第1回UNIXゼミ

川崎 赤塚 堤

1999年4月27日

目 次

第1章　なぜUNIXを使うのか?

1.1　Windows/Macintoshを使えば良いのでは?

「UNIXを使わなければならない」この様な状況で多分考えるだろうことは,「WindowsやMacintoshがあるのに，なぜわざわざUNIXを使わないといけないのか」ということでしょう．ここでは，WindowsとUNIXの根本的な違いを指摘しながら，なぜUNIXを使わなければならないのかについて考えていきます．またここで言うUNIXは，主にLinuxを指し示すと考えてください．

1.2　UNIXはシンプルなOSである

世の中にはこの表題にそぐわないUNIXもありますが[1]，通常，UNIXのシステムはWindowsやMacintoshに比べるとシンプルな作りになっています．そして，シンプルであるためにOS自身の環境に及ぼす負荷が低く，本題である計算処理に専念することが出来ます．シンプルということはデバッグも行き届いているため[2]，システムが非常に安定しているという意味でもあります．Windows95などのように再起動をしなければならなくなることは滅多にはありません．長時間にわたって計算をし続ける計算サーバーや，ネットワークサーバーではこのことは非常に重要になります．逆にWindowsとInternet Explorerの統合のような，古いマシンを見捨ててまでの積極的なユーザビリティ[3]の改善はWindowsに比べると少ないと言って良いでしょう．そのため，Linuxなどは80386,16MBなどWindowsで現役を引退したような環境でもそれなりの動作をし，古いマシンの有効活用ができます．

1.3　開発環境

UNIX，特にLinuxなどの開発環境は実は非常に恵まれています．Windowsのように開発環境を導入するだけで何万円とかかったりはしません．ほとんどの場合，ただ，あるいは，かかったとしても本代の数千円というレベルで済みます．加えて，最近は雑誌にもCD-ROMが付いているので，うまくいけば，ほんの数百円の出費でOSから開発環境までもが揃うこともあります[4]．
また，これらは無料だからと言って決して機能的にVisual C++や，Borland C++などに劣るという物ではありません．GNU[5]が配布するgccや，makeといった開発ツール，そして何よりもOSそのもの[6]全てがタダとは思えないほどのクオリティを持っています．その上，gcc，makeや，他のツールは多くのプラットホームに移植されており[7]gccを使ってプログラムを作れば，多くのプラットホームへ

[1] Sunの Solarisは重すぎるとの意見もある．
[2] Linuxなどの場合，これにはOPENSOURCEなどの別の要因もある．
[3] 使いやすさのこと．
[4] もちろん，インターネット上でも公開されています．
[5] Freeware Software Foundationが主催するプロジェクトの名前．GNU is not Unix.と定義が再起的になっており，本当は何なのか良く分からないが，この世界では最も良くお世話になるところである．
[6] GNU Debian Linuxのこと．
[7] Windows用のgccも存在し，POSIX準拠のアプリケーションだけではなく，Windowsのアプリケーションも作成することが出来る．最も有名なのは，cygnusが提供するcygwin32 `http://sourceware.cygnus.com/cygwin/` であろう．

表 1.1: Windows と Linux の比較

	Windows	Linux
要求するスペック	OS 自身が多くの機能を持つため，高機能な CPU，多くのメモリを要求する．	OS 自身は，最低限の機能しか持たないため，要求は低い．
インターフェイス	GUI(マウス中心)	CUI(キーボード中心)，GUI はオプション[11]．
対象としている分野	ワープロ，表計算，デザインワーク，簡易データベース等	計算サーバ，ネットワークサーバ，大規模データベースサーバ，その他

移植するということも視野に入ってきます．

1.4 UNIX ではやりにくいこともある

ここまで良いことずくめのように書いてきましたが，当然ながら UNIX にも問題点はあります．きっとほとんどの人がこの部分で躓きそうになるのでしょうが，UNIX は，Windows のように一筋縄では行かない部分があります．UNIX という OS 自身や，UNIX の多くのコマンドは，単機能指向であるということです．簡単に言えば，標準プログラムの多くが，一つの仕事しかできません．Windows のアプリケーションの多くが，使い切れないほどの機能を持っていますが，UNIX ではこれとは正反対にほとんどの場合，個々のプログラムが多くの機能を持つことはありません[8]．従って，アプリケーションというよりは本当に「プログラム」といった方が正しい感じです．通常，これらの単機能のコマンドを組み合わせて大きな連携作業を行なうというのが UNIX の基本発想であるので，Word のような高機能なワープロは無いといった方が良いでしょう[9]．つまりは，そのような作業 (ワープロや表計算，プレゼンテーション作り) をするには向いていないのです[10]．面倒だと感じるかもしれませんが，ある意味ではこれが便利だと思う瞬間もあることを断っておきます．

1.5 Windows と Linux の比較

これまでのことを Windows と Linux の比較という観点でまとめると，表 1.1 のようになります．プレゼンテーション用のスライドを作成するには Windows の方が良いし，GA や，並列計算，最適化などでの計算機サーバーとして使うのであれば Linux の方が良いことはこれまでに述べてきた通りです．つまりは，三木研でこれから活動をしていくためにはこの両方を上手に使いこなすことが重要です．Windows を計算サーバーにしたり，UNIX で無理をしてプレゼン資料を作ることのような無駄なことをしないために，UNIX を正しく理解しましょう．

[8] tar や gzip は高機能と言えなくもない．
[9] オムロンから dpNote というオフィススイートが発売されてはいるが，Word97 などの表現力には遠く及ばない．また，漢字変換のシステムも Windows からすれば紀元前並の馬鹿さ加減である．これを使える環境とは言えないであろう．
[10] 向いてなくても無理をする人はいることを断っておく．
[11] これについては，1.6 で述べる．

1.6 UNIX の Window システム

UNIX というと，つい，真っ暗な画面に，文字だけが浮かんでいるという光景を考えるようであるが，これはある意味では間違いである．UNIX には，X Window というウィンドウシステムが存在するからである．この X Window には次のような特徴がある．

1. OS から独立した存在の GUI

   UNIX 自身は X Window を必ずしも必要としておらず，インストールしたくなければインストールしなければ良い風になっている．Windows ではそのようなことはできない．

2. クライアント&サーバーモデル

   1 台のマシンに X Window クライアントをインストールすれば，X Window サーバーを用いることによって複数の人が 1 台のマシンで**同時に**作業を行うことができる．Windows の GUI システムでは，1 台のマシンで同時に作業が出来るのは，1 人のユーザーだけである．

3. Window マネージャという概念

   ウィンドウマネージャとは，ウィンドウの管理を行うシステムである．このシステムは，ウィンドウの外見や振る舞いを制御するのであるが，X Window システムではこの部分を自由に入れ替えることが出来る．従って，ユーザーは数多く提供されたウィンドウマネージャの中から，自分の好きな外見，振る舞いのウィンドウシステムを選択することが出来る．Windows では，無条件に Explorer を使わなければならない．

第2章　ssh環境を構築する

2.1　ssh環境をインストールする

実際にUNIXマシンをネットワークを通じて操作するには，ssh，telnet，あるいは，X-Windowシステムのサーバー[1]を利用することになります．ここではX-Windowシステムは置いておいて，一番使うであろう，ssh環境を構築します．その端末としてはWindows環境で最も定評のあるターミナル[2]エミュレータTera Term Pro(`http://hp.vector.co.jp/authors/VA002416/`)を使うことにします．しかし，Tera Term Proの通常版はそれ単体ではsshには対応していませんので，ここでは，`http://www.zip.com.au/~roca/ttssh.html`で，Robert O'Callahan氏によって公開されているTera Term Pro用アドインモジュールTTSSHを使うことにします．ただし，今回は，sshの導入を図ることが目的であり，ダウンロード，インストールのプロセスを簡略化するために，ttinstという独自パッケージを用意しました．このパッケージは，`\\cosmos\DIA shared\ttinst\ttinst.exe`を直接実行するか，あるいは，`\\cosmos\DIA shared\ttinst`以下のファイルを全て自分のコンピュータにコピーしてから，ttinst.exeを実行することによりインストールできます．ttinst.exeを実行すると，図2.1のような画面が表示されます．ここでは，インストールを続行しますので，`OK`を選択します．途中，Tera Term Proのインストール経過が表示されたり，ファイルのコピー状況が文字で表示されたりしますが，特に何かの入力を求められることはありません．図2.2のように表示されたらインストール完了です．

2.2　最初のログインとパスワード変更

それでは，第2.1章でインストールした環境で，実際にsshを使って通信してみましょう．デスクトップ上に既にssh - cosmosというショートカットがあるでしょうから，それをダブルクリックしてください．初めて起動する場合には，図2.3のような警告が出ますが，ここでは気にせずに`Continue`を押して先に進んでください．次回以降のcosmosへのアクセスでは表示されなくなるはずです[3]．また，このとき，図2.4の様なエラーも同時に表示されますが，インストールしたばかりの段階では`known_hosts`

図 2.1: インストール開始

[1]通常のクライアント・サーバーモデルと逆であることに注意。ちなみにrev.1ではこのレジュメの表記も間違っていました。
[2]基本的にネットワークや，特定の回線越しにコンピュータを操作する端末(機械)のことをターミナルと呼びます．これ以降では，ssh，telnet，X Windowのxterm，ktermなどを総称してターミナルと記述します．

図 2.2: インストール完了

ファイルがないことを示しているだけで，これも次回以降は表示されません．

次に，2.5 の様なダイアログが表示されます．ここでは，User name と書かれたボックスにユーザー名を，Passphrase[4]と書かれたボックスにはパスワードを入力します．また，[Use plain password to log in] にチェックが入っていることを確認してください．全て問題なければ，OK を押すか，Enter キーを押すとログインできます．

初めてログインする場合には，管理者によって何らかのパスワードが設定されていますが，このキーワードはセキュリティ上問題になることが多いので，ログインしたら，とりあえず最初にパスワードの設定をすることにします．これには，passwd コマンドを使います．以下の部分では，パスワードを変更しようと試みています (太字はキーボードからの入力)．最初に apple というパスワードに変更しようとし,「パスワードが簡単すぎる」と警告[5]されたので，次には複雑なパスワードを設定しています．なお，mikilab でのパスワードの変更では，パスワードは必ず 3 回入力を求められます．

```
Last login: Sun Apr 18 16:57:05 1999 from 192.168.6.***
You have mail.
[kawasaki@mikilab ~]$ passwd
Changing password for kawasaki
(current) UNIX password: pipin@[6]
New UNIX password: apple[6]
BAD PASSWORD: is too simple
New UNIX password: Rh345!sa[6]
Retype new UNIX password: Rh345!sa[6]
passwd: all authentication tokens updated successfully
[kawasaki@mikilab ~]$
```

これで，ssh 及び，mikilab の環境を設定し終えました．cosmos は，外部ネットワークに接続されていないため，ssh ではなく，telnet という仕組みを使ってアクセスすることになりますが，ほぼ同じ手順でログイン，パスワードの変更が行えます．

[3]ここでは，known_hosts いうファイルに接続先の RSA 公開鍵が登録されます．次回の接続以降にはそれを使って接続先が入れ替わっていないか確認され，入れ替わっていた場合には図 2.3 の警告が出るようになります．また，この公開鍵は通信内容の暗号化の一部のプロセスにも使われます．

[4]パスフレーズというと，実際には空白を含んだパスワードよりも長い文章というイメージがありますが，これは単にパスワードとした場合，短い単語というイメージがあるために，敢えてこうしたのでしょう．

[5]passwd プログラムのパスワード評価は結構厳しく，この他にもいろいろな警告をしてくる可能性があります．

[6]実際には入力されたパスワードは表示されません．

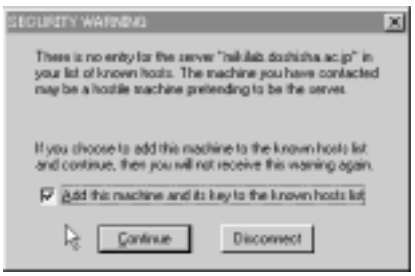

図 2.3: ホストが正しくないかもしれないという警告

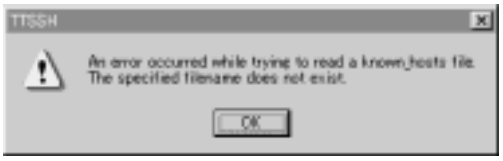

図 2.4: ファイルを読み込めないという趣旨のエラー

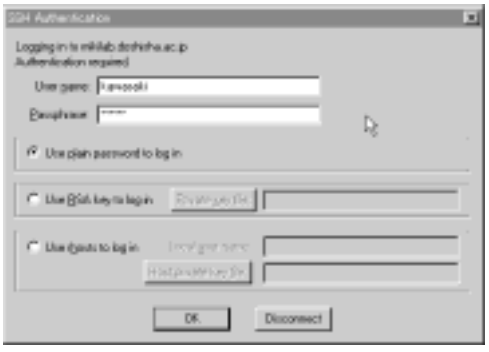

図 2.5: SSH のログイン認証

第3章　ターミナル上での操作

3.1　この章の概要

この章では，ターミナル（Telnet,ssh,X-Window のコンソールなど）上で，実際に使用するコマンドを説明します．また，それに先立ちファイルを扱う上で非常に重要な概念[1]である，ディレクトリ構造について第 3.2.1 節で詳しく説明しています．続く第 3.3 節では，ディレクトリの実際の操作方法について解説します．また，第 3.4 節でパーミッションについて，第 3.5 節でファイル操作について，第 3.6 節でその他のコマンドについて紹介しています．

3.2　ディレクトリ構造

3.2.1　ディレクトリ構造とは

UNIX は，複数のユーザーが利用します．これらの人がごちゃごちゃにファイルを作っては，どのファイルが誰のファイルかわからなくなってしまいます．また，自分のファイルでもファイルが増えてくると，どのファイルが何のファイルなのか区別がつきにくくなってしまいます．そこで，UNIX では[2]，ディレクトリ構造という仕組みを利用して，ファイルを管理しています．ひとことで言えば，ファイルを種類ごと[3]にまとめて，そのまとまりに名前を付けるようなものです．また，第 3.4 節で詳しくふれますが，UNIX ではファイルごとに読み込み/書き込み/実行の権限を設定できるのですが，これはディレクトリにも設定する事ができるのです．これにより，自分のディレクトリに他人が書き込めないようにしたりする事ができます．また，他人に見られたくない画像をまとめて保管することなどもできます．このように，ファイルを管理する上において，ディレクトリ構造は非常に有効なのです．ディレクトリ構造は，図 3.1 のように木構造になっています．

この図では各教室をファイル，建物をディレクトリと見立てて，話をしています．同志社というもっとも大きな建物の中に，知真館という建物，デイビスという教室[4]，ラーネッドという教室，恵道館という建物があります．さらに，知真館という建物の中には，１号館，２号館，３号館という建物があり，恵道館という建物の中には，KD103，KD201，KD302 という教室があります．そして，３号館という建物の中には，3-102，3-201，3-202 と言う教室があります．

同じ事を，ディレクトリとファイルという言葉に置き換えて考えてみます．同志社というもっとも大きなディレクトリの中に，知真館というディレクトリ，デイビスというファイル，ラーネッドというファイル，恵道館というディレクトリがあります．さらに，知真館というディレクトリの中には，１号館，２号館，３号館というディレクトリがあり，恵道館というディレクトリの中には，KD103，KD201，KD302 というファイルがあります．

---

[1] ディレクトリの概念は，意識していない方が多いようですが，MS-Windows を使う上でも非常に重要な概念ですので，ここで完璧に理解しておいて下さい．

[2] 先にも述べたように，MS-Windows をはじめとした，ほぼすべてのＯＳで，この仕組みは利用されています．

[3] 自分の好きなようにまとめることができます．

[4] 普通デイビスを教室とは言いませんが，ここでは便宜上そのように扱います

図 3.1: 同志社ディレクトリ

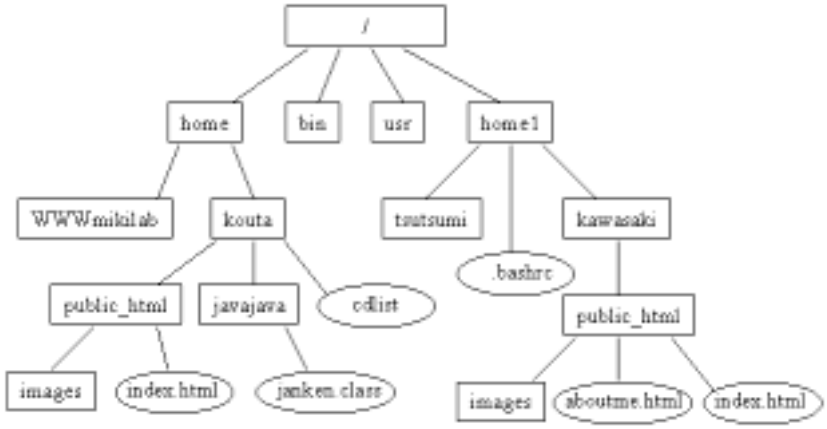

図 3.2: mikilab マシンのディレクトリ

実際のコンピュータの中も，このように管理されています．また，我々は必ずどこかのディレクトリを見ている（どこかのディレクトリにいる）ことになります．ここで少し用語を解説しておきます．同志社というもっとも大きなディレクトリがありますが，このすべてを含むディレクトリを，**ルートディレクトリ**といいます．また，自分のファイル（またはディレクトリ）より１つ上のディレクトリをそのディレクトリの**親ディレクトリ**，１つ下のディレクトリをそのディレクトリの**子ディレクトリ**または**サブディレクトリ**と言います．たとえば，知真館ディレクトリの親ディレクトリは同志社ディレクトリ（ルートディレクトリ），サブディレクトリは１号館，２号館，３号館ディレクトリになります．また，現在自分のいるディレクトリを，**カレントディレクトリ**と言います．

3.2.2 mikilab マシンでの実際

では，mikilab マシンでは，実際にはどのような構造になっているのでしょうか．それを表したのが図 3.2 です[5]．

図中１番上の部分にある記号”/”は，ディレクトリの区切りに利用する文字ですが，ルートディレクト

[5]各ディレクトリには，実際はもっと多くのファイルやディレクトリが存在しますが，説明の便宜上，大幅に省略しています

リを表すときにも用います．mikilab マシンのルートディレクトリには，home，home1，bin，usr の４つ[5]のディレクトリがあることがわかります．また，kawasaki ディレクトリの親ディレクトリはhome1，サブディレクトリには public_html があり，kouta ディレクトリの親ディレクトリはhome，このディレクトリにはファイルcdlistが，サブディレクトリには public_html と javajava があることがわかります．

3.2.3 絶対パスと相対パス

UNIX 上のファイルやディレクトリを指すには，絶対パスと相対パスという２つの方法があります．絶対パスは，常にルートから考えて，指定したいファイルやディレクトリがどこにあるか指定する方法です．相対パスは，カレントディレクトリから考えて，指定したいファイルやディレクトリを指定する方法です．たとえば，カレントディレクトリがkouta だとすると，kawasaki ディレクトリを指定するには，絶対パスでは/home1/kawasaki となり，相対パスでは../../home1/kawasaki となります．相対パス中に出てきた../は，親ディレクトリを示します．つまり，カレントディレクトリがkouta の場合，../はhome ディレクトリを指します．javajava ディレクトリを指すには絶対パスでは，/home/kouta/javajava となりますが，相対パスではカレントディレクトリがkouta なら javajava となり，カレントディレクトリがWWWmikilab なら../kouta/javajava となります．

3.3 ディレクトリ操作

3.3.1 ホームディレクトリ

第 3.2.1 節でディレクトリの構造を説明しましたが，ここでは実際にディレクトリを移動・作成・削除する方法について，説明します．その前に，ホームディレクトリについて説明します．ホームディレクトリとは，UNIX にログインした時最初にいる場所のことで，みなさんの場合ユーザー名と同じになっています．特殊な場合を除いて，ホームディレクトリは UNIX 上に与えられた，みなさん専用の場所です．みなさんは，ここにホームページや，メールのセーブなどをおくことになります．これは mikilab マシンでも同じで，みなさんはログインすると自動的にhome かhome1 の下の自分のユーザー名のディレクトリに移動します．たとえば，赤塚の場合ホームディレクトリは，kouta であり，川崎の場合kawasaki です．従って，通常はホームディレクトリより上のディレクトリ構造をあまり意識する必要はありません．また，kouta ディレクトリの下のファイルやディレクトリはすべて赤塚が作成したものですし，kawasaki ディレクトリの場合は川崎が作成したものです．

3.3.2 カレントディレクトリの取得と移動

現在のディレクトリを知るには，pwd コマンドを用います．プロンプトから

```
pwd
```

と入力します．現在，kouta にいるとすると，

```
/home/kouta
```

と結果が表示されます．

ディレクトリを移動するには，cd コマンドを用います．たとえば，ユーザー赤塚がログインし，javajava ディレクトリに移動するには，絶対パス指定では

```
cd /home/kouta/javajava
```

相対パス指定では

```
cd javajava
```

と入力します．

3.3.3 ディレクトリの内容を表示

指定されたディレクトリには，どんなファイルやディレクトリがあるのかを知るには，ls コマンドを使用します．ls コマンドには，様々なオプションがありますが，ここではよく使われるls（オプション無し）とls -la について説明します．ls コマンドでは，ファイルとディレクトリの一覧が，区別無く羅列されます．たとえば，/home/kouta ディレクトリの内容を見るには，

```
ls /home/kouta
```

と入力します．もちろん，相対パス指定も可能です．また，ディレクトリを指定しないとカレントディレクトリの内容を表示します．実行すると，

```
public_html       javajava          cdlist
```

のように表示されます．もっと詳しい情報を知りたい場合は，ls -la を使用します．同じく，/home/kouta ディレクトリの内容を見るには，

```
ls -la /home/kouta
```

と入力します．実行結果は，

```
drwxr-xr-x   5 kouta    mikilab      1024 Apr 19 03:54 .
drwxr-xr-x  60 root     root         2048 Apr 13 13:24 ..
drwxr-xr-x   3 kouta    mikilab      1024 Nov  6 19:12 javajava
drwxr-xr-x   7 kouta    mikilab      1024 Jan 16 02:45 public_html
-rw-r--r--   1 kouta    mikilab     18944 Apr  8 15:03 cdlist
```

のようになります．右端がファイル名，左端がファイル・ディレクトリの区別で，ディレクトリには”d”がつきます．d の後に続く”rwx”等の文字は，第 3.4 節で詳しく述べます．

3.3.4 ディレクトリの作成と削除

ディレクトリを新たに作成するには，mkdir コマンドを用います．カレントディレクトリがkouta の状態で，kouta の下にunix ディレクトリを新たに作るには，絶対パス指定では

```
mkdir /home/kouta/unix
```

とします．すると，kouta 以下のディレクトリ構造は図 3.3 のようになります．

すでにあるディレクトリを削除するには，rmdir コマンドを用います．カレントディレクトリがkouta の状態で，kouta の下にunix ディレクトリを削除するには，相対パス指定では，

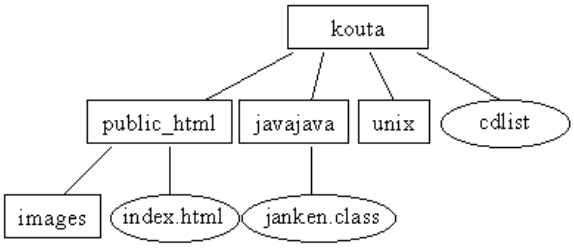

図 3.3: mkdir unix

```
rmdir unix
```

とします．すると，kouta のディレクトリ構造は図 3.2 のように戻ります．

3.4 パーミッション

3.4.1 パーミッションとは

UNIX では，複数ユーザーが同時に同じマシンを使うため，各ファイルやディレクトリごとにパーミッションと呼ばれる，操作に対する制限が設定できます (されています)．ここで言う操作とは，Read/Write/eXecute の 3 種です．これらの設定を各ファイルごとに設定できます．

ファイルに対するパーミッションの設定は，そのファイルに対しての読み/書き/実行ですが，ディレクトリに対するパーミッションは，若干意味が異なります．ディレクトリの読み込みとは，そのディレクトリにどんなファイルがあるか見ることで，書き込みとはそのディレクトリに新たにファイルを作ることを意味します．また，実行とはそのディレクトリをカレントディレクトリにすることを意味します．

また，ここからが重要なのですがこれらの権限はさらに，自分に対して/グループユーザーに対して/その他のユーザーに対して，許可するかどうかを設定できます．自分は Read/Write することができるが，グループユーザーは Read のみ，他の人は何もできない，という設定などができます．現在，どのファイルがどんな設定なのかは，第 3.3.3 で紹介した`ls -la` コマンドで見ることができます．あるディレクトリの`ls -la` の結果が，

```
drwxrwxr-x   2 sin      mikilab      1024 Apr 10 01:05 students
drwxr-xr-x   2 sin      mikilab      1024 Jan  4 17:30 database
-rw-rw-r--   1 sin      mikilab       215 Apr 10 01:08 index.html
-rw-r--r--   1 makoto   mikilab     16831 Jan  9 16:55 logoISDL.gif
```

であったとして，説明します．左に表示される，`drwxrwx---`というようなのが，そのファイルに対するパーミッションの状況です．先頭の”`d`”は先にも述べたように，ディレクトリであるときに表示されます．続く９文字は，`rwx` で１セットで全部で３セットあり，前から順に自分/グループユーザー/他人に対するパーミッションを表しています．見てわかるように，３文字の最初が Read，次が Write，最後が eXecute の権限になっています．`rwx` の文字が表示されていればその操作が許可されていることを意味し，-が表示されていれば，その操作は許可されていないことを意味しています．

表 3.1: chmod の数字の意味

0	すべての権限なし	4	Read
1	eXecute	5	Read + eXecute
2	Write	6	Read + Write
3	Write + eXecute	7	Read + Write + eXecute

たとえば，上の例であれば，students ディレクトリは，自分とグループユーザーは Read/Write/eXecute のすべてを操作できますが，それ以外の人は Read と eXecute しかできませんし，logoISDL.gif では，自分は Read と Write をできますが，グループユーザーも含めて他の人は，Read しかできません．ちなみに，リスト中の sin とか makoto が個人の名前で，この名前の人がこのファイルにおける個人ユーザーに，mikilab がグループユーザーになります．ちなみに，みなさんは，mikilab グループですので，これらのファイルには，グループユーザーとしてアクセスできます．しかし，もし次のようなリストだと，

```
-rw-r-----   1 tsuchiya student      1041 Apr  7  1997 profile.htm
drwxr-xr--   2 tsuchiya student      1024 Dec  1 23:48 seiji
```

みなさんは，個人でもグループユーザーでもない他人になりますので，profile.htm ファイルには何もできません．また，そのディレクトリに移動する（中に入る，そのディレクトリをカレントディレクトリにする）には，eXecute 権限が必要ですので，tsuchiya 本人と student グループユーザーしか，seiji ディレクトリには，入れないことになります．当然，中に何があるかも調べることができません．したがって，みなさんがもしホームページを作ったときに，public_html のアクセス権限が

```
drwxr-xr--   2 kouta    mikilab      1041 Apr  7  1997 public_html
```

とかになっていると自分と mikilab ユーザー以外は，ページを見れないと言うことになります．かならず，

```
drwxr-xr-x   2 kouta    mikilab      1041 Apr  7  1997 public_html
```

となるようにしてください．ちなみにだからといって，

```
drwxrwxrwx   2 kouta    mikilab      1041 Apr  7  1997 public_html
```

などとすると，だれでも何でも書き込みができるようになり非常に危険ですので，絶対にしないでください．

3.4.2 パーミッションの設定

パーミッションを変更するのが，chmod というコマンドです．

```
chmod 777 filename
```

という書式を持っており，filename にはパーミッションを変更したいディレクトリやファイル名を指定します．また，777 の数字の部分は順に個人/グループ/他人のパーミッションを設定します．数字にはそれぞれ表 3.1 のような意味があります．

つまり，chmod 777 index.html とすればindex.html には，すべての人にrwx の権限が設定され，chmod 755 index.html とすればindex.html には，自分はrwx，自分以外はr-x の権限が設定されます．また，chmod 640 index.html とすればindex.html には，自分はrw-，グループユーザーはr--，他人は---となります．したがって，たとえばホームページの為のファイルはchmod 644，ディレクトリはchmod 755 とすれば，誰でも読めるが自分以外書き込めない設定ができます．

3.5 ファイル操作

3.5.1 ファイルについて

この節では，ファイルの操作について説明します．ファイルの操作には，内容の表示，コピー，移動，消去があります．

3.5.2 ファイルの内容の表示

ファイルの内容を表示するには，cat コマンドかmore コマンドを用います．書式は

```
cat filename
more filename
```

です．両者の違いは，cat は全部一度に，more は 1 画面ずつ区切って表示すると言うことです．ちなみに，両者とも表示できるのはテキストファイルのみですので，バイナリファイル[6]を表示しないように気をつけて下さい．

3.5.3 ファイルの消去

ファイルを消去するには，rm コマンドを利用します．書式は，

```
rm filename
```

です．

3.5.4 ファイルのコピー/移動

ファイルをコピーするにはcp コマンドを，移動するにはmv コマンドを利用します．書式は，

```
cp filename1 filename2
mv filename1 filename2
```

です．基本的にはfilename1 にコピー/移動元ファイル名を，filename2 にコピー/移動先ディレクトリ名を指定します．図 3.4 を例に説明します．

カレントディレクトリを kouta として，ディレクトリ javajava の中の janken.class を，ディレクトリ public_html にコピーするには，

```
cp javajava/janken.class public_html
```

とします．コピー後は，図 3.5 のようになります．

また，コピー先にディレクトリ名ではなく，ファイル名を指定する事もできます．その場合，同じ内容で，ファイル名の異なるファイルができます．

[6]画像ファイルや実行ファイルなど，メモ帳で読めない形式すべて

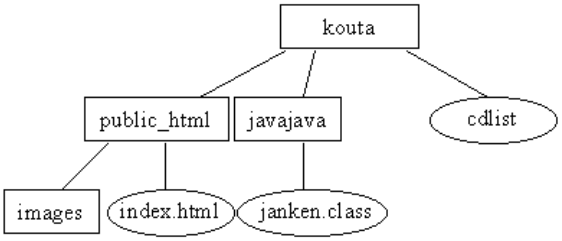

図 3.4: cp example

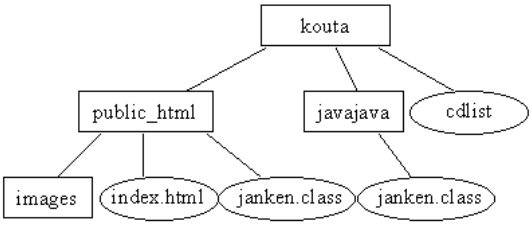

図 3.5: cp file dir

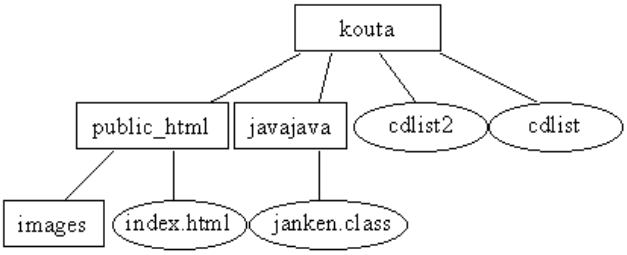

図 3.6: cp file file

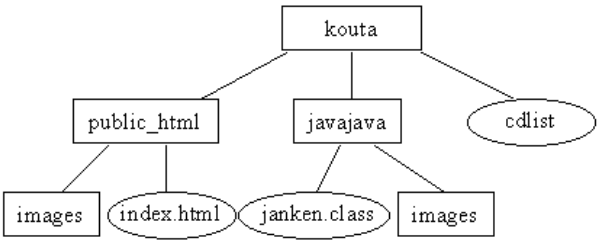

図 3.7: cp -r dir dir

```
cp cdlist cdlist2
```

実行後は，図 3.6 のようになります．

さらに，コピー元にディレクトリ名を指定し，オプション-r を指定すると，ディレクトリ丸ごとコピーができます．

```
cp -r public_html/images javajava
```

結果は，図 3.7 のようになります．

移動はコピー元が残らないだけで，あとはコピーと同じです．

3.6 その他のコマンド

3.6.1 logout

UNIX では自分の処理が終了したらログアウトしなければなりません．これを実現するのが，logout コマンドです．

3.6.2 passwd

パスワードを変更する命令です．変更は以下の手順で行います．

```
[kouta@mikilab ~]$ passwd
Changing password for kouta
(current) UNIX password:
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully
```

passwd と入力すると，現在のパスワードの入力を促されますので，現在のパスワードを入力します．次に，新しいパスワードを入力し，再度新しいパスワードを入力すると，変更が完了します．