初心者用
UNIX 講座

第 1 版 1999 年 5 月 2 日

文責：斎藤輪太郎・鷲尾映太郎

1.　　はじめに

UNIX はワークステーション上（最近は PC 上も含む）で動く OS（基本ソフト）の名前です。よりかみくだいて言えば、コンピュータとヒトが対話するための基本的なコンピュータ環境です。他に OS で有名なものとしては、MS-DOS,WINDOWS95 などがあります。OS によってコンピュータの基本的な操作方法が異なります。この講座では UNIX のごく基本的な知識を身につけることを目標とします。

 マークのところではより詳しいコマンドの使い方を解説しています。すぐに全部読んで理解する必要はありません。より詳しい使い方を後で知りたくなったときに利用するといいでしょう。

2.まずはファイルの概念から

コンピュータでは当然の事ながらデータを扱います。これはUNIXの場合でも例外ではありません。そのデータを効率良く扱うためにはデータの性質ごとにうまく整理する必要があります。例えば日常生活では関連のある書類同士をファイルにまとめて入れたりします。コンピュータの世界でいうファイルはこの書類を入れておくファイルと非常によく似た性質を持っています。つまり、データの集合ということです。コンピュータの世界ではファイルとは、データを効率よく整理するためのデータの集合です。

ではさっそくファイルを作ってみましょう。UNIXが動いているコンピュータの前に座って下さい。次のように打ち込んでみましょう（%はプロンプトですから、打ち込まないように。コンピュータによって#だったり、$だったり、hostname:(time):だったりします。）。

```
% cat > test1.txt
Hello, everyone!
Let's enjoy UNIX!
(ここで CTRL ボタンを押しながら d ボタンを押します。)
```

そして次のように打ち込んで下さい。

```
% ls
```

すると、test1.txt という文字がどこかに現れるはずです。これが今作ったファイルのファイル名です。つまり cat > test1.txt は test1.txt というファイル名のファイルを作るためのコマンドです。

では次のように打ち込んでみましょう。

```
cat test1.txt
```

すると、次のような出力が得られるはずです。

Hello, everyone!
Let's enjoy UNIX!

これがファイルの中味です。つまり test1.txt は上のようなアルファベットのデータの集合を管理するファイルだということができます。

ファイル test1.txt

Hello, everyone!
Let's enjoy UNIX!

ちなみにこの場合の cat はファイルの中味を表示するコマンドです。先ほど出てきた ls は（ちょっと正確さに欠ける言い方をすれば）そこにあるファイルのファイル名の一覧を出力するコマンドです。

cat コマンド

機能 ファイルの中味を表示する。

書式 cat　［オプション］　［ファイル］

オプション -n　行番号を付ける。(-b との併用で空行以外に行番号付ける

-s　連続した複数の空行を 1 行にして表示。ﾌｧｲﾙが存在しなくてもメッセージは出力しない。

-u　バッファリングしない。

-v　非印字文字等を識別可能な状態で出力。

-v　と同時に使用するものとして以下のオプションがある。

-e　行末に $ を付ける。

-s　読み込みが出来ないファイルについてもメッセージは出力しない。

-t　タブを　^I で出力、用紙送りを　^L で出力する。

例）cat ﾌｧｲﾙ　　cat　ﾌｧｲﾙ 1　ﾌｧｲﾙ 2

↓　　↓

ファイルの内容を表示　　ファイル 1 とファイル 2 を結合させて表示

```
cat　>　exfile.dat
abc
def
(CTRL+d)
```

↓

標準入力からのﾃﾞｰﾀを exfile.dat にかき出す。

 ls コマンド

機能 指定されたディレクトリ内に存在するファイルの一覧を表示する

書式 ls　　［オプション］　［ファイル］

　　オプション　-F ディレクトリ、実行ファイルなどを区別できるようにする。

　　-R サブディレクトリ以下の一覧も表示

　　-a ピリオド”.”で始まるファイル名を持つファイルも表示

　　-l ファイル情報の詳細。

課題：上の例にならって test2.txt,test3.txt という名前のファイルを作ってみましょう。中味は何でもかまいません。

3.ファイルのコピー

cp コマンドは同じ内容のファイルをもう 1 つ作ります。これをファイルのコピーといいます。まずは testcp.txt というファイル名のファイルを作りましょう。中身は何でもいいです。そして cat コマンドで中味を確認しましょう。

例:

```
% cat testcp.txt
This is a pen.
```

次にこのファイルをコピーします。以下のコマンドを打ち込んでみて下さい。

```
% cp testcp.txt testcp2.txt
```

次に ls でファイルがコピーされたか確認します。状況によりますが、以下のような出力が得られるはずです。testcp2.txt というファイルが新しくできていますね。

```
% ls
testcp.txt     testcp2.txt
```

そして testcp2.txt の内容を確認しましょう

```
% cat testcp2.txt
This is a pen.
```

testcp.txt と全く同じ内容であることが確認できましたね。

課題：同じ要領で testcp.txt と同じ内容のファイル testcp3.txt,testcp4.txt を作成し、cat コマンドで内容を確認しましょう。

4.ファイル名の変更

ファイルの内容を変更せずにファイル名だけを変更する手段があります。先程の testcp.txt のファイル名を変更してみましょう。以下のように mv コマンドを実行してみましょう。

```
% mv testcp.txt testmv.txt
```

そして ls コマンドでファイルの一覧を確認しましょう。例えば次のような出力が得られるはずです。

```
% ls
testcp2.txt testcp3.txt testcp4.txt testmv.txt
```

testcp.txt というファイルが消え、testmv.txt というファイルができているのが分かりますね。testmv.txt の内容を確認しましょう。

```
% cat testmv.txt
```

testcp.txt と内容は同じですね。

cp の場合はもとのファイル(testcp.txt)が残りましたが、mv の場合はもとのファイルが残らないことが分かります。

課題：同じ要領で testmv.txt のファイル名を、testmv2.txt に変えてみましょう。そしてファイルの内容を cat で確認しましょう。

5.ファイルの削除

ファイルを削除するには rm コマンドを使います。例えば ls でみて testcp4.txt がある状態で、

```
% ls
testcp2.txt testcp3.txt testcp4.txt testmv2.txt
```

testcp4.txt を消すには、

```
% rm testcp4.txt
```

とします。そして ls を見ると、testcp4.txt が消えているのが分かりますね。

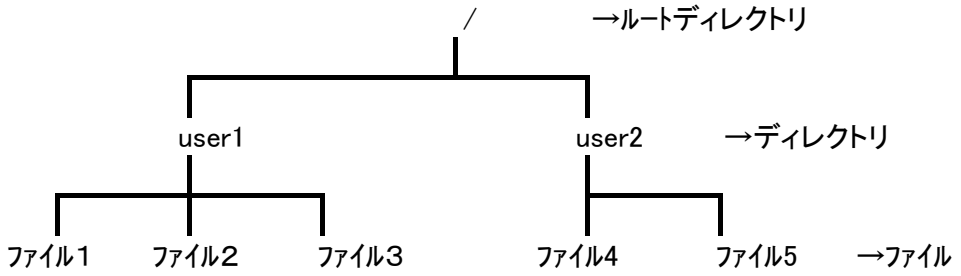
```
% ls
testcp2.txt testcp3.txt testmv2.txt
```

注意： 一度削除したファイルはもう元には戻りません。rm コマンドを使う時は誤って大切なファイルを削除しないように細心の注意を払って下さい。

6.ディレクトリの概念

ファイルの数が多くなってくると、ls をしたとき、たくさんのファイルがいっぺんに表示され、大変ですね。またいろんな目的のファイルが一ヶ所に混在しているのも管理上不便です。そこで UNIX をはじめ多くの OS ではディレクトリという概念を提供しています。

・ディレクトリはいくつかのファイルを入れておくための入れ物です。
・ディレクトリの中に別のディレクトリを入れることができます。
・上記の性質を使うとディレクトリの中にディレクトリを入れ、その中にさらにディレクトリを入れることができるので、下のようなディレクトリとファイルの階層構造ができます(後ほどまた説明します)。



まずディレクトリをつくりましょう。

```
% mkdir testdir
```

ls -F で testdir というディレクトリができていることを確認しましょう。ディレクトリには"/"がつきます。

```
% ls -F
testdir/    testcp2.txt testcp3.txt testmv2.txt
```

ディレクトリ testdir の中を見てみましょう。まず cd で testdir の中に移ります。

```
% cd testdir
```

ls で中身を確認してみましょう。何もありませんね。

```
% ls
%
```

testdir の中でファイルを作ってみましょう。

```
% cat > nicefile
It's very nice today.
(Ctrl-d)
```

testdir の中のファイルを確認してみましょう。

```
% ls
nicefile
```

これで testdir というディレクトリの中に nicefile というファイルが作られたことが確認できました。

ここで以下のように打ち込んでみましょう。

```
% pwd
```

すると例えば以下のような出力が得られるはずです。

```
/home/rsaito/testdir/
```

次に以下のように打ち込んでみましょう。

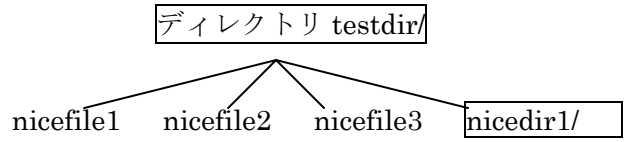
```
% cd ..
```

今度は何も表示されませんが、これはディレクトリを 1 つ上に上がるコマンドです。この場合、testdir から抜け出します。次にまた pwd とうちこんでみましょう。今度は以下のような出力が得られるはずです。

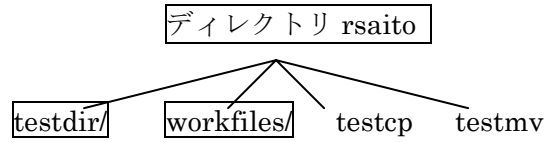
```
/home/rsaito
```

これらの例から類推できるように、pwd は自分が現在見ているディレクトリがどこにあるかを表示するコマンドなのです。ちなみに自分が現在見ているディレクトリのことをカレントディレクトリ(current directory)といいます。

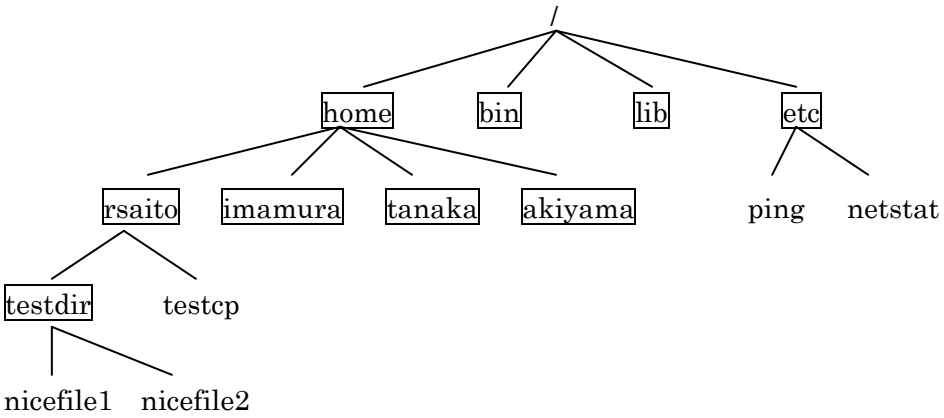
さてここでディレクトリ構造についてもう一度考えてみましょう。
ディレクトリ testdir の中にはファイルや他のディレクトリを入れることができます。



さらにその testdir もまた別のディレクトリに含まれているディレクトリなのです。下の例では testdir は rsaito というディレクトリに含まれていることになります。



このようにディレクトリは階層構造をしているのです。一番上のディレクトリは root ディレクトリ(/)と呼ばれ、その下に下方に向かって枝を伸ばすような構造になっています。



ルートディレクトリを基点として下のディレクトリまたはファイルを指定するには上下関係にあるディレクトリを/で区切って表します。例えば上の例で testcp を指定するには /home/rsaito/testcp と表します。
下のコマンドは自分が見ているディレクトリを testdir にします。

```
% cd /home/rsaito/testdir
```

下のコマンドは/home/rsaito/testcp の内容を表示します。

```
% cat /home/rsaito/testcp
```

このように/を基点としたファイルやディレクトリの指定表現を絶対パスといいます。
これに対してカレントディレクトリ内のファイルを処理の対象とするときはファイル名のみの指定で十分です。例えば上の例では/home/rsaito のディレクトリにいるときは、

```
% cat testcp
```

で十分なのです。

課題：まず ls でカレントディレクトリにどのようなファイルがあるか確認しましょう。その次にその中からファイルを１つ選び、絶対パスによる cat、ファイル名のみによる cat を実行してみましょう(cat で見ることができるのは文字ファイル､つまりテキストファイルだけです)。

7.ファイルの移動

あるファイルを別のディレクトリへ移動させたい場合はファイル名の変更の時に使った mv コマンドが使えます。mv コマンドは次の書式であるファイルを別のディレクトリへ移動させることができます。

mv ファイル名 すでに存在するディレクトリ名

例えば以下のコマンドはカレントディレクトリにある testcp を testdir に移動します。

```
% mv testcp testdir
```

これはもちろん絶対パスで指定することもできます。

```
% mv /home/rsaito/testcp /home/rsaito/testdir
```

この結果、testdir には testcp が入ります。以下のようなコマンドで確認できるでしょう。

```
% ls -F
testdir/                ← testcp がなくなっている
% cd testdir
% ls
testcp                  ← testcp がここにあった。
%
```

課題 : まず mvfile1 というファイルと mvdir1 というディレクトリをカレントディレクトリに作成しましょう。そして mvfile1 を mvdir1 に移動しましょう。

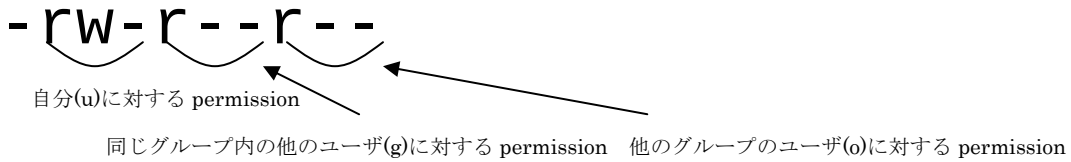
8.ファイルの permission

全てのファイルには permission という属性があります。permission は各ユーザに対してファイルの読み込み(r)、書き込み(w)、実行(x)の権限を与えます。以下のコマンドでファイルの詳細を見ることができます。

```
% ls –l
-rw-r--r--   1 report   report       32 Mar 28 15:51 test1.txt
-rw-r--r--   1 report   report       14 Mar 28 15:58 test2.txt
-rw-r--r--   1 report   report       26 Mar 28 15:55 test3.txt
-rw-r--r--   1 report   report       14 Mar 28 15:59 testcp3.txt
-rw-r--r--   1 report   report       14 Mar 28 16:00 testcp4.txt
-rw-r--r--   1 report   report       14 Mar 28 15:57 testmv2.txt
```

permission　所有ユーザ　所有グループ　ファイルの大きさ　ファイルが最後に更新された日時　ファイル名

“-rw-r—r—“のようになっているところが permission を表しています。test1.txt の permission について見てみましょう。



r は該当するファイルが読み込み可能であることを示しています。
w は該当するファイルが書き込み可能であることを示しています。
x は該当するファイルが実行可能であることを示しています。

読み込み可能であるとはどういうことか、確かめていきましょう。
まず、modtest.txt というファイルを作りましょう。

```
% cat > modtest.txt
Hello!
(CTRL-D)
```

ls –l で permission を確認しましょう。

```
% ls –l modtest.txt
-rw-r--r--   1 report   report       32 Mar 28 15:51 modtest.txt
```

cat でファイルの中味を見ることができることを確認しましょう。

```
% cat modtest.txt
Hello!
%
```

次に chmod コマンドを使ってファイルの permission を変更します。

```
% chmod u-r modtest.txt
```

これは自分(u)の該当ファイルを読み込む権限(r)をなくします(-)。ls で permission を確認しましょう。

```
% ls –l modtest.txt
--w-r--r--   1 report   report       32 Mar 28 15:51 modtest.txt
```

r が 1 つ消えていますね。これは自分に該当するファイルを読む権限がなくなったことを示しています。試しに cat で modtest.txt を見てみましょう。

```
% cat modtest.txt
cat: cannot open test2.txt: Permission denied
```

ファイルを見ることができませんね。
ファイルを読む権限をもとに戻すには、

```
% chmod u+r modtest.txt
```

とします。ls –l modtest.txt, cat modtest.txt でファイルを読む権限が回復したことを確認しましょう。

課題：modtest2.txt というファイルを作成し、読む権限をなくしてみましょう。ls や cat コマンドを使って permission の確認を行って下さい。

次に書き込み(w)のパーミッションを変更してみましょう。ファイルを読む権限があることを ls –l コマンドで確認してから、

```
% chmod u-w modtest.txt
```

として、ls –l でパーミッションを確認しましょう。

```
% ls –l
-r--r--r--   1 report   report       32 Mar 28 15:51 modtest.txt
```

今度は w が 1 つ消えていることを確認しましょう。そして、

```
% cat > modtest.txt
```

でファイルの上書きを試してみましょう。できませんね。

```
% cat > modtest.txt
modtest1.txt: cannot create
```

これは modtest.txt というファイルに書き込む権限(w)が失われたからです。この権限をもとに戻すには、

```
% chmod u+w modtest.txt
```

とします。上書きをしてみましょう。今度はうまくいきますね。

```
% cat > modtest.txt
Hello!
(Ctrl-d)
```

課題：modtest3.txt というファイルを作成し、書き込む権限をなくしてみましょう。ls や cat コマンドを使って permission の確認を行って下さい。上書きなどを試してみて下さい。

上記と同様に自分のファイルを他のユーザが読んだり、書いたりできないように設定できます。

```
% chmod go-r file
```

は読む権限をなくし、

```
% chmod go-w file
```

は書き込む権限をなくします。実際に他のユーザに確かめてもらいましょう。

9.テキストファイルの表示と文字列検索

ここで２つ便利なコマンドを紹介しましょう。

more　　このコマンドはテキストファイルの中味を表示します。cat コマンドによく似ていますが、ファイルがとても大きい場合、一頁ごとに表示を止めてくれます。そこでスペー

スを押すと画面が進みます。使い方は、

more ファイル名

です。実際に１００行くらいのファイルを cat と more を使って表示させてみましょう。

grep　　このコマンドはある文字列を含む行を表示します。使い方は、

grep 検索文字列 ファイル名

です。実際に以下のようなファイルを作り、試してみましょう。まず以下のようなファイル swallows.txt を作ります。

```
Iida      03-1234-5678
Ikeyama   03-9876-5432
Furuta    0426-1111-2222
```

そして以下のコマンドを実行してみましょう。

```
grep  Furuta  swallows.txt
```

すると以下のような出力が得られると思います。

```
Furuta    0426-1111-2222
```

grep コマンド

機能 指定されたファイルの中で検索文字列に一致する行を表示する。
書式 grep　　［オプション］　[検索文字列]　［ファイル］
　　オプション　　-e 複数のパターンの指定(-e 検索文字列 1 –e 検索文字列 2 .....)。
　　-i　アルファベットで大文字と小文字を区別しない。
　　-n　行番号を付ける。
　　-v　一致する行以外を全て表示。

課題：上記ファイル swallows.txt の中で東京に住んでいる(電話番号が 03 から始まる)人を grep で表示させてみましょう。

10.　プロセス

ずっとファイル関係の話が続きましたが、ここでちょっと話題を変えてプロセスの話をします。

プロセスとは UNIX 上における仕事の実行単位のことです。例えば ls と打てば ls というプロセスが走ることになり、pwd と打てば pwd というプロセスが走ることになります。ただしこれらのプロセスはほとんどの場合、あっという間に終わってしまうでしょう。では長く走るプロセスの例をみてみましょう。

X-Window の環境で、

```
% xclock –update 1
```

と打ち込んでみましょう。時計は出ましたか？（環境によっては出ないこともあります。以下は時計が出たという前提で話を進めます。）これで xclock という時計を表示させるプロセスが走っている状態です。ここで他の term を開いて(term の開き方はシステムによってちがいます。周りの詳しい人に尋ねましょう。)以下のコマンドで確認してみましょう。

```
% ps –ef | more
```

環境によっては以下のコマンドの方を使います。

```
% ps –ax | more
```

たくさん行が出てきますが、よく探すとどこかに xclock という文字が含まれている行があると思います。これは xclock がプロセスとして走っていることを意味します。ps –ef は走っているプロセスの一覧を表示するコマンドなのです。

```
% ps –ef
    UID   PID  PPID C STIME  TTY      TIME COMMAND
   root     0     0  0  Jan  1  ?       0:05 swapper
   root     1     0  0  Apr  2  ?       0:00 init
   root     2     0  0  Apr  2  ?       0:00 vhand
   root     3     0  0  Apr  2  ?       0:00 statdaemon
   root     7     0  0  Apr  2  ?       0:00 unhashdaemon
                       :
                       :
 rsaito 10603 10549 0  17:10  pts/0    0:03 xclock –update 1
```

実にたくさんのプロセスが走っていますね。PID の行はプロセスＩＤと呼ばれるものです。例えば上の出力例によると、xclock は 10603 というプロセスＩＤを与えられています。

ls や pwd などのコマンドも実行中はプロセスＩＤを与えられています。

プロセスＩＤを使ってプロセスを止めることができます。書式は、

kill プロセス ID

です。例えば上の例で xclock のプロセスを止めるには、

```
% kill 10603
```

とします。

注意： 初心のうちは危険なので必ず詳しい人に見てもらいながら練習して下さい(とくにPIDの隣にある数字PPIDをkillしてしまうのは大変危険です)。ここではプロセスおよびプロセスＩＤの概念だけつかんでおいて下さい。

課題：X-Window 環境で `ls -lR /`と打ち込みましょう。次に他の term でこの ls のプロセスＩＤを調べましょう。さらに UNIX に詳しい人に見てもらいながらこのプロセスを kill しましょう。

11　出力ファイル

基本的に pwd や ls などのコマンドの実行結果は画面に出力されます。しかし、> の記号を使うことにより、コマンドの実行結果を指定したファイルに格納することができます。書式は以下の通りです。

コマンド　>　出力ファイル名

例えばカレントディレクトリにあるファイル名の一覧をファイル outfile に書き込むには、

```
% ls -l > outfile
```

とします。cat で outfile の中味を確認してみましょう。

注意： すでにoutfileというファイルが存在していた場合、古い内容は上書きされてしまいます。もし古いoutfileの内容にls –lの結果を追加したい場合、以下のように>を２つ続けて書きます。

```
% ls -l >> outfile
```

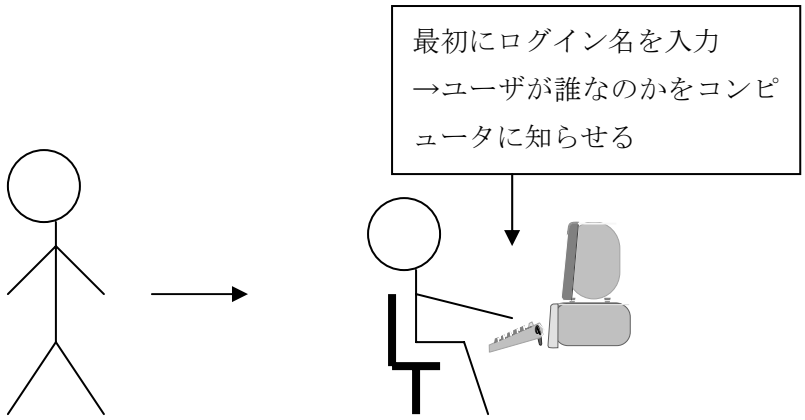
課題：date コマンドの出力結果を date.res に格納して下さい。
課題：cat コマンドと > を使って上記の date.res を date.res.cp にコピーして下さい(cp コマンドは使わないこと)。

12　ユーザについて

UNIX システムはパソコンと違って基本的に複数のユーザが使用することを前提として作られており、各ユーザごとに使用できる資源(resource、例えばディレクトリなど)が割り当てられています。従ってコンピュータの方ではユーザの情報を管理しておく必要があります。アカウントとはコンピュータに管理されるユーザの情報で、銀行に例えると口座です。

ユーザのアカウントの中でも重要な情報は**ログイン名**と**パスワード**、**ホームディレクトリ**です。

・ログイン名はコンピュータによって認識されるユーザの名前です。
・パスワードはユーザがコンピュータの使用を始めるときの暗証番号です（実際は数字だけでなく、アルファベットを使用する事もできます)。銀行の ATM の暗証番号と概念的に似ています。ちなみにログイン名、パスワードを指定してコンピュータの使用を始めることをログインするといいます。
・ホームディレクトリはログインした直後のカレントディレクトリです。各ユーザごとに与えられているディレクトリと考えることもできます。



コンピュータの使用開始

コンピュータにログインしてみましょう。そして、

```
% whoami
```

とするとあなたのログイン名が出てきます。またログイン直後に、

```
% pwd
```

と打ち込むと、ホームディレクトリが表示されます。

練習問題：

☐ ファイルの中味を見るコマンドを２つ挙げて下さい。

☐ 同じ内容のファイルをもう１つ作るコマンドは？

☐ ディレクトリとは何か？

☐ カレントディレクトリにあるファイル名を表示するコマンドは？

☐ ファイルの場所を移動するコマンドは？

☐ カレントディレクトリを移動するコマンドは？

☐ ファイルの permission の概念について説明して下さい。

☐ grep はどのような働きをするコマンドですか？

☐ システム上で走っているプロセスの一覧を見るコマンドは？

☐ 記号”>”にはどのような役割がありますか？