2000年度 第1回UNIXゼミ

指導：川崎　　担当：佐野 吉田 奥田

2000年4月14日

目 次

第1章　なぜUNIXを使うのか?

UNIXは，WindowsやMacintoshなどと同じく，OSの一つです．しかし，普通のパソコンユーザにはなじみが薄いものでしょう.「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」と思う方もいると思います．そこで本章では，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[1]を指し示すと考えてください.

1.1　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[2]，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[3]，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．Windows98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[4]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.

1.2　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[5].
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[6]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[7]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[8]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移植するということも視野に入ってきます.

1.3　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

[1]フィンランドのヘルシンキ大学の学生だったLinus B．Torvalds氏がカーネルを作成した，UNIX互換のOS．正確には，UNIXではない.
[2]SunのSolarisは重すぎるとの意見もある.
[3]Linuxなどの場合，これにはOPENSOURCEなどの別の要因もある.
[4]使いやすさのこと.
[5]もちろん，インターネット上でも公開されています.
[6]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，本当は何である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.
[7]GNU Debian Linuxのこと.
[8]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[12]．
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

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[9]．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[10]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[11]．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．

1.4 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．

1.5 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，OS に GUI の機能が含まれているため，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

[9] tar や gzip は高機能と言えなくもない．
[10] パイプや論理演算子で命令をつなぎます
[11] 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．
[12] これについては，1.5 で述べる．

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．これは，先ほども述べたように，OS と Xwindow システムが独立であることに起因する．Windows では，無条件に Explorer を使わなければならない．

1.6 補足

これまで，UNIX はシンプルで初心者には難しく，信頼性が高いのでサーバ向きであり，Windows は信頼性が低く，ワープロや表計算，プレゼンテーションなどに向いているという話の展開をしてきました．しかし，Windows2000 や WindowsNT は信頼性が高くなってきておりサーバーとして用いることができます．また，現在の Linux はインストールの自動化や Window マネージャの高機能化が進み，ユーザとのインタフェースがかなり改善されています．

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，実験室からサーバへのリモートログインに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[3]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るので，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，`file://aardvark/inst/net/TeraTerm/ttinst/ttinst.exe` を直接実行することで利用できます[4]．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 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 2.4 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[5]．また，このとき，図 2.5 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` 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，図 2.6 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[6] 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．

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．

[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．

[3]ベクター http://www.vector.co.jp などのダウンロードサイトからダウンロードできます．

[4]インストールの詳しい方法は，http://192.168.6.13/telnet/index.html に掲載しています．

[5]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．初回の接続ではまだこのファイルが存在しないので，図 2.4 のような警告が出ます．次回の接続以降にはこのファイル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も図 2.4 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．

[6]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

図 2.2: インストール完了

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．図 2.3 では，パスワードを変更しようと試みています．

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．一つ目の入力（図 2.3 の `"(current) UNIX password:"` という部分）では現在のパスワードを入力します．2 つ目および 3 つ目の入力（図 2.3 の `"Enter new UNIX password:"`, `"Retype new UNIX password:"` という部分）では，変更後のパスワードの入力を行います．このとき，入力したパスワードはターミナル上には表示されません．

図 2.3 の例では，最初のパスワード変更において，4 行目と 5 行目の入力が異なるというメッセージが 6 行目に表示されています．また，次のパスワード変更（8 行目から）では,「パスワードが短すぎる」と警告[7]されたので，14 行目からの入力では複雑なパスワードを設定しています．なお，ログアウトしたい場合は，`Ctrl + d` と押してください．

これで，ssh 及び，mikilab の環境を設定し終えました．

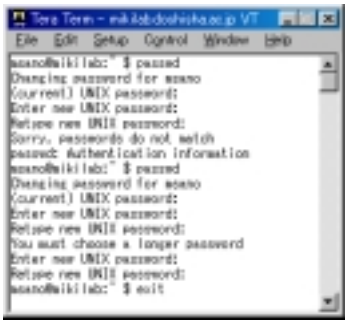

図 2.3: パスワードの変更の様子

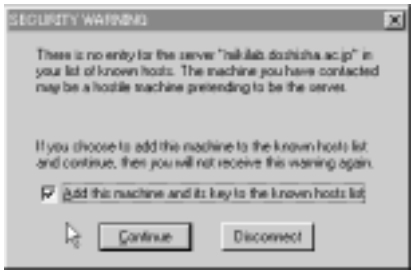

図 2.4: ホストが正しくないかもしれないという警告

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．

[7] `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．

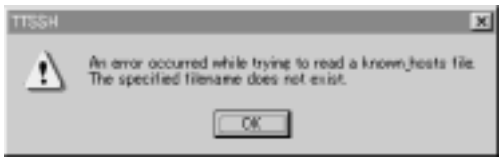

図 2.5: ファイルを読み込めないという趣旨のエラー

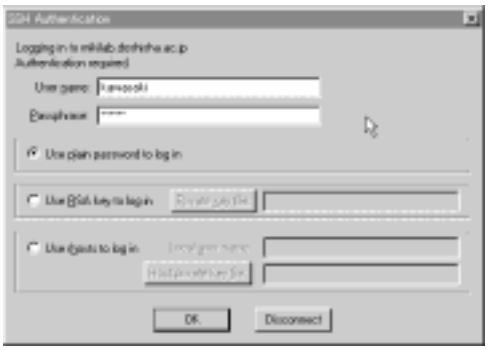

図 2.6: SSH のログイン認証

2.3 ログイン先コンピュータから他のコンピュータへのログイン

本節では，`slogin` コマンドについて説明します．2.1 節では，ssh 環境を用いて他のコンピュータにリモートログインする方法を説明しました．windows 上では，デスクトップ上のアイコンをダブルクリックすることによって，ssh を起動しました．しかし，接続先のコンピュータから，ターミナルを用いて別のコンピュータへログインする場合，このような方法は使えません．この場合には，`slogin` コマンドを使用します．ターミナル上で，

`slogin` [-l ユーザー名] [ログインしたいコンピュータ名]

と入力してください．そこからさらに別のコンピュータにログインする場合も同様です．'[' と']' で囲まれた部分は省略可能です．例として，zodiac から mikilab にログインしたときの画面を，図 2.7 に示しておきます．

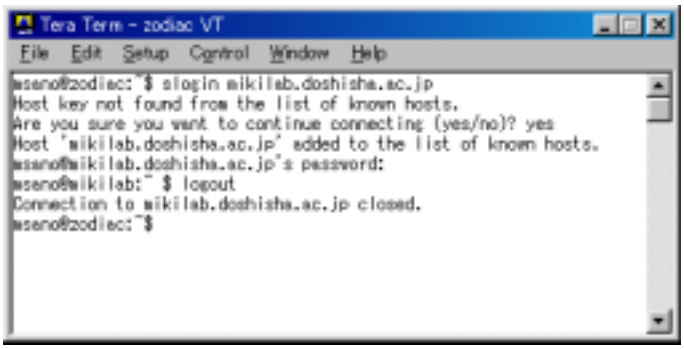

図 2.7: `slogin` コマンドを用いたログイン

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, デイビス，ラーネッド，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

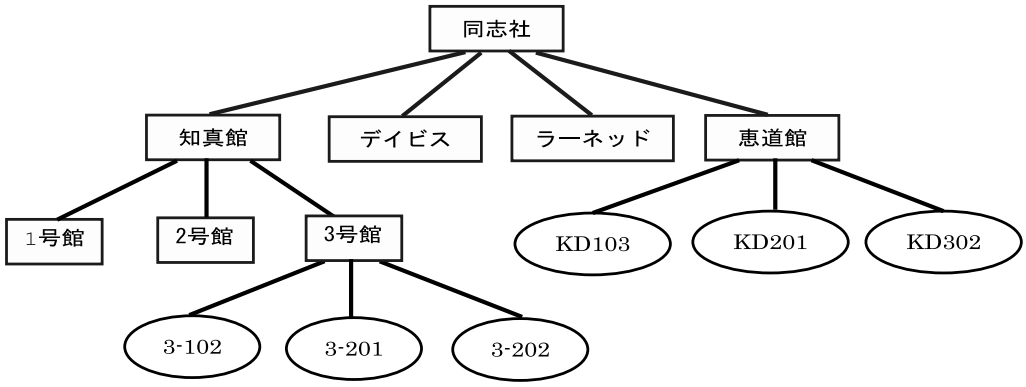

図 3.1: 同志社ディレクトリ

[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．

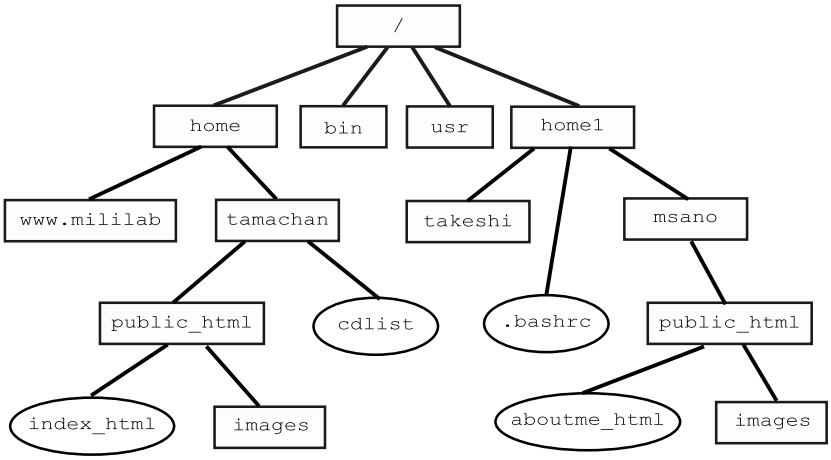

図 3.2: mikilab マシンのディレクトリ

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 という教室があります．

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，デイビス，ラーネッド，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．

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[4]．

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[4]のディレクトリがあることがわかります．また，`msano` 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，`tamachan` 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` 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`tamachan` だとすると，`msano` ディレクトリを指定するには，絶対パスでは`/home1/msano` となり，相対パスでは

[4]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

表 3.1: ディレクトリ操作コマンド

コマンド名	概　　要
pwd	現在いるディレクトリが表示されます.
cd [*dir*]	指定した（[*dir*]）ディレクトリに移動します.
ls [*option*] [*names*]	指定されたディレクトリ内のファイルとディレクトリの一覧が表示されます. -a オプションで名前の先頭に".”のついたファイルを表示し，-l オプションで詳細な一覧を表示します.
mkdir [*dir*]	ディレクトリを作成します.
rmdir [*dir*]	ディレクトリを削除します.

../../home1/msano 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tamaki の場合，../はhome ディレクトリを指します．public_html ディレクトリを指すには絶対パスでは，/home/tamachan/public_html となりますが，相対パスではカレントディレクトリがtamachan なら public_html となり，カレントディレクトリがWWWmikilab なら../tamachan/public_html 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，実際にディレクトリを移動・作成・削除する方法について，第 3.3.2 以降で説明します．簡単にまとめたものが表 3.1 です.

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，奥田の場合ホームディレクトリは，tamachan であり，佐野の場合msano 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，tamachan ディレクトリの下のファイルやディレクトリはすべて奥田が作成したものですし，msano ディレクトリの場合は佐野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，tamachan にいるとすると，

```
/home/tamachan
```

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，ユーザー奥田がログインし，public_html ディレクトリに移動するには，絶対パス指定では

```
cd /home/tamachan/public_html
```

相対パス指定では

```
cd public_html
```

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/tamachan ディレクトリの内容を見るには,

```
ls /home/tamachan
```

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       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/tamachan ディレクトリの内容を見るには,

```
ls -la /home/tamachan
```

と入力します．実行結果は,

```
drwxr-xr-x   5 tamachan  mikilab      1024 Apr 19 03:54 .
drwxr-xr-x  60 root      root         2048 Apr 13 13:24 ..
drwxr-xr-x   7 tamachan  mikilab      1024 Jan 16 02:45 public_html
-rw-r--r--   1 tamachan  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tamachan の状態で，tamachan 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/tamachan/unix
```

とします．すると，tamachan 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tamachan の状態で，tamachan 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,

```
rmdir unix
```

とします．すると，tamachan 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.

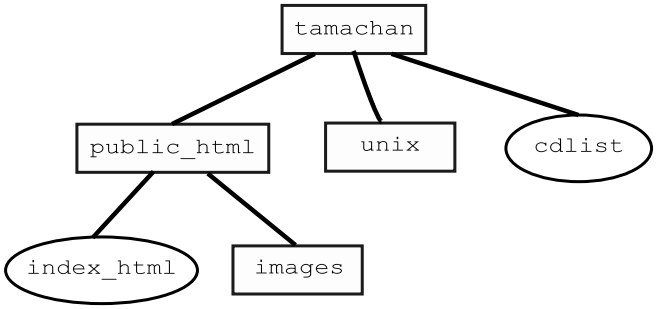

図 3.3: mkdir unix

これらの設定を各ファイル, 各ディレクトリごとに設定できます．ただし, ファイルに対しての設定とディレクトリに対する設定では意味が異なります．ディレクトリ, ファイルに対してのパーミッションの設定は，どちらも読み/書き/実行が行えます．ファイルに対しての読み/書き/実行はそのままですが, ディレクトリに対する読みは，そのディレクトリにどんなファイルがあるかを見ることで，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．
では実際にファイル，ディレクトリがどのような設定なのかを見てみます．あるフォルダに対して，先ほど説明した`ls -l` コマンドを実行した結果を図 3.4 に示します．

```
tyoshida@mikilab:~$ ls -l
total 5
-rwxr-xr-x    1 tyoshida mikilab      0 Apr 10 21:40 b
-rw-r--r--    1 tyoshida mikilab     12 Apr  8 21:06 d
drwxr-sr-x    2 tyoshida mikilab   1024 Apr 10 23:33 takeshi
-rw-r--r--    1 tyoshida mikilab     28 Apr  8 21:17 test1
drwxr-sr-x    2 tyoshida mikilab   1024 Apr  8 19:52 vazeana
drwxr-sr-x    2 tyoshida mikilab   1024 Apr  8 19:52 yoshida
```

合計ブロック数
ハードリンク数
所有グループ
更新日時
ファイル名
パーミッション
所有者
ファイルサイズ
ファイルの種類

図 3.4: ls -l の結果

図 3.4 の各項目について説明します．

- **合計ブロック数**

  一覧に表示されたファイルやディレクトリが占めるディスクの容量がブロック数[5]で表示されます．

- **ファイルの種類**

  一文字でファイルの種類を示します．ディレクトリであれば”d”，通常のファイルであれば”-”になります

- **パーミッション**

  ファイルやディレクトリに対してどの程度のアクセスが許可されているのかを示します．”r”は読み込み，”w”は書き込み，”x”は実行ができることを示します．最初の 3 文字はそのファイルの所有者，次の 3 文字はグループのメンバー，最後の 3 文字はその他のユーザーのアクセス権を示しています．たとえば図 3.4 に示すファイル`test1`は，所有者である`tyoshida` にだけ読み込みと書き込みが許されており，グループユーザーである`mikilab` とその他のユーザーには読み込みだけが許されていることを示しています．

[5]ブロックのサイズはシステムによって異なります．Linux の標準的なファイルシステムでは，1024〜4096 バイトが 1 ブロックになります．

なお，上記の 4 種類の文字 (d,r,w,x) の他に，図 3.4 には”s” という文字が用いられていますが，これについては高度な内容のため，ここでの説明は割愛します.

- **ハードリンク数**

  ファイルやディレクトリにリンクされているハードリンクの数を示します．ハードリンクとは Windows でいうショートカットみたいなものです.

- **所有者**

  ファイルやディレクトリの所有者のユーザ名を示します．通常はファイルを作成したユーザーになります.

- **グループ**

  ファイルやディレクトリを所有するグループ名．通常は，ファイルを作成したユーザーが現在所属しているグループになります.

- **ファイルサイズ**

  ファイルやディレクトリのサイズをバイト単位で示します.

- **更新日時**

  ファイルやディレクトリの更新日時を示します.

- **ファイル名**

  ファイルやディレクトリの名前を示します.

また r（読み込み），w（書き込み），x（実行）は，ファイルに適用される場合とディレクトリに適用される場合とで，多少意味が異なっています．違いを表 3.2 に示します．ちなみに，皆さんは，mikilab グループですので，これら

表 3.2: ファイルとディレクトリのパーミッションの違い

パーミッション	ファイル	ディレクトリ
r	内容を読み込むことができる	ディレクトリの内容を`ls` コマンドのなどで表示できる
w	内容を変更したり，書き換えることができる	ディレクトリ内にファイルを作成したり，削除したりできる.
x	ファイルを実行できる（実行可能ファイル）	ディレクトリにアクセスできる

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 takeshi    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 takeshi    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 takeshi    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

では次にパーミッションを設定してみます．パーミッションを変更するのが，chmod というコマンドです．このコマンドを利用する形式は

```
chmod [モード][ファイル名]
```

となります．ただしモードを指定する方法には，次の 2 通りがあります．

シンボリックモード

それぞれ表 3.3 に示す記号が割り当てられています．たとえばファイルtest1 を所有者以外読み出し禁止にするには，

表 3.3: 記号の割り当て

対象ユーザー	u（所有者），g(グループ)，o(その他)，a(すべて)
操作	+(追加)，-(削除)，=(設定)
保護モード	r(読み込み)，w(書き込み)，x(実行)

```
chmod go-r test1
```

と入力します．

オクタルモード

オクタルモードは 8 進数によるモードの指定方法です．

```
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．これは 2 進数で考えるとわかりやすいです．例えばrw-というパーミッションが設定されていれば，2 進数で110 となり，これを 10 進数に直すと 6 となります．オクタルモードにおける数字を表 3.4 にまとめます．

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．

表 3.4: 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

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 3.5 を例に説明します．

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

```
cp javajava/star.class public_html
```

とします．コピー後は，図 3.6 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

```
cp cdlist cdlist2
```

実行後は，図 3.7 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 3.8 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．

3.5.5 ファイルの名前変更

ファイル名を変更したいときにも，mv コマンドを使用します．コマンドの形式としては，次のようになります．

```
mv [変更前のファイル名] [変更後のファイル名]
```

この方法で，ディレクトリ名を変更することもできます．入力形式も同じです．

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．

```
[takeshi@mikilab ~]$ passwd
Changing password for tyoshida
(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．

なお，マシンによっては，passwd コマンドではなく，yppasswd コマンドでパスワード変更する必要があるので，パスワード変更の際は，管理者に問い合わせてください．以下に，yppasswd コマンドを使用した場合のターミナル上の表示例を示しておきます．(太字はキーボードからの入力です．)

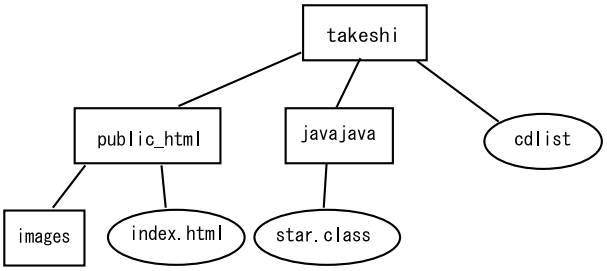

図 3.5: cp example

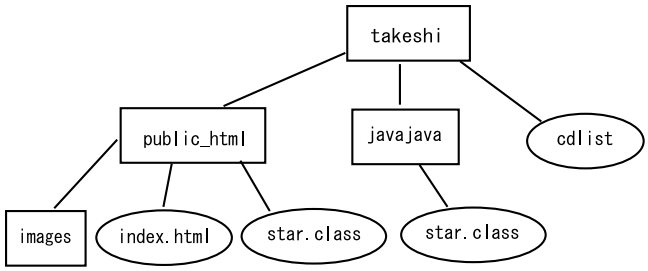

図 3.6: cp file dir

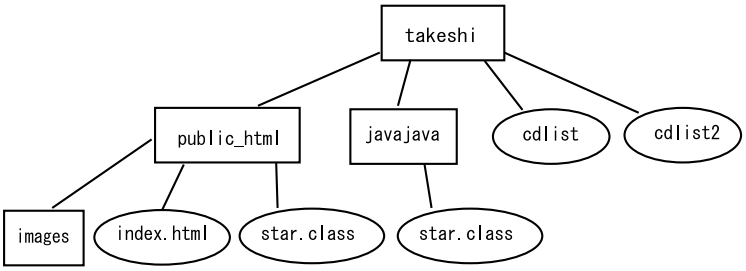

図 3.7: cp file file

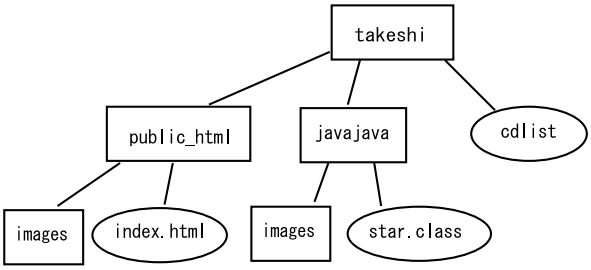

図 3.8: cp -r dir dir

```
[kawasaki@fraulein hoge]$ yppasswd
Changing NIS account information for hoge on fraulein.
Please enter old password:Ys78kjd
Changing NIS password for hoge on fraulein.
Please enter new password:test
The password must have at least 6 characters.
Please enter new password:sample
The password must have both upper and lowercase letters, or non-letters.
Please enter new password:sRtt47HE
Please retype new password:sRtt47HE


The NIS password has been changed on fraulein.


[kawasaki@fraulein kawasaki]$
```

3.6.3 find

目的のファイルを探すコマンドです．入力形式は以下のようになります．

```
find [基点となるディレクトリ名] -name [ファイル名] -print
```

このコマンドで見つけられたファイルはパスネームで表示されます．図 3.7 でstar.class をホームディレクトリ ( ) の下で探してみると次のようになります．

```
[takeshi@mikilab ~]$ find ~ -name star.class -print
/home/takeshi/javajava/star.class
```

3.6.4 man

UNIX ではオンラインマニュアルが用意されています．このマニュアルを検索，表示するコマンドが man コマンドです．形式として次のようになります．

```
man [コマンド名]
```

実際に使用してみて，重要さを実感してみてください．

3.6.5 |(パイプ)

パイプ機能とは，あるコマンドの標準出力を，別のコマンドの入力にすることです．これはUNIX のシェルが提供するプロセス間通信の方法のひとつです．例えば次のコマンドの場合，

```
$ ls -l | more
```

ls -l コマンドの出力をmore コマンドの入力にすることによって，ディレクトリやファイルの一覧を一画面ごとに表示することができます．よく目にするコマンドにはmore がありますが，UNIX ではいくつものコマンドを|記号でつなぎ，コマンドからコマンドへと出力を次々に渡していくことができます．