window のサーバーとアプリケーションは **X プロトコル**という通信規約を使用してネットワーク通信することができます。このため、サーバーとクライントはネットワーク上に存在すれば、**異なるコンピューターで別々に稼動させ処理を行うことができます**。

たとえば、実行したいアプリケーション（X クライアント）が自分のコンピューター上に存在しない場合、他のコンピューターで実行させ、自分のコンピューター上の X サーバーを通じ、表示、操作することが可能です。

X Window の外観と操作

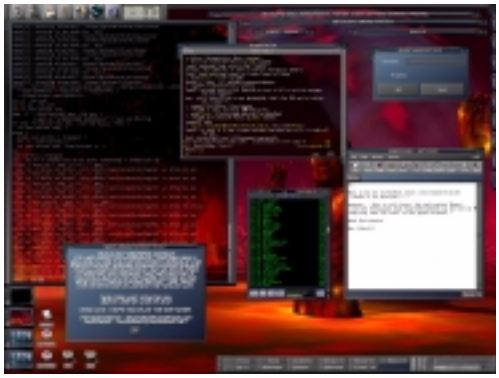


TurboDesk(Pacific HiTech http://www.pht.co.jp) Window Manager: AfterStep



Window Manager: FVWM

Window Manager: GNUstep



Window Manager: Enligthenment

Window Manager: KDE

X Window の外観と操作

X Window の外観と操作は、**設定されているウィンドウ・マネージャーによって異なります**。しかし、ウィンドウ・システムは、全てのことをあらかじめ知っていないユーザーであっても、操作できるよう、工夫がなされています。いろいろなポイントを操作しながら、操作感覚をつかむのに、それほどたくさんの時間はかからないでしょう。

使用する入力装置 キーボードとポインティング・デバイス

キーボード

X Window 上で作業する際であっても、UNIX コマンドは性格の変更無く使用できます。また、X Window のアプリケーション (X クライアント) も、コマンドの一種ですので、コマンドラインから起動することもできます。

ポインティング・デバイス（マウスや、トラックボール、パッド等）

ウィンドウ内の意味のあるポイントを状況に応じて指定し、マウスについているボタンを押す動作と組み合わせ、システム上での作業を行います。

マウスには、通常、１～３のボタンがついています。一般的な操作では、**左のボタンが選択、決定**に使われ、**右のボタンは、メニュー操作**を行うときに使われます。又、**３つめの真ん中のボタンは、さらに特殊な機能**が割り当てられる場合があります。左ボタンが最も使われるでしょう。

様々なポインティング・デバイス

Intelli Mouse

2Button

3Button

Track Ball

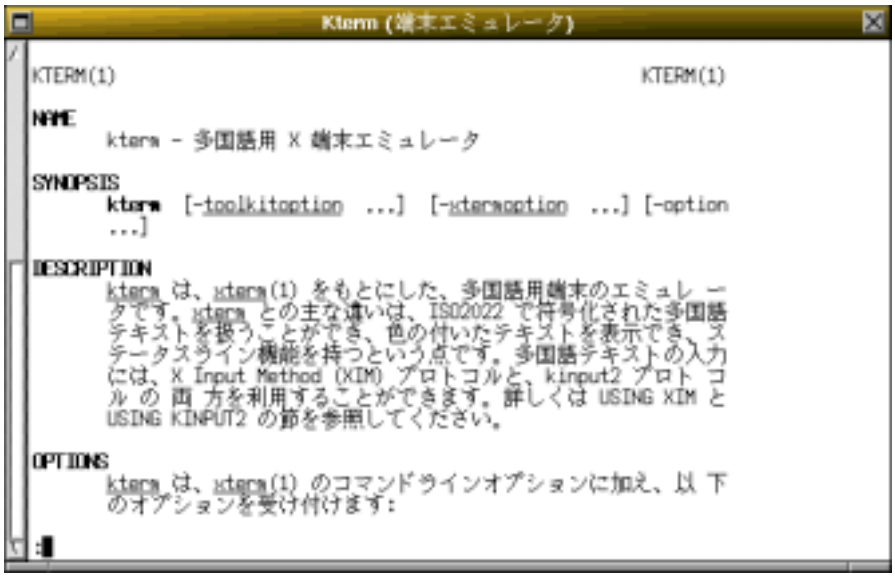
ウィンドウシステムの起動

```
$ startx
```

仮想端末 kterm の起動

メニューから選択（左ボタンをクリック）

kterm からは UNIX コマンドが使用できる



ウィンドウ・システムの起動

X Window システムを起動するには、X サーバー、ウィンドウ・マネージャの起動が必要です。Linux, FreeBSD 等で使用されている、XFree86 では、**startx というコマンド**でそれらを起動することができます。

```
$ startx
```

ウィンドウ・マネージャー AfterStep を用いての操作

起動するウィンドウの状態は、Linux のディストリビュート毎に異なります。ここではパシフィック・ハイッテック社 (http://www.pht.co.jp) のディストリビュートである "Turbo Linux3.0" の初期のセットアップ状態を使用します。(*TurboDesk の詳細は、パッケージ同封のユーザーガイドをご覧ください)

ウィンドウ・マネージャ AfterStep をベースにした、TurboDesk が起動します。

基本的な操作

UNIX コマンドを使用する -kterm-

ウィンドウ・システム上で UNIX コマンドを実行するために、仮想的な端末を起動します。xterm はその代表的なアプリケーションで、kterm は xterm の日本語版です。

画面、右隅に次のようなメニューがありますので、「ユーティリティ」と書かれたメニューを、**左ボタンでクリック**します。

横方向にメニューがスライドしてきます。"kterm" と書かれたメニューを同様に左ボタンでクリックます。

kterm が起動し、UNIX コマンドが使用できるようになります。

ウィンドウは、複数個開くことができます。重なり具合を調整したときは、前に出したいウィンドウのタイトルの部分を左ボタンでく陸してください。

初期状態のメニューから起動できるその他のアプリケーション

GIMP

GNU Image Manipulation Program http://www.gimp.org

高機能なグラフィック操作プログラム

xpaint

グラフィック・ドロー・プログラム

Netscape Communicator

Netscape Communications Co. http://home.netscape.com

インターネットのサービスを利用する複合的なアプリケーション

dp/NOTE

オムロン・ソフトウェア社 http://www.omronsoft.co.jp

商用ワープロソフトの試用版

mule

GNU http://www.gnu.org

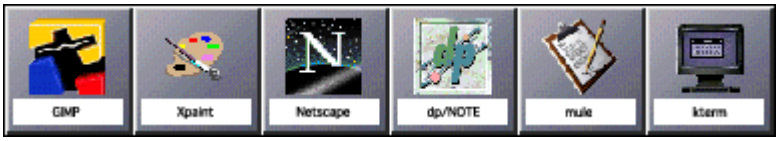
柔軟な拡張性を持つ高性能なテキスト・エディタ
Emacs の多国言語バージョン

アプリケーションの起動と終了

kterm 以外にもたくさんのアプリケーションを使用することができます。アプリケーションを起動するにはウィンドウ・マネージャーで提供されているメニューから起動するか、または、コマンド・ラインからアプリケーションを実行します。

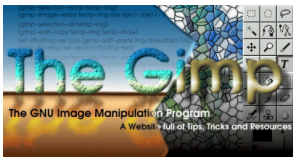
初期状態のメニューから起動できるアプリケーション

メニューは好みのものにカスタマイズ可能ですが、kterm のように初期状態で用意されているアプリケーションには次のものがあります。



GIMP (GNU Image Manipulation Program http://www.gimp.org)

非常に強力な、フリーのグラフィック・ツール。画像のフィルター処理、加工、作成などが行え、特殊効果を加えるためのたくさんのプログラムがプラグ・インとして利用できます。Adobe 社 (http://www.adobe.co.jp) の PhotShop というグラフィックス・ツールよく比較されます。



xpaint

フリーのドロー系グラフィックス・ツール。フリーハンド、線、図形、塗りなどのツールを使用して、ビットマップを描くことができます。

DonateWare 社の Torsten Martinsen 氏 (torsten@danbbs.dk) によってメンテナンスされており、「同社にクッキー等を寄贈してくれるとうれしい」とヘルプに記述されています。

Netscape Communicator 4 (Netscape Communications Co. http://home.nescape.com)

インターネットで利用されている、ウェブや FTP、電子メール、ニュース等のサービスを利用するための複合的な機能を持つアプリケーション。Netscape Communications 社より無料で提供されています。日本語化は様々な人からの寄贈によって実現されています。

dp/NOTE

Turbo Linux3.0/Pro に付属してくる、オムロンソフトウェア株式会社 (http://www.omronsoft.co.jp) が開発した、多機能な日本語ワードプロセッサの試用版。正式な利用のためには製品版を購入する必要があります。

mule (多国言語化された GNU emacs http://www.gnu.org/manual/emacs-20.3/emacs.html)

UNIX のエディタとして、非常に長い歴史を持っています。様々な機能拡張が行え、LISP という言語を用いてプログラミングも可能なテキスト・エディタです。

それぞれのアプリケーションの使い方は、付属のマニュアルや書籍などをご覧ください。

一般的なウィンドウ操作

ダブル・クリック

同じポイントを短時間に２回連続でクリック

ドラッグ

ボタンを押した状態で、マウスを移動

一般的なウィンドウ操作

ウィンドウの方法は、ウィンドウマネージャごとに多少異なります。TurboDesk を例に基本的なウィンドウの操作をみいきましょう。

ウィンドウの終了

各ウィンドウの右上に X というボタンを押すとプログラムは、終了します。

ウィンドウのアイコン化

各ウィンドウの左上にあるボタンををおすと、ウィンドウが小さくシンボルに変わります。このシンボルをアイコンと呼びます。**アイコンの状態でもプログラムは処理を続けています。**

アイコンをウィンドウに復元

アイコンの上で、ボタンをすばやく２回押します。この操作をダブル・クリックといいます。

ウィンドウの大きさを変える

ウィンドウの右下、左下の隅にポインターをあわせ、左ボタンを押したまま、マウスを動かします。好みの大きさが決定したら、ボタンを離します。この操作をドラッグといいます。

ウィンドウの位置を変更する

ウィンドウの上部のタイトル部分をドラッグして、好みの位置に配置することができます。

ウィンドウの重なり具合の調整

タイトル部分をクリックすると、ウィンドウは最前面に表示されます。

アクティブなウィンドウ

複数のウィンドウを画面に開いた状態だったとしても、実際に操作の対象になるウィンドウは一つです。アクティブなウィンドウにするには、タイトル部分をマウスでクリックします。タイトルに色がつき、視覚的にか確認することができます。

コマンドラインからアプリケーションを実行

通常のコマンドの実行と同じ

& をつけて実行すると便利（バックグランド・ジョブ）

コマンドラインからのアプリケーションの起動

ウィンドウ・システムにおけるアプリケーションも、UNIX コマンドと同様、コ**マンド・ラインから起動することができます**。たとえば、Netscape Communicator は netscape というコマンドで実行できます。メニューに用意していないコマンドは、コマンド・ラインから起動します。もし、頻繁に使うようでしたら、便利性をあげるために、メニューのカスタマイズをするとよいでしょう。

コマンドラインからの実行

コマンド・ラインから、アプリケーションを実行します。kterm 等を起動し、コマンド・ラインからアプリケーション名（コマンド名）を入力します。つまり、メニューの操作と違い、あらかじめコマンド名を知っておく必要があります。

時計 (xclock) を実行する

```
$ xclock
...
```

xclock が実行されます。コマンド・プロンプトに注目してください。プロンプトが返ってきません。xclock を終了すれば、プロンプトはもどってきます。(コマンド・ラインから ^C を入力しても終了できます)

ウィンドウのアプリケーションをコマンドラインから起動するときは、**末尾に & をつけて実行すると便利です**。& をつけたコマンドは、UNIX のマルチタスクの機能で、**バックグランド・ジョブ**として実行されます。

```
$ xclock &
$
```

プロンプトはすぐ復帰する

X Window のアプリケーションの例

X Window のいくつかのアプリケーションを紹介します。詳しい説明はオンライン・マニュアルをみてください。実際に実行して楽しんでください。

ユーティリティ

xcalc　電卓

xdaliclock時計

背景

xsetroot壁紙の色や、画像を変更します ex.) $ xsetroot -sold white

xsnow　背景に雪を降らせます

xearth 背景に地球を出します

ゲーム、その他

xeyes　マウス・ポインタの位置を追跡する目が出てきます

xbill　ユーモアのあるゲーム

xboing ブロック崩しのゲーム

グラフィックスのデモンストレーション

kumppa laser

lament ifs

hopalongjulia

kaleidescopehypercube

Netscape Communicator の起動

$ netscape &

URL の構成

サービス名 :// ホスト名 . ドメイン名 / ディレクトリ / ファイル

EX.) http://www.knowd.co.jp/staff/rotake/index.html

Netscape Communicator を使用してウェブにアクセスする

インターネットのウェブへのアクセスや、電子メールをやり取りするためのアプリケーションはたくさんありますが、ここでは、Netscape Communicator を使用した例を紹介します。

Netscape Communicator の起動

コマンドラインから netscape と入力します。又は、ウィンドウ・システムのメニューから Netscape を選択しても起動できます。

```
$ netscape &
```

いくつか警告が出てきても、深刻なものではないので無視してください。

いくつかのウェブサイトにアクセスする

UNIXやLinuxに関連の深い情報を公開しているウェブサイトに実際に、アクセスしてみましょう。ウィンドウ上部のフォームに URL(Uniform Resource Locator) を入力します。

URL は、インターネット上に存在するコンピューターの名前と、そのコンピューターが持っているデータのファイル名からなります。コンピューターの名前の一意性は各地域の Network Information Center (日本では、JPNIC http://www.nic.ad.jp) への登録制になっています。

先頭の http:// は省略できます。

URL は、次のような構成になっています。

サービス名 :// ホスト名 . ドメイン名 / ディレクトリ / ファイル名

このクラスに関係のあるいくつかのウェブサイトにアクセスできることを確認してみましょう。目的のサイトを探すには、検索エンジンでキーワード検索すると見つかるかもしれません。

検索エンジン

www.yahoo.co.jp
www.infoseek.co.jp
www.lycos.co.jp
www.goo.ne.jp
www.fresheye.co.jp
www.excite.co.jp

UNIX,Linux 関連

www.pht.co.jp	パシフィック・ハイテック社
www.redhat.com	レッド・ハット社
www.linux.org	Linux の情報を集めたサイト
www.linux.or.jp	日本 Linux 協会 ,1999 年４月に発足
www.gnu.org	GNU のサイト
sunsite.sut.ac.jp	東京理科大のサイト、フリーの UNIX を多数集積、公開している
www.oreilly.co.jp	株式会社オライリー・ジャパン、UNIX, Linux の信頼性の高い書籍を多数刊行、ローカライズ

日本語の入力

```
$ kinput2 -canna &
```

日本語入力の基本操作方法

日本語英語の切り替え	[Shift] + [スペース]
変換、次候補	[スペース]
文節の選択、前進、後進	→、←
次候補、前候補	↓、↑
選択範囲の延長、縮小	[Shift]+ →、[Shift]+ ←
決定	[リターン]

Netscape Communicator を利用して電子メールを送受信する

ウェブ・ブラウザと同様、電子メールを使用するアプリケーションも様々なものがありますが、ここでは一例として、Netscape Communicator を用いたメールの送受信を紹介します。

日本語を扱えるようにする

表示

TurboLinux3.0 に含まれている Netscape Communicator では、日本語の表示に必要な設定があらかじめ行われています。もし、日本語の表示がうまくいかないときは、メニューの [表示][文字コードセット][日本語（自動選択)] を選択します。

入力

X Window で日本語を**入力**するときは、日本語入力のためのプログラムを起動させなくてはなります。初期状態では、起動していないので、次のコマンドを実行し起動します。

```
$ kinput2 -canna &
```

kinput2 がパック・グランドで稼動している間は、日本語を入力することができます。しかしながら、**kinput2 の起動前に実行されているアプリケーションには作用しない**ので、すでに起動しているアプリケーションは、終了して、再び立ち上げ直す必要があります。日本語入力を頻繁に行うのであれば、最もいい方法は、ウィンドウシステム起動と同時に、このプログラムを起動することです。

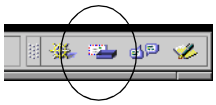
日本語を入力する際のキー操作（canna）

kinput2 を起動する際、使用するかな漢字変換の方式を canna（かんな）にしました。ここでは、canna の日本語入力の基本的操作方法の一覧をあげます。

日本語英語の切り替え	[Shift] + [スペース]
変換、次候補	[スペース]
文節の選択、前進、後進	→、←
次候補、前候補	↓、↑
選択範囲の延長、縮小	[Shift]+ →、[Shift]+ ←
決定	[リターン]

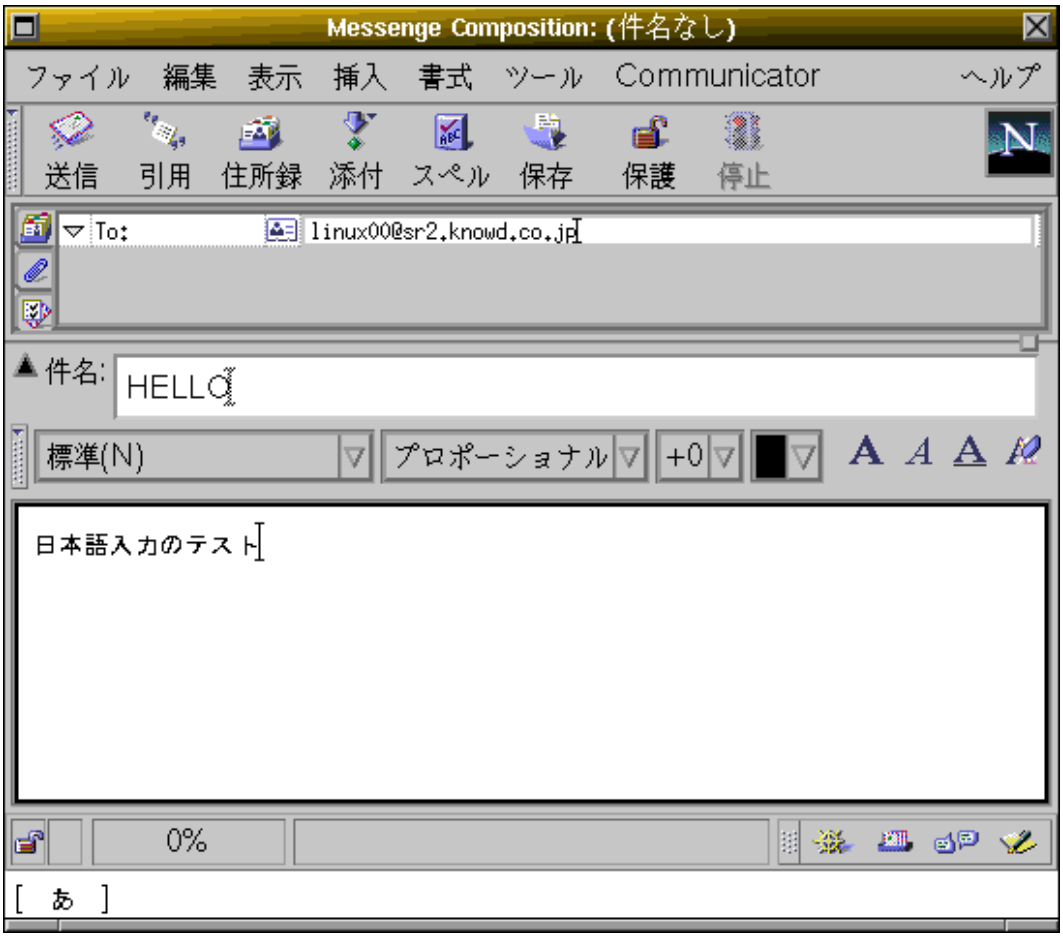
メールツール (Netscape Messenger) を起動する

Netscape Communicator の右下のコンポーネント・ドックには、Communicator が用意しているいくつかのサービスのアイコンが表示されています。ここから、電子メールをやり取りするための、メールツール（左から２番目）を起動します。



電子メールを書く

To: rotake@knowd.co.jp
件名： HELLO
メッセージ： 日本語入力のテスト



メールを読む

左上の、**「取り込み」**ボタンを押すと、あなた当てにきている目屡があるかどうかを確認し、あれば、読み込んできます。

メニューの下に、到着しているメッセージの一覧が表示されます。**件名、差出人、到着時刻**などの情報が、１行で表されますので、読みたいメールの上でクリックすると、下に、内容が表示されます。

別のメールを読みたいときは、別のメールをクリックしてください。

また、メールのリストの並び替えを行いたいときは、「件名」「差出人」「日付」などのリストのタイトルをクリックししてください。

メールを書き送信する

メールを送信する際に必要なのは、相手のメールアドレスです。メールアドレスは、前もって相手から聞いておきましょう。形式は次のようになります。

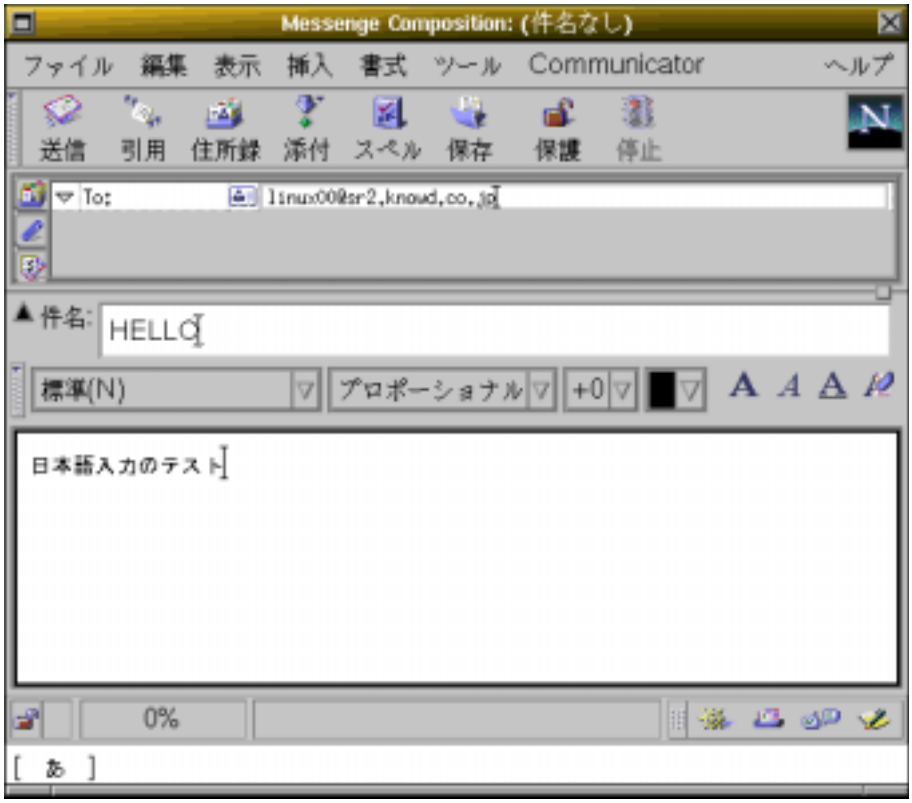
ユーザー名 @ ドメイン名 (ex. rotake@knowd.co.jp)

メニューの下の To: という行があるので、その後ろに**送信先**を書いてください。

件名は、受け取った人がメッセージをすばやく知ることができるように、内容を簡単にまとめたものを指定して下さい。

本文に、送りたいメッセージの内容を記述します。日本語が必要であれば、[Shift]+「スペース」でモードを切り替え、入力します。

送信は、上の**「送信」ボタン**を押すと、今作ったメールを相手に向かってメッセージを送信します。



ウィンドウの終了

又は [Ctrl] + [Alt] + [Backspace]
ログアウトはしない

ウィンドウのロック

又は
$ xlock

元に戻すためにはパスワードが必要

ウィンドウシステムの終了

ウィンドウ・システムを終了するには、メニューの [TurboLinux] から [終了] を選択してください。ウィンドウシステムは終了しますが、**ログアウトしたわけではない**ことに注意します。その後、logout コマンドでログアウトしてください。

ウィンドウシステムの強制終了

何かの理由で、ウィンドウ・システムが制御不能に陥ってしまった場合は、**[Ctrl] + [Alt] + [Backspace]** を同時に押してください。ウィンドウ・システムを強制的に終了することができます。**作業していたウィンドウも警告なしに終了**します。たとえば、メールを書いていた途中であっても、その文章は失われてしまいますので、最終的な手段として使用します。

ウィンドウをロックする

一時的に席を離れる場合や、明日、作業を継続したい場合など、ウィンドウを他人に見られたり触られないようロックすることができます。メニューの **[TurboLinux]>[ロック]** を選択すると、画面が黒くなりグラフィックスのデモンストレーションが始まります。これを解除してもとに戻すには、何かのキーを押した後、ユーザーのパスワードを入力しなければ復帰できません。画面のロックは、xlock コマンドでも行えます。

```
$ xlock
```

第５章 練習問題

１） X Window システムを構成する、３つの要素をあげてください。

２） X というコマンドについての日本語のマニュアルを表示してください。[実習]

３） Linux の作者、Linus Torvalds 氏のホームページにアクセスしてみましょう。[実習]
http://www.cs.Helsinki.FI/~torvalds/

３） http://www.yahoo.co.jp にアクセスし、「ナレッジデザイン」というキーワードで検索を行ってください。[実習]

４） １０秒後にウィンドウをロックするようなコマンドラインを実行してください。[実習]

５） 講師に日本語でコースに関する現在までの、質問、意見、提案、要望などを送ってください。メールアドレスは講師から聞いてください。[実習]

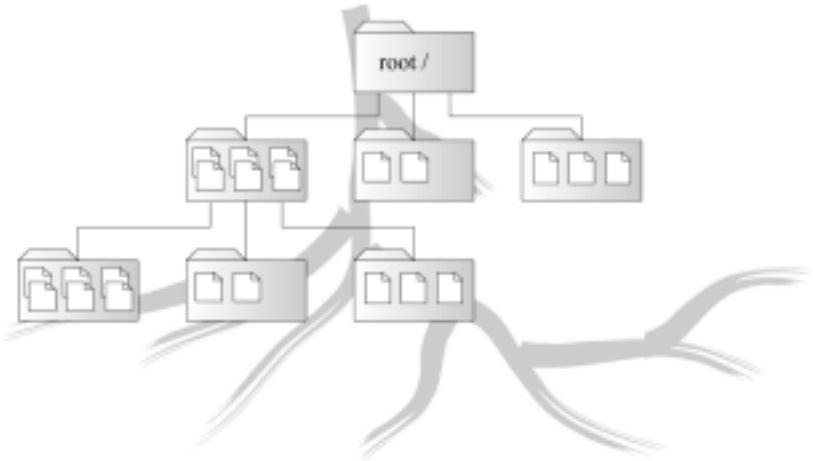
６） xbill で遊んでください。[実習]

第６章 ファイルとディレクトリの操作

学習目標

ソフトウェアや、ユーザの作成したデーターを保存、管理するということは OS の最も重要な役割の一つです。UNIX システムでは、それらの情報がシステム上で、どのように管理されているかという概念の理解と、利用者はそれをどのようなコマンドを用いて利用するかを学習します。

ファイル・システムの概念



情報（データ）保存方法

ハードディスク、フロッピーディスク、CD-ROM 等は、すべて大量の情報を保存することを目的で作られたメディアです。メディア上では通常、単に磁気信号が並んでいるに過ぎません。たくさんの情報がある中で重要な要素は、目的の情報が見つけ出しやすく整理できる仕組みです。

情報を整理し、扱いやすいものにするのが、ファイルシステムです。整理の方法として、それらを**「分類」**すると言う方法が考えられます。UNIX のファイルシステムは、メディアに記憶されている情報を次のような概念で、扱うことができるような仕組みを提供します。

ファイル・システムの提供する概念

ファイル

UNIX システムでは情報をある一かたまりとして扱え、それに名前をつけることがきます。一冊の本を想像するとわかりやすいでしょう。

表紙にタイトルがなければ本は非常に探しにくくなります。ある単位に情報をまとめ、その内容に見合った名前をつけておくことは、情報の管理に有効な手段です。UNIXでは、一冊の本に対応するのが、**ファイル**と呼ばれます。

あらかじめ用意されていないファイルの内容と名前は利用者が決めます。

UNIX システム上でファイルはデータの保存以外に、**コマンド、デバイス、ディレクトリ**として使われます。

ディレクトリ

ファイルが大量になると、名前だけでは管理が困難になる場合があります。大きな図書館で本を探すときのことを想像してください。何万冊もある本の中から、目的の本を探し出しやすくするために、「分類」ガなされているはずです。似たような目的を持つものをまとめ、場合によっては分けておくことで、目的にたどり着きやすい場合があります。

UNIX ではその分類を行うために、ディレクトリを用います。ディレクトリには複数のファイルを**入れておくことができます。**このディレクトリも利用者が自由に名前をつけ、構成することができます。たとえば、会社用のファイルと、私用のファイルを、異なるディレクトリに配置することができます。

ディレクトリの中にディレクトリを含んでも問題ありません。たとえば、「私用」というディレクトリの中に、「旅行プラン」「手紙」というディレクトリを作ることもできます。

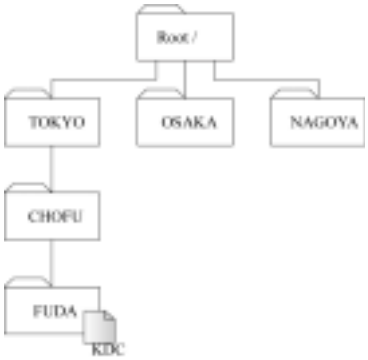
階層構造

これらを全体的にみると、一本の木を逆さまにしたような姿になります。UNIX のファイル・システムでは、最も、頂点にあるディレクトリを ルート・ディレクトリと呼びます。全てのディレクトリは、**ルート・ディレクトリ**の下に位置することになります。

１つの UNIX システムは、必ず、唯一のルート・ディレクトリを持ちます。もし、複数の物理的記憶装置が合ったとしても、ルート・ディレクトリが増えることはありません。その記憶装置は、一つのルート・ディレクトリから始まるディレクトリの中の１ディレクトリとして扱うことになります。

絶対パスによる表現

ルート・ディレクトリからの経路で表わす



/TOKYO/CHOFU/FUDA/KDC

カレント・ディレクトリ

現在作業中のディレクトリ

pwd コマンドで確認できる

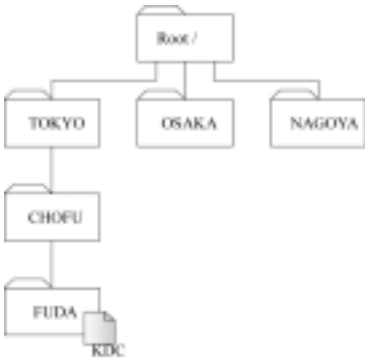
.（ピリオド）でも表現できる

ファイルの表現方法

ファイルシステム上のファイルを文字を使ってコマンドライン上から指定する際の表現方法似ついてみていきます。

絶対パスによる表現

上ののようなディレクトリ構造の場合、ルート・ディレクトリから KDC というファイルまでの経路を指定することで KDC というファイルが表現されます。



/TOKYO/CHOFU/FUDA/KDC

もっとも左に位置する /(スラッシュ）は、ルート・ディレクトリを表現しています。それ以外の / は特別な意味を持たない区切り記号です。

ルート・ディレクトリから、目的のファイルまでに経過した、ディレクトリの経路（パス）が順番に書かれます。

ルート・ディレクトリを起点とした表記方法を、**絶対パス**といいます。

作業ディレクトリ（カレント・ディレクトリ）

UNIX システムで作業をする際（ログインしている間)、利用者は必ず、ファイルシステムのどこかで作業することになります。現在作業を行っているディレクトリを、**作業ディレクトリ**、又は、**カレント・ディレクトリ**と呼びます。

カレント・ディレクトリを調べるには、pwd を実行します。pwd は利用者がどこで作業を行っているかを、絶対パスにより表示します。

```
$ pwd
/home/yyamazaki
```

カレント・ディレクトリは絶対パスで表現することもできますし、. (ピリオド）で表現することもできます。

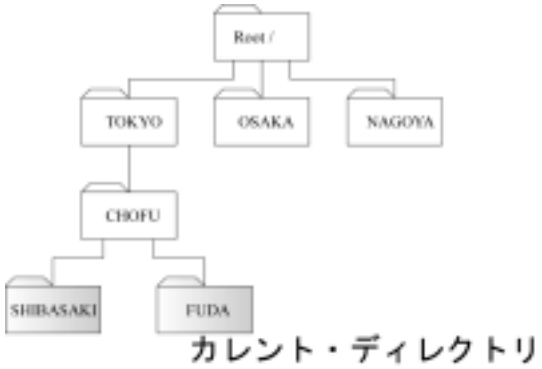
親ディレクトリ

カレント・ディレクトリより一つ上

..（ピリオド２つ）で表現できる

相対パス

カレント・ディレクトリからの経路で表わす



../SHIBASAKI

絶対パスとの相違点

/（スラッシュ）で始まらない
カレント・ディレクトリが異なると指し示すファイルも変わる

親ディレクトリ

親ディレクトリとは、カレント・ディレクトリから見て、一つ上のディレクトリのことを言います。

カレント・ディレクトリが /TOKYO/CHOFU/FUDA であるという前提では、親ディレクトリは /TOKYO/CHOFU になります。

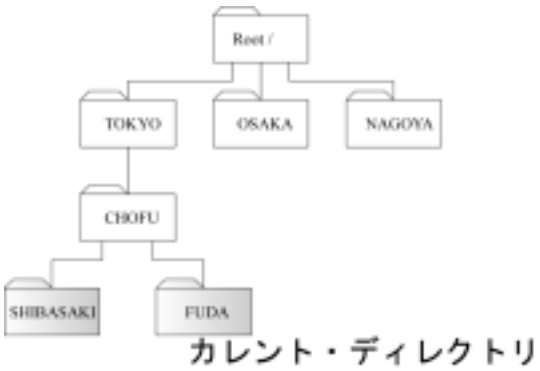
また、別の表記で、カレント・ディレクトリがどこであっても、親ディレクトリを ..(ピリオド２つ) で表わすこともできます。

カレント・ディレクトリが /TOKYO/CHOFU/FUDA である場合、.. という表現をすると、絶対パスでは、/TOKYO/CHOFU になります。同一の場所を示す表記の仕方が２種類あるということです。

ただし、.. を用いて、/TOKYO/CHOFU を指し示すには、/TOKYO/CHOFU/FUDA がカレントディレクトリであるという前提が必要になります。カレント・ディレクトリが、別のディレクトリならば、.. で指し示されるディレクトリは、/TOKYO/CHOFU ではなくなります。

相対パスによる表現

絶対パスはルート・ディレクトリからの経路で、ファイルの場所を表現したのに対し、相対パスは、カレント・ディレクトリからの経路でファイルの場所を表現した表記方法です。



/TOKYO/CHOFU/SHIBASAKI を相対パスで表現すると次のようになります。

../SHIBASAKI

絶対パスと異なる点は、先頭が / で始まっていない点だけです。先頭が、/ で始まらない場合は、カレント・ディレクトリが起点になります。ですから、上の表記方法は ./../SHIBASAKI と同義です。

同一の階層にあるディレクトリ SHIBASAKI を指定する際は、「親のディレクトリの下にある SHIBASAKI」と指定します。もし、先頭に .. をつけない場合は、[カレント・ディレクトリの下の SHIBASAKI」という意味になります。

絶対パスでも相対パスでもどちらでも、あるファイルを示すことはできます。

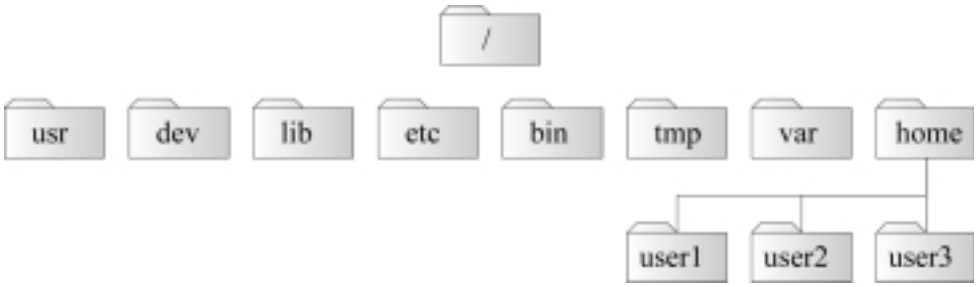
必要に応じて、使い分けてください。

ホーム・ディレクトリ

システム管理者が決めた、一般ユーザーの作業ディレクトリ

ログインしたときはホーム・ディレクトリがカレント・ディレクトリ

UNIX ディレクトリの典型的な例



ホーム・ディレクトリ

利用者は、システム管理者から、専用の作業ディレクトリを与えられます。このディレクトリをホーム・ディレクトリと呼びます。**ログインすると、利用者は自動的にホーム・ディレクトリがカレント・ディレクトリになります**。利用者は、ホーム・ディレクトリ内で自由にファイル操作を行うことができます。

UNIX ディレクトリの典型的な例

システム管理者や利用者は、自分で好みのディレクトリを作成することが可能です。しかしながら、UNIX には、初期の状態でシステムが使用するディレクトリが用意されています。様々な UNIX システムで一般的に使用されるディレクトリは以下のとおりです。

/bin　UNIX コマンド、ユーティリティが収められています。(binary)

/dev　デバイスファイルが収められています。(device)

/etc　システムを維持するために必要な設定ファイル群が収められます。

/var　システム監視のためのログ・ファイルなどが収められます。(variable)

/lib　プログラムが使用するライブラリが収められています。(library)

/usr　一般ユーザーが使用するファイルが収められています。(user)

/home　一般ユーザの作業用のホーム・ディレクトリ。

/tmp　システムや、ユーザーが一時的に使うディレクトリ。システムの再起動でないようはクリアされる。(temporary)

これらのファイルのいくつかは、システム管理者が UNIX システムの環境を整えるためにメンテナンスを行うもので、一般ユーザーの誤った作業によりシステムに不具合が起こらないような仕組みを持っています。。

ls

ディレクトリ内のファイルの情報を表示

$ ls /bin

/bin 内のファイルを表示

$ ls -F

/	ディレクトリ
*	実行可能ファイル
@	シンボリック・リンク

$ ls -x | less

標準出力以外であってもマルチカラムで表示

ファイル操作コマンド

UNIX システムのファイルを操作する基本的なコマンドを紹介します。

ディレクトリのリスト表示
ls

概略　ディレクトリ内の情報を表示します。引数に何も指定しなければ、カレントディレクトリ内のファイルだけを表示ます。

```
$ ls
file1 file2 file3 file4.....
```

引数　ディレクトリ名　指定したディレクトリ以下にあるファイルのリストを表示します。

例）/bin 以下のファイル名を表示する

```
$ls /bin
arch           df             hostname       nisdomainname  sync
ash            dmesg          igawk          ping           tar
ash.static     dnsdomainname  ipcalc         ps             tcsh
awk            doexec         kill           pwd            touch
basename       domainname     ksh            red            true
bash           echo           ln             rm             umount
bsh            ed             login          rmdir          uname
cat            egrep          ls             rpm            usleep
chgrp          ex             mail           runwait        vi   man
```

-Fディレクトリの後ろに /、コマンドのような実行可能なファイルの後ろに *, シンボリックファイルの後ろに @ をつけて、ファイルの種類がわかりやすい表示をします。

```
$ ls -F /etc
DIR_COLORS            HOSTNAME              X11/
adjtime               amd.conf              at.deny
atalk/                auto.master           auto.misc
bootparams            codepages/            conf.modules
cron.daily/           cron.hourly/          cron.monthly/
cron.weekly/          crontab               csh.cshrc
default/              dosemu.conf*          dosemu.users
dumpdates             ebw3d.conf            enscript.cfg
exports               fdprm                 fstab
fstab.bak             ftpaccess             ftpconversions
ftpgroups             ftphosts              ftpusers
gated.conf.sample     gettydefs             gpm-root.conf
....
```

-x　マルチカラムで表示します。通常の ls はマルチカラム（複数の桁）で表示されており、たくさんのファイルがコンパクトに表示されます。しかし、標準出力以外に ls の結果を出力すると、１行１ファイルで表示されます。標準出力以外であっても、マルチカラムで表示させたいときは、-x を使用します。

例）| less の方法を使ってもマルチカラムで表示させたい

```
$ ls -F /etc | less
DIR_COLORS
HOSTNAME
X11
adjtime
一行１ファイルで表示される
```

```
$ ls -Fx /etc | less

DIR_COLORS            HOSTNAME              X11/
adjtime               amd.conf              at.deny
atalk/                auto.master           auto.misc
```

$ ls -l

ファイルの詳細な情報を表示

```
-rw-r--r--   1 linux00  train  22 Apr 13 20:29 context
```

保護モード　所有者　サイズ　ファイル名
リンク数　グループ　更新日時

$ ls -a

隠しファイルを表示
隠しファイルは .（ピリオド）で始まるファイル名を持つ

```
$ ls -a
.       .bash_profile   .newsrc-news   .trash
```

$ ls --color

種類別に色分けして表示

...ls コマンドの続き

-l　　ファイルの詳細な情報を表示します。それぞれのファイルに関して次の情報が表示されます。１つのファイルに関し次のような情報を得ることができます。

```
-rw-r--r--   1 linux00     train         22 Apr 13 20:29 context
```

ファイルのパーミション (-rwxr--r--)

設定されているセキュリティのための保護モード

リンクカウント (1)

そのファイルを参照しているファイルの数

所有者 (linux00)

ファイルの所有者

グループ (train)

ファイルが属しているグループ

ファイルのサイズ (22)

ファイルのサイズをバイト数で表示

ファイルの更新日時 (Apr 13 f20:29)

ファイルが更新された日時

ファイル名 (context)

ファイルの名前

```
$ ls -lF /etc | less
total 852
-rw-r--r--   1 root     root         2045 Aug 10  1998 DIR_COLORS
-rw-r--r--   1 root     root           24 Mar 17 09:52 HOSTNAME
drwxr-xr-x  12 root     root         1024 Mar 17 09:50 X11/
-rw-r--r--   1 root     root           10 Nov  6 10:05 adjtime
```

-a　　隠しファイルを表示ます。隠しファイルとは .(ピリオド) で始まる名前を持つファイルで -a なしでは、表示できません。

隠しファイルは、あまり変更頻度の高くないファイルを ls で出力させたくないとき使用します。よく、環境設定用のファイルが、隠しファイルとして用いられます。

```
$ ls -a
.              .bash_profile  .newsrc-news   .trash         a
..             .bashrc        .prxa06786     .xcopilot      aa
.ICEauthority  .cshrc         .seyon         .xinitrc       ccc
.Xdefaults     .gimp          .steprc        .xsession      nsmail
```

-color　　ファイルの種類別に色をつけます。初期状態では、ディレクトリに青、実行可能ファイルに緑、シンボリックリンクされたファイルは水色で表示されます。

ディスプレイや端末によってうまくいかないときもあります。

ディレクトリの移動

$ cd ディレクトリ名

絶対パス、相対パスで移動先を指定する
ディレクトリを指定しなければ、ホーム・ディレクトリへ移動

ファイル名による検索

$ find ディレクトリ名 -name ファイル名

ディレクトリ以下から、ファイルの絶対パス名を検索する

作業ディレクトリを変更
cd

概略 Change Directory。作業ディレクトリを変更します。引数の指定を省略するとホーム・ディレクトリにもどります。

```
$ cd
$ pwd
$ /home/yyamazaki
```

引数 ディレクトリ名　移動先のディレクトリ名を絶対パスか相対パスで指定します。

例）/bin, /usr, /etc、親ディレクトリ、そしてホームディレクトリに移動します。

```
$ pwd
/home/yyamazaki
$ cd /bin
$ pwd
/bin
$ cd /usr
$ pwd
/usr
$ cd ..
pwd
/
$ cd
$ /home/yyamazaki
```

ファイルの検索
find

概略 指定されたディレクトリ以下全てを対象に、特定のファイルを検索します。比較的複雑な引数をとりますので、ここでは、特定のディレクトリ以下から、名前によりパス名を得るという方法を紹介します。

例）/usr 以下から rgb.txt というファイル名を検索します。

```
$ find /usr -name rgb.txt
/usr/X11R6/lib/X11/rgb.txt
find: /usr/doc/fetchmail-4.5.5/contrib/RCS: Permission denied
find: /usr/doc/pstotext-1.7: Permission denied
find: /usr/doc/readline-2.2.1: Permission denied
find: /usr/doc/xforms-0.88: Permission denied
find: /usr/lib/netplan.dir: Permission denied
/usr/share/emacs/19.34/etc/rgb.txt
```

説明 第１番目の引数は、検索対象のディレクトリを指定します。find はそれ以下のディレクトリ全てを対象とします。第２番目の引数に -name を指定すると名前による検索が行えます。次の引数には、探したいファイルの名前を指定します。

find: .....: Permission denied

という表示は深刻ではないエラーで、そのファイル、またはディレクトリ以下に関して、find を実行したユーザーの検索が許されていないというものです。これは、ファイルの所有者に許可を出してもらわなければ検索できません。

ディレクトリの作成

mkdir ディレクトリ名・・・・

ディレクトリは複数個指定できる

ディレクトリの削除

rmdir

ディレクトリ内にファイルが存在すると削除できない

ディレクトリの作成
mkdir

概略 Make Directory。ディレクトリを作成します。引数に作成したいディレクトリの名前を１つ以上、指定します。

例）personal, official, confidential という３つのディレクトリを作成します。

```
$ mkdir personal official confidential
$ ls -F
confidential/   official/    personal/
```

引数 -p　存在しないディレクトリがパスとして指定されても、強制的に作成します。

例）OS というディレクトリを作成し、その下に unix というディレクトリを作成します。os というディレクトリは存在しません。

```
$ mkdir os/unix
mkdir: cannot make directory ‘os/unix’: No such file or
director ディレクトリが存在しませんというエラーが出る
$ mkdir -p os/unix
$ cd os
$ ls -F
unix/
```

ディレクトリの削除
rmdir

概略 Remove Directory。空のディレクトリを削除します。ディレクトリの中にファイルが入っている場合は削除できません。

```
$ rmdir os
```

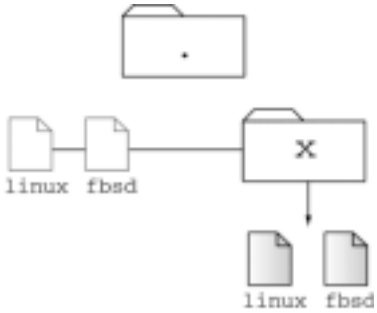
rmdir: os: Directory not empty	削除しようとした OS というディレクトリは空ではないというエラーが出る
$ ls -F os	os とうディレクトリの下を確認
unix/	unix というディレクトリがある
$ rmdir os/unix	os のしたの unix を削除
$ rmdir os	
$	

ファイルの移動、名前の変更

mv

ファイル名の前変更は、同一ディレクトリ内における移動

$ mv 移動元 ・・・ 移動先



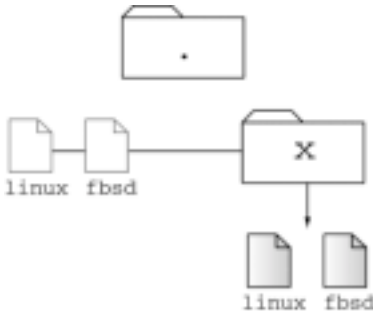

```
$ mv linux fbsd x
```

ファイルの移動、名前の変更

mv

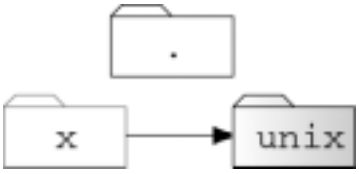
概略 Move。ファイル、ディレクトリの移動を行います。引数に移動先、移動元をパス名で指定します。ファイルの名前の変更はこのコマンドで行います。

例１）ファイル、linux, fbsd をディレクトリ x に移動します。



$ mv linux fbsd X

例２）ディレクトリ x を unix に改名します。



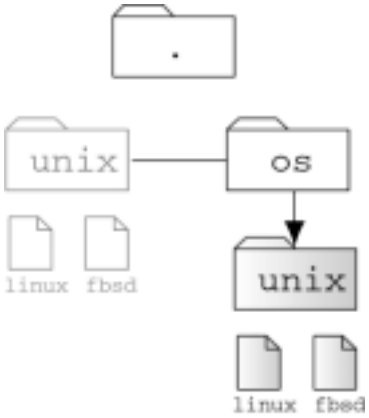
$ mv x unix

解説 mv コマンドでファイルを移動する際、引数の指定の仕方に気をつけてください。まずはじめに、移動元のファイルを指定します。これは、複数指定してもかまいません。最後の引数として、移動先のファイル名（ディレクトリ名）を１つだけ書きます。

$ mv 移動元 移動元 ... 移動先

ディレクトリのツリーごと、他のディレクトリに移動することもできます。この時、移動先のディレクトリが、すでに存在していれば、指定したディレクトリの下に移動元ディレクトリが移動されます。存在していなければ、新しく作成されます。つまり、ディレクトリ名の名前の変更と同義になります。

例）ディレクトリ unix（linux, fbsd ファイルを含む）を、os というディレクトリに移動します。os というディレクトリはすでに存在しています。

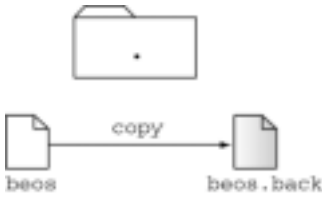


```
$ mv unix os
```

ファイル、ディレクトリのコピー

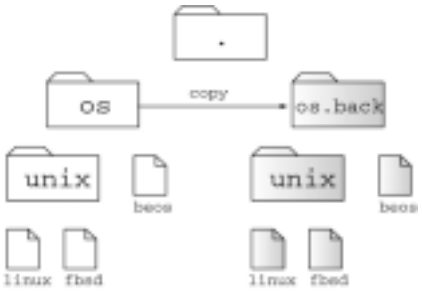
cp

$ cp コピー元　・・・ コピー先




```
$ cp beos beos.back
```

コピー元がディレクトリの場合は、-r オプションを指定

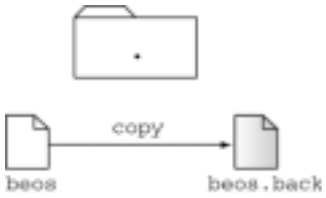



```
$ cp -r os os.back
```

ファイル、ディレクトリのコピー
cp

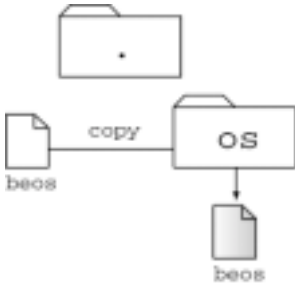
概略 Copy。ファイル、ディレクトリのコピーを行います。引数に移動先、移動元をパス名で指定します。mv コマンドと使い方はほぼ変わりませんが、ディレクトリを移動する際は、-r という引数をつけなくていけない点が異なります。

例１）ファイル、beos を beos.back というファイル名でコピーします。



$ cp beos beos.back

例２）beos を os というディレクトリの下にコピーします。



$ cp beos os/beos

又は

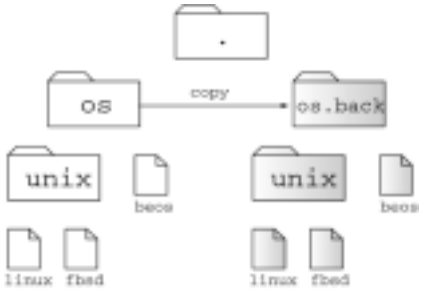
$ cp beos os

解説 cp コマンドも mv コマンド同様、引数のはじめにコピー元のファイルを指定します。これは複数指定できます。引数の最後にコピー先を指定します。コピー先は１つしか指定できません。

$ cp コピー元 コピー元 ･･･ コピー先

ディレクトリのツリーごと、他のディレクトリにコピーする場合、**mv コマンドと異なり、-r という引数をつけなくてはいけません。**

例）多くのファイルを含む os ディレクトリ以下のコピーを、os.back というディレクトリ名でコピーする。

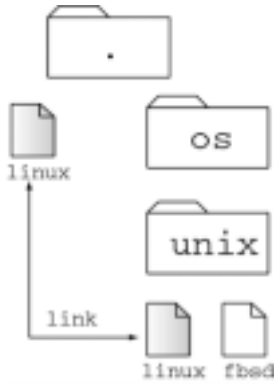


$ cp -r os os.back

ファイルのリンク

ln

１つのファイルに２つ以上の名前がつく
データはコピーされない



ハード・リンク

$ ln リンク元 リンク先

ディレクトリはリンクできない
異なるパーティション間でリンクできない
ls -l で調べるとリンク数が２以上になる、この時削除してもデータは残っている

シンボリック・リンク

$ ln -s リンク元 リンク先

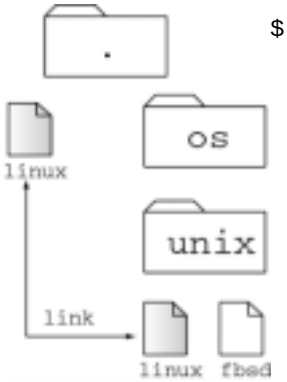
ハード・リンクの制限がない
ls -F で調べると , 後ろに @ がつく
ls -l で調べると、リンクされたファイル -> オリジナルファイル と表示される

ファイルのリンク
ln

概略　リンクはファイルのコピーとよく似た結果になりますが、**同一のファイルに、異なる２つの名前を持たせることを言います**。コピーは同じデータを異なる名前で複製しますが、リンクは、データの複製はせず、単に新しい名前だけをつけ、実際は、同じデータを指しています。

リンクは、同じデータを異なるパス名で扱いたい場合に使用できます。

左の図のように os/unix/linux をカレント・ディレクトリの linux にリンクを張るには




```
$ ln os/unix/linux .
```

又は

```
$ ln os/unix/linux linux
```

と指定します。

linux というファイルは、./linux と os/unix/linux という名前を持ちますが、これは同一のファイルで、どちらの名前を使っても同一のファイルにアクセスすることになります。上記の方法はハード・リンクで行っています。

解説　リンクには２種類の方法があります。

ハード・リンク

リンクする際には、リンク元のファイルと、リンク先のファイルを引数に指定します。

$ ln リンク元 リンク先

ハード・リンクには以下の二つの制限があります。

ディレクトリをリンクすることができない
ディスク・パーティションをまたいではいけない

ハード・リンクされたファイルを ls -l で調べると、リンク数が２以上になります。リンク数が２以上のとき、ファイルの削除を行っても、データ自信が消えることはありません。

```
ls -l linkedfile
-rwxr--r--   2 yyamazak users      6401 Apr 18 01:18 linkedfile
```

シンボリック・リンク

シンボリック・リンクはハード・リンクのような制限がありません。シンボリック・リンクを作成するには、-s を引数に指定します。

$ ln -s リンク元 リンク先

シンボリック・リンクされたファイルは、**ls -F コマンドみたとき後に @** がつきます。また、ls -l でみたときに、以下のように実際のファイルがどこにあるかを指し示します。

リンクされたファイル -> オリジナルのファイル

```
ls -l /bin
...
lrwxrwxrwx   1 root     root            3 Mar 30 12:30 vi -> vim
...
```

/bin/vi というファイルは実際は vim というファイルへのリンクであることがわかります。

上書きに関する注意点

mv, cp は既存のファイルを上書きする

-i オプションをつけることで警告を出す

```
# cp -i win sun
cp: overwrite sun?   y かそれ以外で答える
```

ファイル、ディレクトリのサイズ

du

ディレクトリ内の合計サイズを調べる
単位はキロバイト

上書きに関する注意点

mv, cp 共に、移動先、コピー先がディレクトリであった場合には、ファイルはそのディレクトリの下にコピー、移動しますが、対象がファイルであった場合は、上書きされ、オリジナルのデータは消去されます。

-i という引数をつけると、上書きを行おうとした場合、警告を出しますので、気づかないうちにファイルを上書きしてしまうことを防ぐことができます。

例）win というファイルを sun というファイルに、上書きでコピーしようとします。

```
# cp -i win sun
cp: overwrite sun? n        sun というファイルが既存であり、上書きしますか？と尋ねられます。y
                            以外の返答をすると、上書きは行われません。
$
```

ファイル、ディレクトリのサイズを調査する
du

概略　Disk Usage。ファイルやディレクトリのサイズを調べることができます。出力される単位は、キロバイト単位です。引数をつけなかった場合、カレント・ディレクトリ以下が全てのディレクトリ毎の小計が表示された後、最後に、カレント・ディレクトリ以下の合計サイズが出力されます。

例）カレント・ディレクトリ以下の際の合計を得る。

```
$ du
1       ./.gimp/scripts
1       ./.gimp/gflares
305     ./.gimp
13      ./.xcopilot
1       ./Mail2/xx
3       ./Mail2
3099    .               最後に出力されるのがカレントディレクトリ以下の合計
```

引数　ディレクトリ名　指定したディレクトリ以下の合計について集計します。

例）/bin 以下の合計サイズを求めます。

```
$ du /bin
3552    /bin            約 3.5M バイトある事が分かります。
```

-s　トータルだけを出力します。

-a　ディレクトリの小計だけではなく、各ファイルについてのサイズも表示します。

ファイルの表示

cat

ファイルをディスプレイに表示する
複数のファイルを指定すると結合して表示される

$ cat ファイル名

less

１画面に収まらないファイルを制御できる
more コマンドも同等の働きを持つ

$ less ファイル名

ファイルの表示

cat

概略 conCATination。表示したいファイルのファイル名を引数に指定するとディスプレイに表示します。複数のファイルを指定すると、全てのファイルが連続して表示されます。

引数を指定しないと、標準入力（キーボード）からデータを読み取ります。キーボードから、一行入力すると、それを単にディスプレイに表示するという振る舞いを、^D が押されるまで繰り返します。

例）fromfortune というファイルを表示する

```
$ cat fromfortune1
We don't understand the software, and sometimes we don't understand the
hardware, but we can *see* the blinking lights!
$
```

```
$ cat fromfortune1 fromfortune2
We don't understand the software, and sometimes we don't understand the
hardware, but we can *see* the blinking lights!
After a time, you may find that "having" is not so pleasing a thing,
after all, as "wanting."  It is not logical, but it is often true.
                -- Spock, "Amok Time", stardate 3372.7
```

less

概略 一画面に収まりきれないようなファイルを、ページ単位で出力するコマンドです。画面の操作は、オンライン・マニュアルと同じです。（オンライン・マニュアルが less を使用しているからです）

もし、linux 以外の UNIX で less が使えないときは、より一般的に用いられてい more コマンドを使用してみてください。操作方法はほぼ同じです。

[スペース]	一画面進みます。
[b]	一画面戻ります。
[q]	マニュアルの表示を終了します。
/ キーワード	ページ内から指定したキーワードを検索し、そこに表示を切り替えます。続けて [n」をタイプすると、同じキーワードを繰り返し検索します。[N] は逆方向（ページの先頭方向）に向かって検索をします。

例）一画面に収まらないファイル purpose.txt を less 表示します。

```
$ less purpose.txt
[Image][Image]

                    Purpose of establishing a new company

        ( This is a tenative translation of  the Japanese original)

1. Reason for establishment

Today , our sense of value is in state of confusion.  Change of economic
structure caused by rapid technical inovation greatly affects our daily
life and this leads to the result that we have various sense of value not
limited to generation gap.
This makes it difficult that organization such as company makes quick
decision. Because  recognition of current situation and direction to take
are based on our sense of value.

In this situation,  it is impossible for everyone to have mutual
understanding, and one can not make full use of one's ability without it.
Another problem is that it takes time to reach some agreement and sometimes
it is moreover not a best one.
```

スペースを押すと次のページが現れる

ファイルの削除

rm

$ rm ファイル名・・・

ディレクトリの削除を行うときは、-r をつける

$ rm -r ディレクトリ名・・・

リダイレクション機能

コマンドの出力をファイルに保存する

$ コマンド > ファイル名

コマンドの出力をファイルに追加する

$ コマンド >> ファイル名

既存ファイルへのリダイレクションを禁止する (bash)

$ set -o noclobber

ファイルの削除
rm

概略 Remove。引数に指定したファイルを削除します。ファイル名は複数指定できます。ディレクトリを削除する際は -r を引数につけて指定します。慎重にファイルの削除を行うのであれば、-i を引数に指定することで、確認付きで削除が行えます。

例）beos.backup と win というファイルを削除します。

```
$ rm beos.backup win
```

引数

-r　空のディレクトリは rmdir で削除しますが、ファイルを持っているディレクトリは rm コマンドに -r をつけて削除することができます。

例）os.backup というディレクトリにはファイルやディレクトリが入っています。それらを削除します。

```
$ rm -r os.backup
```

-i　ファイルを実際に削除する前に警告を出します。削除する場合は、y を、そうでない場合はそれ以外の入力をします。

```
$ rm -i sun
rm: remove sun? n
```

リダイレクション機能

コマンドの出力結果をファイルに保存しておく機能をリダイレクション機能といいます。リダイレクション機能は、> 記号を使って以下の書式でコマンド・ラインに記述します。

$ コマンド > ファイル名

実行されたコマンドの結果は、通常、ディスプレイに表示されますが、それを、ファイルに保存します。指定されたファイルは新規に作成されます。画面には何も表示されません。

例）date コマンドによって得ることのできる現在の時刻を datetime というファイルに保存する

```
$ date > datetime
$ cat datetime
Sun Apr 18 05:53:38 JST 1999
```

指定されたファイルが既存のものであった場合、内容は消去され上書きされます。もし、コマンドの出力を「追加」したければ、>> を使います。ファイルの内容のの末尾に追加されます。

$ コマンド >> ファイル名

例）datetime ファイルに更に現在の時刻を追加する

```
$ date >> datetime
$ cat datetime
Sun Apr 18 05:53:38 JST 1999
Sun Apr 18 05:57:39 JST 1999
```

>を使ったリダイレクション機能は既存ファイルを上書きします。上書きを禁止したい場合は、set -o noclobber を実行してください。（シェルが bash の場合 ）

```
$ set -o noclobber
```

メタキャラクタ

メタキャラクタ	意味
*	0 文字以上の任意のファイル名
?	任意のファイル名一文字
[]	括弧（ブラケット）で囲まれた文字のうちどれか一文字
~	ホーム・ディレクトリまでの絶対パス名

ファイル名補完

ファイル名の先頭文字を入力し [Esc][Esc] を押す

メタキャラクタ

長いファイル名や、特定の規則を持った複数のファイルを、一つ一つ入力していくのは大変な作業ですが、それらを簡単に表現できる機能があります。ファイル名用メタキャラクタを使用すると、ファイル名やパス名を省略して入力できます。ファイル名用メタキャラクタには次のものがあります。いずれも、コマンドライン上でのみ意味を持ちます。

メタキャラクタ	意味	例
*	0文字以上の任意のファイル名	*.html　.html で終わるファイル名 YAMA*　YAMA で始まるファイル名
?	任意のファイル名一文字	fromfortune?　fromfortune と何か一文字 ???　３文字のファイル名
[]	括弧（ブラケット）で囲まれた文字のうちどれか一文字	sun[12]　sun1 と sun2 というファイル名 sun[1-9]　1 から９まで全て
～	ホームディレクトリまでの絶対パス名	～/.profile　ホームディレクトリの下の .profile

例１）ファイル名が .html で終わっている全てのファイルを docs というディレクトリに移動

```
$ mv *.html docs
```

例２）X1, X2, X3, X4, X5, X6 というファイルを削除

```
$ rm X[1-6]
```

ファイル名補完機能

コマンドラインから、ファイル名を入力する際、先頭の何文字かだけを入力し、[Esc] を２回押すと、その文字を含むファイル名に補完してくれます。（シェル bash の機能）

もし、同一の名前を持つファイルがあったら、補完可能な部分まで表示されるので、残りはキーボードから入力してください。

例）非常に長いファイル名、thisfilenameistoolong が合った場合にファイル名補完機能を使う

```
$ cat thi[Esc][Esc]        補完される
$ cat thisfilenameistoolong
HELLO
```

第６章 ファイルとディレクトリの操作 演習問題

１） UNIX ファイル・システムの起点になるディレクトリは何ですか？

２） ~~ディレクトリが、２つ以上の親ディレクトリを持つことはありますか？~~

３） ~~カレント・ディレクトリを調べて下さい。[実習]~~

４） /etc/profile をホーム・ディレクトリにコピーしてください。[実習]

５） コピーしてきた profile というファイルの内容を表示してください。[実習]

６） mydocument, publich_html, log, settings というディレクトリを作成してください。「実習]

７） コピーしてきた profile というファイルを setting というディレクトリに移動してください。[実習]

８） settings というディレクトリを mydocument というディレクトリに移動してください。[実習]

９） settings というディレクトリを settings.backup という名前でコピーしてください。[実習]

１０） mydocument/settings/profile というファイルを、ホーム・ディレクトリの下に .profile という名前で、シンボリック・リンクを作ってください。[実習]

１１） シンボリック・リンクした .profile の詳細な情報を ls コマンドで調べてください。[実習]

１２） 現在の時刻が収められているファイルを、log というディレクトリの下に finishwork という名前で作成してください。[実習]

１３） log の下の finishwork を表示して下さい。[実習]

第 7 章 vi エディタ

学習目標

vi は UNIX の標準のエディタとして用いられています。vi エディタを使用して、ファイルを作成、編集する方法を学習します。

vi エディタの起動

$ vi practice

```
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"practice" [New File]
```

vi エディタ

vi は UNIX で標準的に用意されている、テキスト編集用のプログラム（エディタ）です。vi エディタを用いると、ファイルの作成、編集を行うことができます。

linux では、Bram Moolenaar 氏によって作られた vim というプログラムが vi のかわりに使われます。vim は多国言語に対応した vi です。ここでは、vim を vi と同義として解説していきます。

vi エディタの起動

vi エディタを起動するときに、作成、編集するファイル名を指定する事ができます。省略すると、ファイルを保存するときにファイル名を指定します。

$ vi ファイル名

存在しないフィルを指定した場合は新しいファイルを、存在するファイルの場合は、そのファイルを開くことになります。

例）新規のファイル practice を vi で開きます。

$ **vi practice**

practice は新規のファイルですので、ファイルは空です。画面に vi のエディット画面が現われます。

～は空白行を表わしています。

最下行には、vi からの情報が表示されます。practice が新規のファイルであることが分かります。

"practice" [New File]

モードの切り替え

コマンド・モードから入力モードへは、a i o A I O のいずれか
入力モードからコマンド・モードへは [Esc]

```
Nice to meet you.
My name is Yuki Yamazaki
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT --
```

モードの切り替え

vi には、コマンドを実行するモードと、文字を入力するモードの２種類があります。利用者は現在のモードがどちらであるかを把握しながら使用しなくてはなりません。

vi 起動時は、コマンドモードです。

コマンドモード

vi のコマンドを入力します。vi のコマンドは、アルファベット一文字又は二文字です。コマンドモードでは次のようなことが行えます。

機能	キー
上下左右へのカーソルの移動	↑↓←→ 又は ｋ ｊ ｈ ｌ
画面のスクロール (下方向、上方向)	^F ^B
入力モードへの切り替え	ａ ｉ ｏ Ａ Ｉ Ｏ
行ののコピーとペースト	yy p
削除 （文字、単語、行）	x dw dd
一つ前の編集の取り消し	u

入力モード

コマンド・モードからａ ｉ ｏ Ａ Ｉ Ｏ のいずれかが押されると入力モードに切り替わります。ステータス行に

--INSERT--

と表示されることを確認し、文字を入力していきます。コマンドモードにもどるには [Esc] を押します。

例）practice ファイルを対象に、コマンド・モードから入力モードに切り替え、文字を入力した後、コマンド・モードでいくつかのコマンドを確認してみます。

i をタイプします。

入力モードに切り替わったのを確認し、以下の２行を入力します。

```
Nice to meet you.
My name is XXXXXXXXXX
```

入力し終えたら、[Esc] を押して、コマンド・モードに切り替えます。

カーソルキー、または、kjhl を使用して、カーソルが移動するのを確認します。

削除

文字単位 x

```
Nice to meet you
My name is Yuki Yamazaki
```

x をタイプする

```
Nice to meet you
My name is uki Yamazaki
```

単語単位 dw

```
Nice to meet you
My name is uki Yamazaki
```

cw をタイプする

```
Nice to meet you
My is uki Yamazaki
```

行単位 dd

```
Nice to meet you
My is uki Yamazaki
```

dd をタイプする

```
My is uki Yamazaki
```

取り消し u

u をタイプする

```
Nice to meet you
My is uki Yamazaki
```

削除

文章を編集すためには、文字を削除しなければいけない場面があります。vi では文字、単語、行の単位で文字を削除することができます。コマンド・モードから削除コマンドを入力してください。

文字の削除 x

文字を削除するときには、削除した文字まで、カーソルを持っていき x をタイプします。文字は、削除されます。x の前に削除したい文字の数を入力すると、カーソルから左に、指定した文字数削除します。

例）practice ファイルの２行目のの名前 (Y 以降の１３文字）を削除します。

カーソルを消したい文字ところまで持っていき、x を押します。もし、消したい文字数がわかるようでしたら、数字を入力後 x を押して一度に消せます。

My Name is **Y**uki Yamazaki

カーソルを **Y** まで持っていき、**13x** と続けてタイプする

My Name is

に変更されます。

単語の削除 dw

単語の先頭にカーソルを持っていき、cw とタイプすると、単語が削除されます。単語は、スペースや、タブ、カンマ、ピリオドなどで区切られています。

例）２行目の name という単語を削除する

カーソルを name の先頭（n) まで移動させ、dw とタイプします。

行の削除 dd

削除したい行にカーソルを持っていき、dd とタイプすると、行が削除されます。dd の前に数字を指定すると、カーソルがある行から指定した数の行が削除されます。

例）１行目を削除する。

カーソルを１行目に移動します。

dd とタイプします。

行が削除されます。

直前の編集の取り消し u

直前に行った、削除などの編集作業を取り消して元に戻します。u は undo を意味します。

例）直前に行った行の削除を取り消す

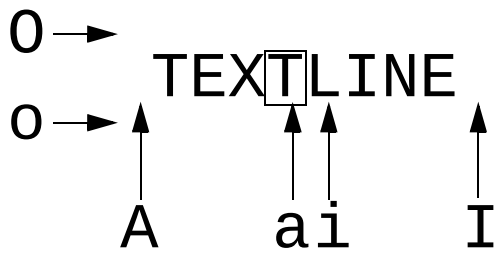
u

削除した行が、元に戻ります。

挿入

コマンドモードから入力モードへ切り替わる
コマンドによって、挿入開始位置が違う

i	カーソルのある位置から
a	カーソルの次の一文字目から
o	次の行の先頭から
I	カレント行の先頭から
A	カレント行の末尾から
O	前の行の先頭から



挿入

文字を挿入するコマンドはいくつかあります。次のコマンドが入力されると、**コマンドモードから入力モードへ切り替わります**。現在のカーソル位置から、どこに対して挿入するかがそれぞれ異なっています。必要に応じて使い分けます。

カーソルのある位置から挿入 i (insert)

i はカーソルがある位置に対して、文字を挿入したい場合に使われます。

00**0**00
コマンド **i**ccc
結果
00**ccc**000

カーソルの次の一文字目から挿入 a (append)

a はカーソルの次の一文字目から挿入が開始されます。

00**0**00
コマンド **a**ccc
結果
000**ccc**00

新たに行を作り挿入を開始する o (open)

o は現在カーソルがある次の行に、新たに空行を設け、その先頭から挿入が開始されます。

00**0**00
コマンド **o**ccc
結果
0000
ccc

カレント行の先頭から挿入を開始する I

I は、カレント行（現在カーソルがある行）の先頭にカーソルを移動し、そこから挿入を開始します。

00**0**00
コマンド **I**ccc
結果
ccc00000

カレント行の末尾から挿入を開始する A

A は、カレント行の末尾にカーソルを移動し、そこから挿入を開始します。

00**0**00
コマンド **A**ccc
結果
00000**ccc**

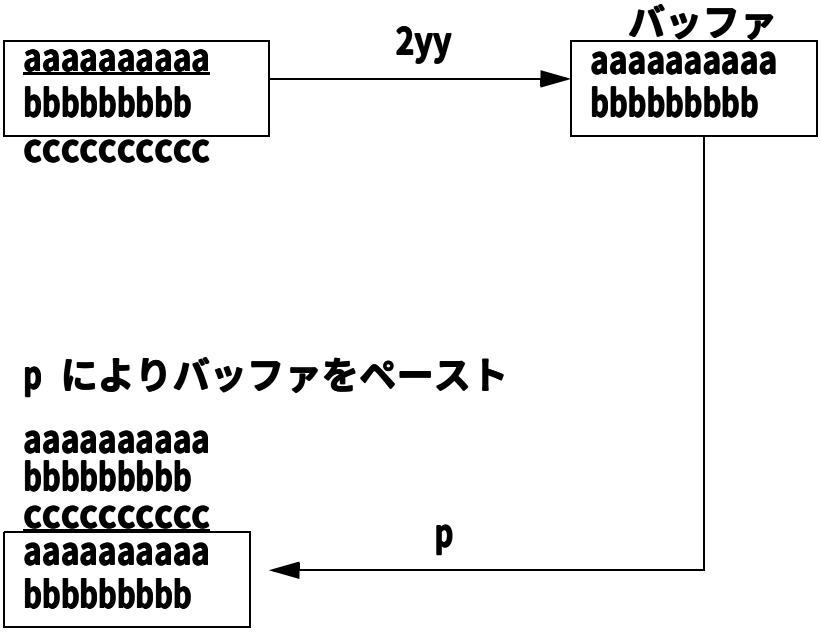
カレント行の上に新たに行を作り挿入を開始する O

O は、カレント行の上に新たに行を作り、その先頭のから挿入を開始します。

00**0**00
コマンド **O**ccc
結果
ccc
00000

コピー・ペースト

yy により行をバッファにコピー
複数行のコピーは 行数 yy と指定する



コピー・ペースト

テキスト内で、同一の内容を持つ行をコピーします。

行のコピー yy

コピーしたい行にカーソルを移動し、yy をタイプします。画面上には変化はありませんが、カーソルの行がバッファー（メモリ）にコピーされます。

もし、複数行をコピーしたいのであれば、その行の先頭にカーソルを移動させ、行数を指定した後、yy をタイプします。3 行コピーしたいのであれば、3yy とタイプします。

```
aaaaaaaaaaaaaaa
bbbbbbbbbbbbbbb   2yy
ccccccccccccccc
```

カーソルが、１行目（ aa... ）にあるときに、2yy とすると、１行目と２行目がバッファに保存されます。

dd による行の削除も、yy と同じようにバッファに保存されます。

ペースト p

バッファ内のテキストを、カレント行の下に呼び出します。

```
aaaaaaaaaaaaaaa
bbbbbbbbbbbbbbb
ccccccccccccccc
aaaaaaaaaaaaaaa
bbbbbbbbbbbbbbb   p
```

カーソルが、３行目（ cc... ）にある時、p をタイプすると、カレント行の下にバッファ内のテキストがペーストされます。

保存

:w
又は
:w ファイル名

終了

:q
又は
:q! （強制終了）

保存終了

ZZ
又は
:wq

保存 :w

vi で編集したファイルは、保存作業を行わなければ、記録されません。ファイルの保存は、**:w** と入力します。

ファイルを開いたときと異なる名前で保存する場合は

:w ファイル名

と指定します。

正しく保存されると

```
"filename" [New File] 1 line, 6 characters written
```

終了 :q

vi を終了するときは :q と入力します。もし、ファイルに変更を加え、かつ、それが保存されていなければ、安全のため（せっかく編集した内容を失ってしまわないよう)、

```
No write since last change (use ! to override)
```

という警告を出し、終了できません。

必要であれば保存した後、再度終了コマンドを入力してください。

もし、保存したくなければ

```
:q!
```

と入力すると、強制的に終了できます。

保存終了 ZZ

保存と終了を同時行うには、**ZZ** と入力するか、または、**:wq** と入力します。

大きなファイルの編集に便利なコマンド

^G	ファイルの情報の取得
^F ^B	画面のスクロール（下方向、上方向）
/	文字の検索
n N	検索の繰り返し（下方向、上方向）
G	指定した行に移動（ EX．100 行目に移動 100G ）

大きなファイルの編集に便利なコマンド

カーソルの移動、挿入、削除、保存の操作はファイル操作の最低限の機能ですが、特に大きなサイズのファイルを操作するために便利なコマンドのいくつかをご紹介します。

ファイルの情報の取得 ^G

^G を押すと、現在編集しているファイルの、ファイル名、カレント行数、全体の行数、表示しているファイルの割合、左から数えた文字の位置を調べることができます。

例）lgpl.txt を vi で開き、ファイルの情報を調べます。

$ vi lgpl.txt

^G をタイプする

```
"lgpl.txt" line 1 of 481 --0%-- col 5-19
```

lgpl.txt は全体で 481 行あり、そのうち 1 行目にカーソルがある。問いう事が分かります。col 以降の表示は、カーソルが左から何文字目にあるかを表わし、タブがある場合、タブを 1 文字とした数えたのが - の前の数字、タブを 8 文字として数えたのが後ろの数字です。

画面のスクロール（上方向、下方向）^F ^B

長い文章を編集する際に、一画面づつ切り替えることができます。

^F　　下方向の 1 画面の表示

^B　　上方向に一画面の表示

文字の検索 /

編集対象になっている文章中から特定の文字を探し出し、カーソルそこに移動させる機能です。/ をタイプすると、ステータス行にカーソルが移動しますので、検索した文字を入力してください。

例）lgpl.txt の中から、free という文字を見つけ出します。

/free

カーソルが free という文字にジャンプします。

検索の繰り返し n N

/ で行った検索を繰り返します。n は、ファイルの下方向に、N はファイルの上方向に検索を繰り返します。

例）先ほど行った free という検索を繰り返します。

nnnnnnnn

指定行にカーソルを移動 G

G の前に、移動したい行数を指定すると、そこにカーソルを移動します。1G はファイルの先頭に移動します。行番号が指定されず G だけが入力された場合は、ファイルの最終行に移動します。

例)150 行目に移動

150G

編集のために便利な機能

文字の一括置き換え

:%s/old/new/g
ファイル中の全ての old を new に置き換える

:%s/free/proprietary/g
ファイル中の全ての free を proprietary に置き換える

他のファイルからの読み込み

:r ファイル名

コマンドの実行結果の取り込み

:r! コマンド

編集のために便利な機能

ファイルを編集していく上で便利な機能を紹介していきます。

文字の一括置き換え :%g/old/new/g

ファイル中の指定した全ての文字を、指定した文字に一括で置き換えます。どの文字を置き換えるかの指定は、old にします。何という文字に置き換えるかを new に指定します。

例）lpgl.txt の中に存在する全ての free という文字を proprietary に置き換えます。

```
:%s/free/proprietary/g
27 substitutions on 22 lines
```

２７個所、２２行で置き換えを行ったという結果が出ます。

他のファイルからデータを読み込む :r

ファイルを編集中、他のファイルからデータを読み込みたい場合があります。:r ファイル名 とすると、カレント行の下に、指定したファイルが取り込まれます。

例）現在編集している lgpl.txt の末尾に、manifesto.txt というファイルを読み込みます。

ファイルの末尾にということなので、G コマンド、ファイルの末尾に移動した後、読み込みます。

```
G
:r manifesto.txt
```

コマンドの実行結果をファイルに読み込む :r!

UNIX のコマンドを実行した結果を、現在編集しているテキストに読み込みます。実行結果は、カーソルのある行の後ろに読み込まれてきます。

例）現在編集しているファイルの末尾に、今月のカレンダーと現在の時刻を取り込みます。。

ファイルの末尾に移動した後、cal コマンドと date コマンドの結果を読み込みます。

```
G
:r! cal;date

...
Copyright (C) 1985, 1993 Free Software Foundation, Inc., 59 Temple
Place -
Suite 330, Boston, MA 02111, USA
     April 1999
Su Mo Tu We Th Fr Sa
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30

Mon Apr 19 01:31:59 JST 1999
9 more lines
```

環境の設定

設定
:set パラメーター

解除
:set no パラメーター

表示
:set all

タブ位置の変更（初期値 ８文字）
:set tabstop=4

段落の維持
:set autoindent

括弧の対応を表示
:set showmatch

大文字小文字を問わない検索
:set ignorecase

行番号の表示
:set number

環境の設定

vi は起動した時点では、有効になっていないいくつかの便利な機能を持っています。好みに応じて、機能を有効にしたり、初期設定を変更できます

設定を変更するには
:set パラメーター

設定を解除するには
:set no パラメーター

現在の設定を確認するには
:set all

タブの停止位置の変更 tabstop （初期値 tabstop=8）

タブは通常８文字のスペースが取られますが、これを変更することができます。

例）タブを４文字に変更する

```
:set tabstop=4
```

段落を維持する autoindent （初期値 noautoindent）

通常、文章を入力し、改行したときに、カーソルは次の行の先頭に移動しますが、autoindent を設定すると、現在の行の字下げの位置と同じ位置に、次の行のカーソルが配置されます。カーソルを先頭に持ってきたいときは、^D を使い調整します。

括弧の対応を表示する showmatch (初期値 noshowmatch)

括弧の多いテキスト（プログラムなど）を作成している際、対応を自動的にチェックしてくれます。閉じた括弧を入力すると、それに対応する開いた括弧に瞬間的にカーソルがジャンプし、対応を教えてくれます。もし、対応する括弧が見つからなかった場合は、ピという音を鳴らします。

また、テキストを入力した後に対応を調べたいときは、括弧にカーソルを移動し、% コマンド入力すると対応する括弧を指し示します。（これは、このパラメーターを設定しなくても使えるコマンドです）

大文字小文字を無視して検索を行う ignorecase （初期値 noignorecase ）

ic (noic) と省略して指定することもできます。/ で検索を行う際に、大文字小文字を問わない検索をするようになります。

行番号を表示する number (初期値 nonumber)

行番号を表示しながら、編集したいときは :set number を実行します。行番号を消したいときは :set nonumber と指定します。:set nu (:set nonu) というように省略して指定することもできます。

vi の環境設定用のファイル .exrc

ホーム・ディレクトリに作成

```
$ vi .exrc

set number
set tabstop=4
set autoindent
set showmatch
```

次回 vi を起動すると設定が有効にっている

vi の環境設定用のファイル .exrc

vi に設定した環境設定は、その vi を終了するとなくなります。常に同じ設定で使いたいときは、環境設定用のファイルを作成し、そこに環境設定を記述しておきます。

ホーム・ディレクトリに .exrc という名前で、ファイルを作成し、その中に、vi の環境設定用のコマンドを書いておくと、vi を起動したとき常にその設定で起動するようになります。

例）.exec ファイルを作成する

```
$ vi .exec
set tabstop=4
set number
```

上記の設定が有効な状態で、vi が起動するようになります。

第 7 章 vi エディタ 演習問題

1） 次の内容を持つファイルを fss.txt という名前で作成してください。これは GNU の創立者、Richard Stallman 氏によって作成された、Free Software Song という歌の歌詞です。[実習]

```
Join us now and share the software;
You’ll be free, hackers, you’ll be free.

Join us now and share the software;
You’ll be free, hackers, you’ll be free.

Hoarders may get piles of money,
That is true, hackers, that is true.
But they cannot help their neighbors;
That’s not good, hackers, that’s not good.

When we have enough free software
At our call, hackers, at our call,
We’ll throw out those dirty licenses
Ever more, hackers, ever more.

Join us now and share the software;
You’ll be free, hackers, you’ll be free.

Join us now and share the software;
You’ll be free, hackers, you’ll be free.
```

2） fss.txt の後ろに、manifesto.txt というファイルを追加してください。[実習]

3） ホーム・ディレクトリに、vi　起動時に自動的に行番号がつくような環境設定用のファイルを作成して下さい。[実習]

第８章 ファイルのパーミション

学習目標

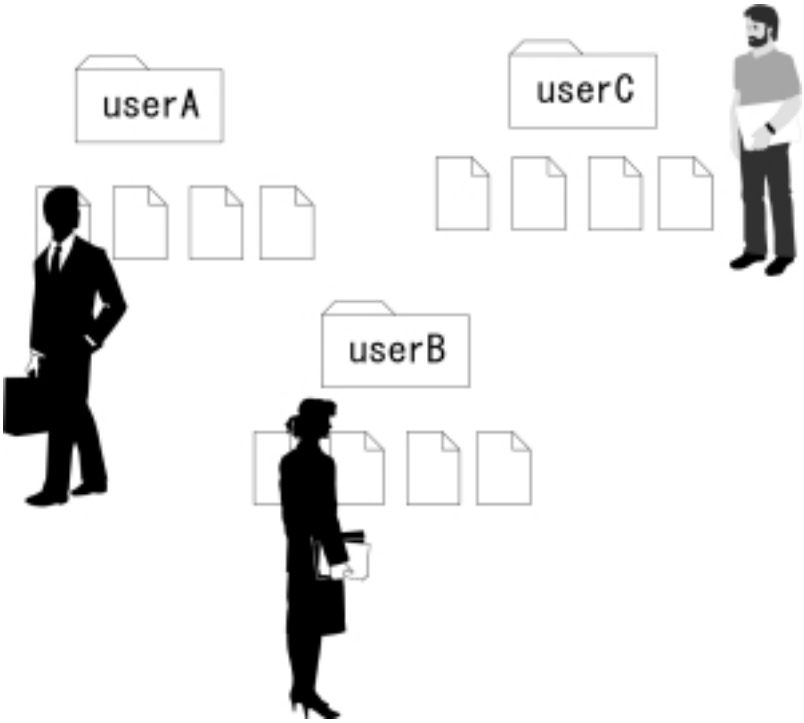
ファイルを他人にアクセスされたくない場合の、ファイルの保護方法について学びます。

ファイルの保護

ファイルは誰かの持ち物である（所有権）

所有者は全権限を持つ

パーミションは所有者のみが変更できる



ファイルの保護

UNIX システムは複数の人が利用することを前提に作られたオペレーティング・システムです。同一のファイルシステムを複数の人が共有しています。自分の持ち物に関して、他人に勝手にアクセスされることがないよう、保護機能を持っています。

ファイルの所有者

UNIX システムの全てのファイル（ディレクトリ）には、所有権が設定されており、誰の持ち物であるかが決められています。ファイルの所有者はそのファイルに対してあらゆる操作が可能です。

各ファイルの所有者は ls -l コマンドで調べることができます。出力されたリストの第３番目のフィールドが所有者を表わします。

```
$ ls -l
total 2
drwxr-xr-x   5 root     root         1024 Apr 26 09:36 GNUstep/
-rw-r--r--   1 root     root          485 Apr 26 09:37 Xrootenv.0
```

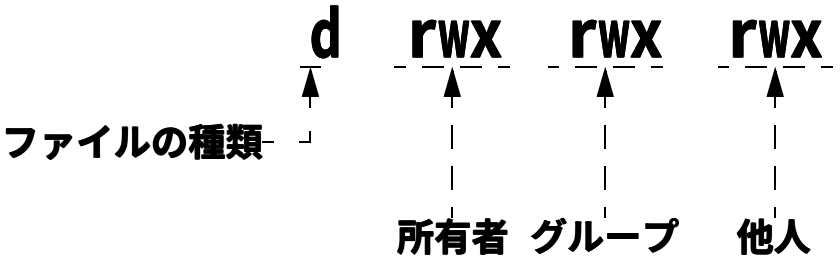
パーミション

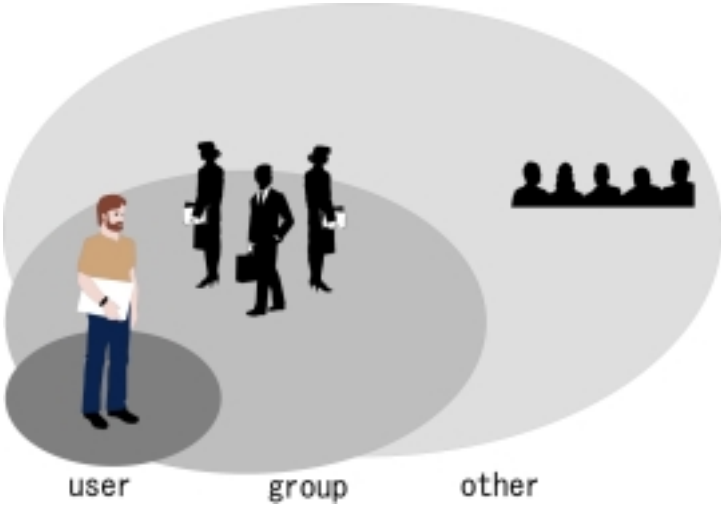
ファイルの所有者は、「誰に」「どのような操作を」許可するかを、ファイル毎に設定することができます。ファイルに設定されている、許可情報をファイルのパーミションと呼びます。システム管理者は特別に、どのファイルでもアクセスできます。

各ファイルに設定されたパーミションの情報は ls -l で調べることができます。出力されたリストの第１番目の１０桁の文字で、パーミション情報を表わします。

```
$ ls -l
total 2
drwxr-xr-x   5 root     root         1024 Apr 26 09:36 GNUstep/
-rw-r--r--   1 root     root          485 Apr 26 09:37 Xrootenv.0
```

パーミションの読み方





パーミションの読み方

１０桁からなるファイルのパーミションは次のような意味を持ちます。

1桁目 ファイルの種類

drwxrwxrwx

一桁目はファイルの種類を表わし、パーミションそのものではありません。

ファイルの種類

一桁目の表記	意味
-	通常ファイル
d	ディレクトリ
l	シンボリックリンク

2桁目 ~ 4桁目 所有者に対するパーミション

d**rwx**rwxrwx

２桁目から４桁目は、「所有者（自分自身)」に対するパーミションを表わします。

5桁目 ~ 7桁目 グループに対するパーミション

drwx**rwx**rwx

５桁目から７桁目は、システム管理者がユーザー作成時に決めた、「グループ」に対するパーミションを表わします。

自分がどのグループに属しているかは groups というコマンドで調べることができます。一般ユーザーは自分の属してるグループを変更することはできません。

```
$ groups
school
```

8桁目 ~ １０桁目 その他の人に対するパーミション

drwxrwx**rwx**

８桁目から１０桁目はその他の人に対するパーミションを表わします

パーミションの種類

r 読み込み許可

ファイルを読み込める cat, more, cp 等
ディレクトリのリストを表示できる ls 等

w 書き込み許可

ファイルを変更できる vi 等
ディレクトリの構成を変更できる mkdir, rmdir, rm 等

x 実行許可

コマンドを実行できる
ディレクトリに移動できる cd 等

- 許可なし

許可がないファイルにアクセスしたとき

```
Permission denied
```

パーミションの種類

どのような操作を許可するかは、rwx の３種類で表わされます。- は許可がないことを表わします。

読み込みの許可 r

ファイルを読むことができるようになります。この許可がなければ、cat コマンドや、more コマンドでファイルの内容を調べることができなくなります。また、ファイルをコピーする際のコピー元のファイルには、読み込み許可が必要になります。閲覧されたくない場合は、この許可を削除しておきます。

ディレクトリに対する読み込みの許可とは、そのディレクトリ内に管理されているファイルの情報を ls コマンドなどで、調べることができることを意味します。

書き込み許可 w

ファイルを変更できるようになります。この許可がなければ、vi コマンドなどで、内容を編集することができなくなります。書き換えられたくないファイルは、この許可を削除しておきます。

ディレクトリに対する書き込みの許可とは、そのディレクトリ内に管理されているファイルの構成を mkdir, rmdir, rm コマンド、または新たにファイルを作成するコマンドなどで、変更ができることを意味します。

ファイルの削除を禁じたい場合は、ファイル自体の書き込み許可を削除するのではなく、そのファイルが置かれているディレクトリの書き込み許可を削除します。

実行許可 x

アプリケーション、コマンド、シェルスクリプトなどのファイルを実行することができます。通常のテキストや、データが収められているファイルでは、この実行許可はあまり意味がありません。プログラムの場合は、実行許可がなければ、実行することができません。

ディレクトリの実行許可は、cd コマンド等で、そのディレクトリに移動できることを表わします。実行許可はディレクトリと、ファイルで意味が異なっているので注意します。

許可がないときの反応

ファイルに操作を行いパーミションがないときは、permision denied という表示が出ます。

```
$ cat a
cat: a: Permission denied
```

パーミションの変更

chmod コマンド

chmod モード ファイル名

モードの指定

シンボリック・モード
下記の文字の組み合わせで表現

誰に？	どうする？	何を？
u | g | o | a	+ | - | =	r | w | x
u user g group o other a all	+ 与える - 削除する = 全て削除する	r 読み込み w 書き込み x 実行

オクタル・モード
３桁の８進数で表現

r	w	x
4	2	1

計算手順

1) ３桁に分解

rwx　　r-x　　r--

2) 各桁の重みを計算

rwx	r-x	r--
4+2+1	4+1	4
=7	=5	=4

3)chmod コマンドのモードに指定

chmod 754 ファイル

パーミションの変更

ファイルのパーミションを変更できるは、ファイルの所有者か、システム管理者だけです。chmod(change mode\e)コマンドを用います。-R オプションをつけ、ディレクトリを指定すると、そのディレクトリ異か全てのファイルのパーミションを変更できます。

chmod [-R] ***モード　ファイル名又はディレクトリ***

モード

パーミションの変更を記号で表わします。表記の仕方は２種類あります。

シンボリック・モード

誰に対する指定なのかを、u, g, o, a で表わします。

u 所有者 (owner)
g グループ (group)
o 他人 (other)
a 上記全て (all)

許可を与えるのか、削除するのかを + , -, = で表わします。

+ 許可を与える
- 許可を削除する
= 全てのパーミションを削除

許可の種類を r, w, x で表わします。

r 読み込み (readable)
w 書き込み (writable)
x 実行 (eXecutable)

例)

ファイル a に対し、グループとその他の人に読み込みと書き込み許可を与えます。

```
$ chmod go+rw a
```

オクタル・モード

目的のパーミションの状態を、８進数で表現して指定します。

計算方法

最初の１桁目を除いた、残り９桁を対象とし、許可が与えられている部分を１、与えられていない部分を０として、９桁の２進数を得ます。その２進数を３桁の８進数に変換し、モードに指定します。

```
rwx r-x r--
111 101 100
  7   5   4

$chmod 754 file_name
```

r は 4, w は 2, x は 1 として、３桁毎に足し算していくと簡単です。

デフォルトのパーミション

ファイル rwxr--r--
ディレクトリ rwxr-xr-x

マスク値 022

マスク値の変更

umask [-S] マスク値

-S オプションでシンボリックモードを用いた表現ができる

e.g.

```
umask 077

ファイルの場　666
マスク値　　077
パーミション　600
```

umask -S u=rwx,g=rx,o=
グループに対して w を削除、他人に対して全てを削除する

デフォルトのパーミション

新規にファイルやディレクトリを作成したときのパーミションは、ファイルであれば、**rw-r--r--** （所有者は読み込みと書き込みが行え、その他の人は読み込むことしかできない）という設定なり、ディレクトリは、**rwxr-xr-x**（所有者は全操作が行え、その他の人は書き込みができない）という設定になります。

デフォルトのパーミションを変更する際は umask というコマンドを用います。

デフォルトのパーミションの変更

umask [-S] **マスク値**

引数を省略すると現在のマスク値を得ることができます。

初期状態でシステムが設定するマスク値は **022** です。

マスク値は３桁の８進数で指定し、主にファイルの場合は ６６６を、ディレクトリの場合は７７７をベースにし、マスク値との引き算をした値（論理和）が、各コマンド (たとえば、mkdir や、vi) がファイルを作るときの初期パーミションとなります。

-S オプションをつけるとシンボリック・モードを使用して、マスク値を指定することができます。

シンボリックモードでの、デフォルトのマスク値の表現は **u=rwx,g=rx,o=rx** です。８進数でのマスク値の設定と異なり、与えたいパーミションを指定します。

例１）

新規ファイルを作成するときの計算。

ファイルのベースの値 **6 6 6**

umask の値 **0 2 2**

作成されるパーミション **6 4 4**

例２）

グループと他人に対し、パーミションを与えないマスク値を設定。

```
$ umask 077
```

例３）

-S オプションを使用して上記の例を表現。

```
$ umask -S u=rwx,g=,o=
```

第 8 章 ファイルのパーミション 演習問題

1） ファイルのパーミションを変更できるのは誰ですか？

_____

2） 次のパーミションをオクタルモードで表現してください。

r--r--r _____

rwxr-x-- _____

r-xr-xr-x _____

3） 次の 8 進数を、パーミションの表記で表現してください。

644 _____

777 _____

511 _____

4） 所有者に書き込み件を与える為の指定を、シンボリックモードで表現してください。

_____

5） グループと他人から書き込み許可を削除する指定を、シンボリックモードで表現してください。

_____

6） /etc/passwd というファイルのパーミションはどのように設定されているかを確認してください。所有者以外のユーザーはそのファイルの編集が行えますか？

_____

7） /bin のディレクトリのパーミションはどのように指定されているか確認してください。そのディレクトリ内のファイルを削除できますか？

_____

8） あなたのホームディレクトリに自分以外のユーザーがアクセスできないよう、パーミションを変更してください。[実習]

_____

9） ホーム・ディレクトリにあるファイルを削除されないようにするには、どのファイル又はディレクトリのパーミションの操作が必要となりますか？

_____

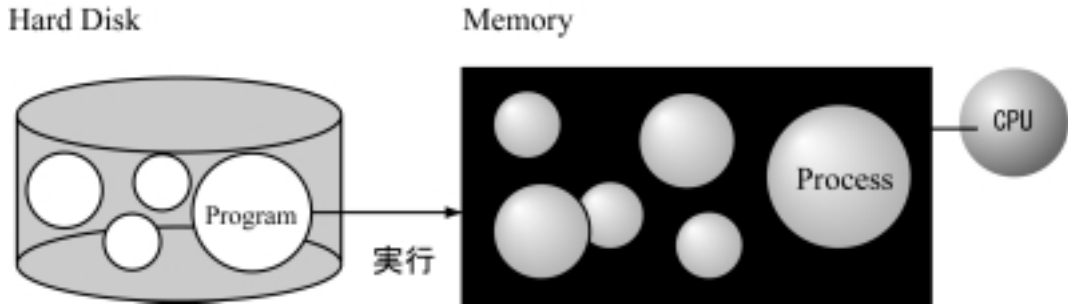
第９章 プロセスの実行と監視

学習目標

UNIX がプログラムを実行するときの単位であるプロセスの概念と、それを制御する為のいくつかのコマンドをみていきます。

プロセス

プログラムが稼動している状態



プロセスとは？

プロセスとは、プログラムが実行されている状態のことを言います。

ハードディスクに保存されている状態のコマンド、アプリケーション、ユーティリティは、単に保存されているだけで、何も処理を行っていません。ここに保存されているプログラムが実行され、処理を行っている状態のことをプロセスといいます。

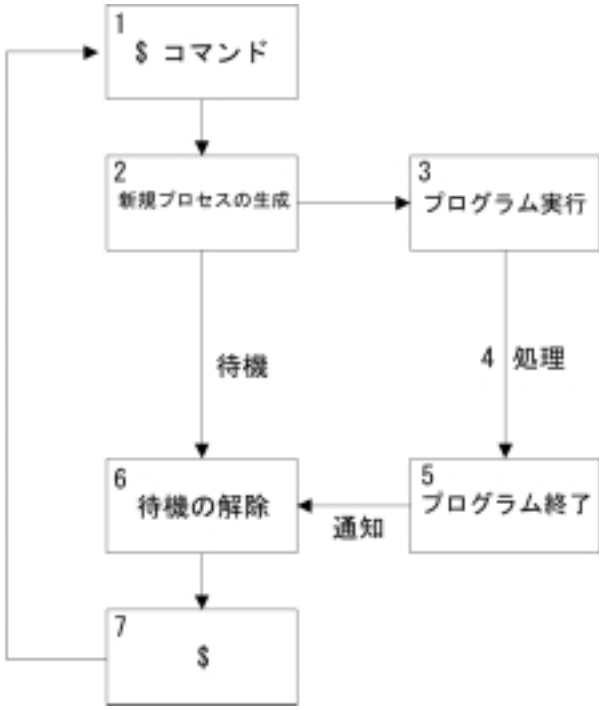
UNIX では複数のプロセスを同時に実行することが可能ですが、その単位は、通常１つのコマンドです。

ハードディスク上の UNIX コマンドは、恒常的にそこに存在しますが、それが実行された状態のプロセスは、必要なときに起動し、処理が終わると消えてなくなります。ですから、プロセスは誕生し、処理し、消滅します。

具体的なプロセスの一生は、プログラムの実行要求により、ハード・ディスクから、メモリに読み込まれ活性化されます。次にプログラムを実行し、処理を終えると、メモリから開放されます。

プロセスには、数日間、生き続けるものあれば、一秒以下で終了するものもあります。

プロセス実行の仕組み



プロセスの実行

プロセスの実行とは、今まで行ってきた「コマンドの実行」と同義です。プロセスはコマンド・ラインから実行されたり、また、UNIX システムが起動する際に、システム管理者によってあらかじめ設定されたものが起動していたりします。

プロセス実行の仕組み

プロセスは、次のような仕組みで起動、実行、終了が行われます。

1. コマンド・ラインからプログラムの実行が依頼されると、シェルはそのプログラムをハードディスクの中から検索し、メモリにそのプログラムを読み込みます。

2. コマンド・ラインから実行されたプログラムが、バック・グランドでの実行（コマンド・ラインが & で終わっている）でなければ、シェルは実行されたプログラムの終了を待ちます。

3. プログラムは、シェルと別のプロセスとして実行されます。

4. カーネルはメモリに読み込まれたプログラムを実行します。

5. プログラムが処理を終えると、そのプロセスは終了すると同時に、シェルに対し終了の通知を行います。

6. プログラムの終了通知を受け取ったシェルは、待つことをやめ、新たにコマンド・プロンプトを出し、ユーザーからの次の処理の依頼を待ちます。

プロセスのバリエーション（デーモン・プロセス）

今までみてきた、コマンド (cat, ls, vi, netscape 等）は、ユーザーの要求によって起動し、処理が終えると自動的に消滅する、比較的短い生存期間のプロセスでしたが、プロセスの中には、一度起動しておくと、明示的に停止するまで、メモリ中に存在しつづけ、別のプログラムからの処理の依頼を待ち続けるものもあります。

このようなプロセスを一般的にデーモン・プロセス (daemon process) と呼びます。デーモン・プロセスが他のプログラムと異なる点は、他のプログラムからの処理の依頼をバック・グランドで待ち続けているということと、画面に対し直接出力を行ったり、ユーザーからの処理要求は直接は処理しないという点です。一度たちあげておけば、管理者がわざわざ実行しなくても、自動的にいろいろなことをしてくれるので非常に便利です。

UNIX システムでも、たくさんのデーモン・プロセスが稼動し、システムの機能を豊富にしてくれます。そのような性格を持つプロセスには次のようなものがあります。

メールサーバー電子メールの配送と受け取りを行います。

cron　プログラム実行スケジューラー。時間ベースで定期的にプログラムを実行します。

ウェブサーバーブラウザから処理の要求があると、ファイルを返送する。

プリンタデーモン印刷の依頼があると、データをプリンタに送る

ftpd　ファイル転送プログラム。

プロセスの稼動状況の監視

プロセスには一意の番号 (PID) がつく

ps コマンド

PID	プロセス ID
TTY	端末
STAT	プロセスの状態
TIME	CPU 使用時間
COMMAND	コマンド名

```
$ ps a
```

自分以外のプロセスも表示
デーモン・プロセスは表示しない

```
$ ps ax
```

x はデーモン・プロセスも表示する
a と共に使うとシステム上の全てのプロセスを表示する

プロセスの稼動状況の監視

UNIX は同時に複数の仕事を行うことができます。ですから、画面上に表示されているプログラム以外にも実行されているものもあります。また、UNIX は同時に複数の人が使用している場合があります。それらの活動を監視するには ps(Process Status) コマンドを持ちます。

ps

概略 プロセスの状況を表示します。

```
$ ps
  PID TTY STAT TIME COMMAND
14545  p2 S    0:00 -bash
15029  p2 R    0:00 ps
```

現在し要しているシェルから起動されているプロセスの情報が出力されます。-bash というコマンドと、ps というコマンドが表示されました。

表示内容は、１行に１プロセスが表示さます。

PIDプロセス ID システムがプロセスにつけた一意の番号です。同じ時間帯において同一のプロセス番号を持つほかのプロセスはありません。

TTYそのプロセスを実行した端末。

STATプロセスの状態を表わします。S は休止状態 (Sleep)、R は稼動状態 (Run) を表わします。

TIMECPU 使用時間の累計。

COMMANDコマンド名を表わします。

引数 a　自分以外のプロセスも全て表示します。ここではデーモンプロセスは表示されません。ですから、自分以外であっても、システム上の全てのプロセスが表示されているわけではありません。

例）自分以外のプロセスも表示する

```
$ ps a
...
13540   1 S    0:00 /usr/GNUstep/Apps/WPrefs.app/WPrefs
13543   1 S    0:00 xterm
13544  p0 S    0:00 bash
13843  p0 S    0:00 kinput2 -canna
...
```

x　デーモン・プロセスも含めて表示します。a と組み合わせて使うとシステム上の全てのプロセスを表示することができます。

例）システム上の全てのプロセスを表示します。

```
$ ps ax
...
14388  p1 S    0:00 telnet gw14480  p2 S    0:00 -bash
  183  ?  S    0:00 portmap
  348  ?  S    0:00 /usr/jp/canna/bin/cannaserver -syslog
  165  ?  S    0:00 /usr/sbin/atd
  334  ?  S    0:00 (netplan)
 8068  ?  S    0:00 httpd
 8069  ?  S    0:00 httpd
 8070  ?  S    0:00 httpd
 8071  ?  S    0:00 httpd
...
```

プロセスの稼動状況の監視 (続き)

```
$ ps u
```

プロセスの詳細な情報を得る

USER	プロセスの所有者
PID	プロセス番号
%CPU	CPU を使用している時間の割合
%MEM	メモリの使用率
SIZE	仮想イメージの合計 (KB)
RSS	実メモリの使用サイズ (KB)
TTY	使用端末
STAT	プロセスの状態
START	プロセスを生成した時刻
TIME	プロセスが CPU を使用した累計時間
COMMAND	コマンド名

...ps コマンドの続き

引数　u　　それぞれのプロセスについて、詳細な情報を出力します。

u をつけて実行すると、次のような出力が得られます。

```
$ ps u
USER       PID %CPU %MEM  SIZE   RSS TTY STAT START   TIME COMMAND
yyamazak 14545  0.0  1.5  1888  1004  p2 S    00:40   0:00 -bash
yyamazak 15089  0.0  0.7   872   504  p2 R    05:46   0:00 ps u
```

USER　プロセスの所有者
PID　プロセス番号
%CPU　CPU を使用した時間を、プロセスの相実行時間で割ったもの、そのプロセスがシステムにおいてどれくらいの割合で実行されているかの目安になります。
%MEM　メモリの使用率
SIZE　text, data, stack を合計した仮想イメージの合計 (KB)
RSS　実メモリの使用サイズ (KB)
TTY　使用端末
STAT　プロセスの状態
START　プロセスを生成した時刻
TIME　プロセスが CPU を使用した累計時間
COMMAND　コマンド名

ps aux はよく使用されます。全プロセスのかなり詳細な情報を得ることができます。

```
$ ps aux
USER       PID %CPU %MEM  SIZE   RSS TTY STAT START   TIME COMMAND
bin        183  0.0  0.3   952   200  ?  S   Apr 13   0:00 portmap
daemon     165  0.0  0.1   808   104  ?  S   Apr 13   0:00 /usr/sbin/atd
nobody     334  0.0  0.0  1020    52  ?  S   Apr 13   0:00 (netplan)
nobody    8068  0.0  1.2  1976   808  ?  S   Apr 18   0:00 httpd
nobody    8069  0.0  1.2  1976   808  ?  S   Apr 18   0:00 httpd
nobody    8070  0.0  1.2  1976   808  ?  S   Apr 18   0:00 httpd
nobody    8071  0.0  1.2  1976   808  ?  S   Apr 18   0:00 httpd
nobody    8072  0.0  1.2  1976   808  ?  S   Apr 18   0:00 httpd
nobody    8073  0.0  1.2  1976   800  ?  S   Apr 18   0:00 httpd
nobody    8074  0.0  1.2  1976   800  ?  S   Apr 18   0:00 httpd
nobody    8075  0.0  1.2  1976   800  ?  S   Apr 18   0:00 httpd
nobody    8076  0.0  1.2  1976   800  ?  S   Apr 18   0:00 httpd
nobody   8077  0.0  1.2  1976   800  ?  S   Apr 18   0:00 httpd
```

プロセスの強制終了

kill

指定したプロセスを終了させる

```
$ kill プロセス番号
$ kill -9 プロセス番号
```

-9 は通常の kill で終了しないプロセスを止める時に使用

プロセスの強制終了

プロセスは処理が終えると自動的に消滅しますが、コマンドから強制的に終了させたい場合は kill コマンドを使用します。プログラムが正常な振る舞いをしない、バック・グランドプロセスなので ^C で止めることができない等のときは、kill コマンドを使用します。

kill 引数として対象のプロセスのプロセス番号が必要です。事前に ps コマンドで調べておきます。

$ kill プロセス番号

例）バックグランドで稼動させた sleep 200 というコマンドを強制終了する

```
$ sleep 200 &
[3] 15177
$ ps
  PID TTY STAT TIME COMMAND
14545  p2 S    0:00 -bash
15177  p2 S    0:00 sleep 200
15178  p2 R    0:00 ps
$ kill 15177
[1]   Terminated              sleep 200
$
```

SIGNAL 9 による終了

ごくまれに、通常の kill コマンドでは、終了できないプロセスや ^C でも終了できないプロセスが出現します。一般的には、プログラムの誤動作によるものですが、こういったプロセスを終了させるには、kill に -9 という引数をつけて実行すると終了できます。

$ kill -9 プロセス番号

通常の終了の仕方、つまり、引数に -9 をつけないで終了する場合は、プログラムが終了時の後処理をする余地を残して終了できますが、-9 はそれを行わせず、即座に終了させます。できるだけ、**通常の終了を優先的に行ったほうがよいでしょう**。

また、プロセスを終了できるのは所有者かシステム管理者のみです。

第９章 プロセスの実行と監視　演習問題

１）　現在、システムで稼動している全プロセスを表示し、確認してください。[実習]

２）　sleep 500 をバック・グランドで実行してください。[実習]

３）　そのプロセスのプロセス番号を調べてください。[実習]

４）　そのプロセスを終了させてください。[実習]

５）　プロセス番号１ 番を kill してください。どのようなメッセージが出ましたか？ [実習]

第１０章 ネットワーク・クライアント

学習目標

UNIX システムは、その発展過程の初期段階からネットワーク機能が OS の一部として組み込まれてきました。UNIX を使ってネットワーク環境を利用するいくつかのコマンドについてみていきます。

ネットワーク機能

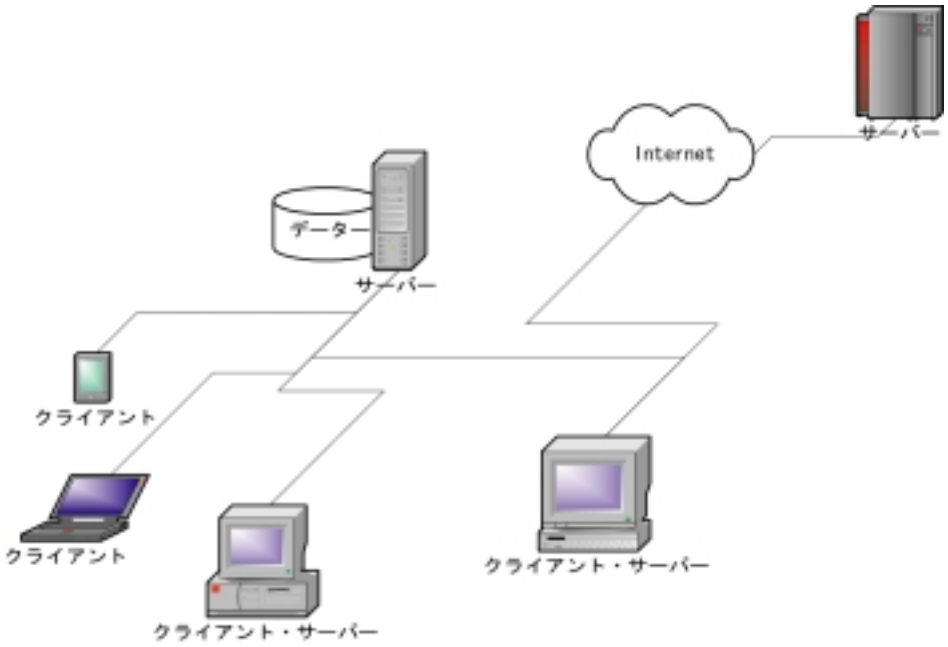
プロトコル

TCP/IP インターネットで用いられているプロトコル

クライアント / サーバー

クライアント　　サービスを利用する

サーバー　　サービスを提供する



ネットワーク機能

異なる２つのコンピューター間で情報の伝達を行う事が、ネットワークの目的です。UNIX ではネットワークを利用する為の機能が提供されています。

プロトコル

ネットワークを利用するときは、２者間で共通のネットワーク・プロトコルを持つ必要があります。UNIX では TCP/IP(Transmission Control Protocol / Internet Protocol) をはじめとするいくつかのプロトコルが実装されています。TCP/IP はインターネットで用いられたプロトコルで、その他の OS(Microsoft Windows, Apple Macintosh, IBM OS/2 等) でも同様に実装されています。同一のプロトコルを実装することが、通信を行う際の最低限の条件となります。

クライアント / サーバー モデル

通信をする際、双方のコンピューターでサービスを提供する側と、サービスを受ける側で、異なる役割を担当するのが一般的です。サービスを提供する側をサーバー、サービスを受ける側をクライアントと呼び、コンピューターのネットワーク通信では、このモデルを利用した仕組みが多く用いられています。

サーバー

クライアントに対しサービスを提供するプログラムです。このサービスを利用する為には、対応するクライアントプログラムが必要になります。通常、サーバーは複数のクライアントからの依頼を処理し、デーモンプロセスとして稼動しています。

サーバープログラムの例

サーバープログラム	提供するサービスの内容
telnetd	ネットワークを介してログインし、遠隔の操作を可能にする
rlogind	ネットワークを介してログインし、遠隔の操作を可能にする
ftpd	他のコンピューターにファイルを転送する
httpd	ウェブのサービスを提供する
smbd, nmbd	MS Windows にリモートファイルシステムのサービスを提供する
nfsd, mountd	UNIX にリモートファイルシステムのサービスを提供する
fingerd	ユーザー情報を提供する
sendmail	メールを配送するサービスを提供する

クライアントプログラムの例

サービスを利用したいユーザーが、クライアント・プログラムを起動し、サーバーの提供するサービスを利用します。

クライアントプログラムの例

クライアントプログラム	対応するサーバー
telnet	telnetd
rlogin, rsh, rcp	rlogind, rshd
ftp	ftpd
Netscape Communicator Microsoft Internet Explorer	httpd, ftpd etc...
finger	fingerd

コンピューターの名前

IP アドレス

32 ビットの整数

172.17.0.222

ホスト名

IP アドレスに対応した名前 /etc/hosts

linux01

ドメイン名

DNS を用いたホスト名の階層的な表記

linux01.sr2.knowd.co.jp

コンピューターの名前

ネットワーク上にある、他のコンピュータを指定するとは、コンピューターの名前を指定しなくてはなりません。TCP/IP のネットワークでは、コンピューターは 32 ビットの番号で識別されます。この番号を IP アドレスと呼びます。IP アドレスは人間が覚えにくいので、覚えやすい名前をアルファベットでつけることができます。これをホスト名と呼びます。更に、ネットワーク上に存在するコンピューターの数がたくさんある場合は、名前の管理作業を軽減する為に、ネットワークを用いた名前の管理技術が使われる場合があります。

IP アドレス

ネットワーク上のコンピューターの識別番号で、ネットワークを共有する（通信可能な範囲）で、各コンピューターに一意の番号を割り当てます。32 ビットで表現されます。8 ビットをドットで区切り、10 進数で表わします。

例）

```
172.12.200.14
```

ホスト名

IP アドレスを元に、人間が扱いやすいような名前をつけたものです。IP アドレスにどのような名前がつけれられているかは、/etc/hosts というファイルにその情報を設定します。

例）

/etc/hosts の例

```
127.0.0.1                   localhost localhost.localdomain
172.16.0.222                linux01
172.16.0.223                linux02
```

ドメイン名

ネットワーク上のコンピューターの数が増えたり、地理的に離れた位置でネットワークを共有している場合、各コンピューターの hosts ファイルによる管理では困難になってきます。組織内のホスト名の管理を小数のサーバーに管理させ階層的に扱うことができます。1 つの組織をドメインと呼び、この仕組みを使用した場合、ホスト名はドメイン名を含んだ形で指定することになります。

この仕組みは、DNS(Domain Name Service) という仕組みを用いた場合の表記方法です。

例）

日本にある、knowd という企業の、スクール・ルーム 2 番の linux01 というホスト名を表わしています。

```
linux01.sr2.knowd.co.jp
```

ネットワークが起動しているかを確認する

ping ホスト名

通信できない場合は無反応

ホスト名と IP アドレスの変換の確認

ホスト名の変わりに IP アドレスを用いて確認
/etc/hosts を確認
DNS を用いている場合は、nslookup コマンドで確認

```
$ nslookup
> ホスト名
```

ネットワークが起動しているかを確認する

ping ホスト名

相手のコンピューターと TCP/IP を用いた通信ができるかどうかを確認します。もし、何らかの理由（ケーブルの断線や、経路の設定のミス、コンピューターが起動していないなど）で通信できないときは応答しません。

通信可能なときは以下のような出力をします。

```
$ ping linux00
PING linux00 (172.16.0.34): 56 data bytes
64 bytes from 172.16.0.34: icmp_seq=0 ttl=128 time=0.3 ms
64 bytes from 172.16.0.34: icmp_seq=1 ttl=128 time=0.2 ms
64 bytes from 172.16.0.34: icmp_seq=2 ttl=128 time=0.2 ms
64 bytes from 172.16.0.34: icmp_seq=3 ttl=128 time=0.2 ms
^C 確認したら ^C で止める
統計結果を出力
--- linux00 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.2/0.2/0.3 ms
$
```

上記のような出力があった場合、そのコンピューターのネットワークが正しく起動していると判断できます。これで、ネットワークの通信を行う最低限のテストができます。

自分のコンピューターのネットワークの確認をしたいときは、自分を対象に ping コマンドで調べます。

ホスト名と IP アドレスの変換ができているかを確認する

ホスト名を指定すると、OS はそれを IP アドレスに変換します。もし、この変換がうまくいかなければ、相手のコンピューターをうまく指定することができず、ネットワーク・コマンドが正しく動作しません。この変換作業を適切に設定するのは、システム管理者の役割の一つですが、次の項目をチェックしてください。

1）ホスト名の変わりに、IP アドレスを用いて通信できるかを確認する。
2）/etc/hosts ファイルに IP アドレスとホストの指定がされているかを確認する。
3）DNS を用いている場合、DNS サーバーが正しく働いているかを確認する。

DNS の働きを確認するには nslookup コマンドを用います。

nslookup

コマンドを起動すると、> というプロンプトが出てきますので、ホスト名を入力します。DNS サーバーから IP アドレスを検索し出力するはずですが、これがうまくいかない場合は、DNS サーバーに関する設定が正しくないことを表わします。exit で終了します。

```
$ nslookup
Default Server:  kdcgw0.office.knowd.co.jp
Address:  172.16.0.254

> linux00
Server:  kdcgw0.office.knowd.co.jp
Address:  172.16.0.254

Name:    linux00.sr2.knowd.co.jp
Address:  172.16.0.222

>exit
```

リモートによるログイン

telnet, rlogin

相手のシステムに必要なこと

telnetd、rlogind が設定されている
自分のユーザー名がシステムに登録されている

ログアウトすると、接続を切断する

リモートによるログイン

他のコンピュータにネットワークを通じて遠隔でログインする為のサービスには、telnet と rlogin があります。これにより、他のコンピューターで UNIX コマンドを実行、操作することができます。

telnet

telnet **ホスト名**

ホスト名でして指定したコンピューターにログインすることができます。相手のコンピューターでは、telnetd（サーバー）が設定されていることと、自分のユーザー名が登録されていることが必要になります。telnet は UNIX 以外の OS でも広く使うことのできるコマンドです。

例）

linux00 というコンピューターに telnet を使ってログインする。

```
$ telnet linux00
Trying 172.16.0.222...
Connected to linux00
Escape character is '^]'.

Red Hat Linux release 5.2(J) (crimson)
Kernel 2.0.36 on an i686
login: root
Password:
Last login: Tue May 25 14:30:20 from beatle
[root@linux00 /root]#
```

rlogin

rlogin [-l **ユーザー名**] **ホスト名**

telnet 同様、他のコンピューターにネットワークを介してログインすることができます。rlogin は主に UNIX で使われており、管理者の設定次第では、パスワードなしでログインするようにもできます。telnet 同様、相手のコンピューターには rlogind（サーバー）が設定されていることと、自分のユーザー名が登録されていなくてはなりません。

```
$ rlogin  -l root linux00

Password:
Last login: Wed May 26 15:50:12 from beatle
[root@linux00 /root]#
```

接続の終了

telnet, rlogin 共に、接続を終了はログアウトの手順と同様、**exit** コマンドを使用します。

リモートによるファイルのコピー、転送

ftp (File Transfer Protocol)

ファイルの転送を対話的に行う
ftpd（サーバー）の設定が必要

rcp

cp コマンドと同じ操作で、リモート・コピーを行う
rshd（サーバー）とトラスティッド・ホストの設定が必要

リモートによるファイルのコピー、転送

他のコンピューターが持っているファイルを、自分のコンピューター上にコピー、転送することができます。

ftp (File Transfer Protocol)

ftp ホスト名

指定したホストに接続し、ファイルを転送します。相手のコンピューターには、ftpd（サーバー）が設定されていることと、ユーザー名が登録されていなければなりません。ftp でログインが完了すると、ftp> というプロンプトが現れますので、そこから ftp のコマンドを実行することで、ファイルの転送が行えます。

ftp のコマンド

ls, cd, mkdir, pwd 等のファイル操作コマンドは､通常の UNIX コマンドと同じ意味で使用することができます。

get file_name	ファイルを、ローカルなコンピューターに転送します。
mget file_name	ファイルを、ローカルなコンピュータに転送します。ワイルドカードが使えます。
put file_name	ファイルを、相手のコンピューターに転送します。
mput file_name	ファイルを、相手のコンピューターに転送します。ワイルドカードが使えます。
prompt	mput, mget で複数のファイルを転送する時、確認するかしないかを設定します。
quit 又は bye	接続を終了します。

例）

```
# ftp sherry
Connected to sherry.office.knowd.co.jp.
220 sherry.office.knowd.co.jp FTP server (Version wu-2.4.2-academ[BETA-18](1) Sa
t Feb 27 15:13:36 JST 1999) ready.
Name (sherry:root): yyamazaki
331 Password required for yyamazaki.
Password:
230 User yyamazaki logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> get /etc/passwd
local: /etc/passwd remote: /etc/passwd
200 PORT command successful.
150 Opening BINARY mode data connection for /etc/passwd (773 bytes).
226 Transfer complete.
773 bytes received in 0.000119 secs (6.3e+03 Kbytes/sec)
ftp>
```

rcp

rcp [-r] [ホスト名]:コピー元　[ホスト名]:コピー先

UNIX の cp コマンドとほぼ同様の操作で、コンピューター間のファイルのコピーが行えます。主に UNIX 間で使われているコマンドで、使用する際は相手のコンピューターで rshd（サーバー）の設定、及び、マシン間での信頼（トラスト・ホスト）の設定が事前に必要になります。

例）

linux00 の /home/mydir/text_file を、自分のコンピューターの /tmp にコピーする。

```
$ rcp linux00:/home/mydir/text_file /tmp
```

リモートによるコマンドの実行

rsh

非対話的にリモートマシンでコマンドを実行する
rshd（サーバー）及び、トラスティド・ホストの設定が必要

リモートマシンのログイン状況の確認

finger

fingerd の設定が必要

リモートによるコマンドの実行

非対話的に、他のコンピューターでコマンドを実行することができます。

rsh

rcp [-l ユーザー名] ホスト名 コマンド名

rlogin と似ていますが、相手のコンピューターで指定したコマンドラインを実行したら、自動的に接続を終了します。主に UNIX で使用されているコマンドで、相手のコンピューターに rshd（サーバー）の設定、及びマシン間での信頼（トラスト・ホスト）の設定が事前に必要になります。

例）

linux00 で date コマンドを実行する

```
$ rsh linux00 date
Wed May 26 18:25:33 JST 1999
$
```

リモートによるログイン状況の確認

他のコンピューターで現在作業している人のリストや、特定のユーザーの詳しい情報を得ることができます。

finger

finger [ユーザー名]@ ホスト名

引数全てを省略すると、自分のコンピューターに関する情報を表示します。@ に続いてホスト名を指定すると、指定したコンピューター上にログインしている人の情報を出力します。ユーザー名を指定するとそのユーザーだけの情報を出力し、詳細な情報を出力します。主にUNIXで使用され、fingerd(サーバー)の設定が必要になります。

例）

linux00 のログイン状況を調べます。

```
# finger @linux00
[linux00]
Login    Name                   Tty  Idle  Login Time   Office     Office Phone
root     root                   *1    30d  Apr 26 09:34
rotake                           p3   18d  May  8 16:52 (petri)
sybase   Sybase ASE DBA account  p2    6d  May 19 14:37 (ant)
```

例）

linux00 の sybase というユーザーの詳しい情報を得ます。

```
# finger sybase@linux00
[linux00]
Login: sybase                           Name: Sybase ASE DBA account
Directory: /opt/sybase                  Shell: /bin/bash
On since Wed May 19 14:37 (JST) on ttyp2 from ant
   6 days 23 hours idle
No mail.
No Plan.
```

第 10 章 ネットワーク・クライアント 演習問題

1） TCP/IP でネットワーク通信する際、相手のコンピューターを特定する 32 ビットの番号を何といいますか？

2） IP アドレスとホスト名の対応が書かれているファイルは何ですか？

3） リモートの linux00 というコンピューターから ftp コマンドを用いて、ファイルを取得したい場合、linux00 に必要な設定はどのようなものですか？

4） 講師のマシンに telnet 又は、rlogin を使用してログインしてください。[実習]

5） 現在、講師のマシンには誰がログインしているかを調べてください。[実習]

6） 隣の人のコンピューターに telnet 又は、rlogin を使用してログインしてください。ログインできますか？できない場合は理由を推測してください。

7） www.knowd.co.jp というコンピューターは /etc/hosts に登録されていますか？

8） www.knowd.co.jp というコンピューターの IP アドレスを調べてください。[実習]