UbuntuでのCUIの使用

岡野浩三

2025年4月12日

1　演習上の注意

一般に**問**の答は唯一とは限らない．また簡単な問もあれば難しい問もある．行間を読んで,「今なぜこの問があるのか」をよく考えた上で，回答することを期待する．

演習は実際に手を動かして，課題を行うこと．レポートでは実行結果やプログラムリストを掲載するだけでなく，考察を記述すること考察で何が書かれてるかが採点のポイントとなる．

2　はじめに —Ubuntu の CUI—

近年の OS では CUI の重要性は小さくなってきているが，CUI を知っていると作業の効率性が上がることもある．例えば，メタキャラクターを用いたパターン表現と CUI コマンドと組み合わせて，共通した名前パターンを持つ大量のファイルを一度に処理できるなどが CUI では容易にできる．このように CUI の特徴をうまく使えば抜群の「切れ味」を見せてくれる．GUI だけしか使えない人と CUI も使える人とではこういうところで差が出る．Windows や MacOS のように本来 GUI を主体とする OS でも CUI はサポートしており Windows では Command Prompt や PowerShell が, また，MacOS では Terminal から CUI を使うことができる．別の例として UNIX 系のサーバーをリモートで管理する状況では CUI を使えるか否かは管理作業の効率性に大きく影響する．省資源で動作することが要求される最新の IoT 端末も Linux ベースのものがある．これらと通信する手段が terminal からの CUI になっていることも多い．UNIX ベースの CUI が登場して 50 年近く経過しているが，まだまだ CUI は根強く残っている．

3　コマンド入力

注意: この資料ではコマンド入力時にそれがシェル経由で人間が入力するものということを強調するためにプロンプトとして$を明示している．シェル変数の先頭記号とまぎらわしいが，これはプロンプトなので当然入力する必要はない．

例 1
例えば以下の記述は

ユーザが `ls` とのみ入力し，`$`や `foo bar` はコンピュータの出力であることを意味している．

```
$ls
foo     bar
$
```

多くの学生は日本語キーボードを使用していることであろう．US Ascii キーボードにおける[\]は日本語キーボードでは[¥]になる．また UNIX の CUI などで比較的使われる[~], [‘](逆シングルクォート，シングルクォート[’]とは向きが逆), [|]などのキーが比較的わかりづらいところに配置されている可能性がある．

ちなみに岡野はできる限り Ascii-US キーボードを使うようにしている．日本語入力はローマ字変換を使うので日本語キーボードの意義の 1 つであるかな入力を使う必要性をほとんど感じていない．

注意: 以下の各節は WSL 上あるるいは Mac Terminal での実行を仮定している．

4 ファイルとディレクトリの記法

UNIX などで用いるディレクトリ (directory) という用語は Windows や MacOS でいうところのフォルダに相当する．

Unix のファイルシステムは大雑把に次のように定義することができる．

- ルートと呼ばれる特別なディレクトリがただ１つ存在する．
- ディレクトリの下にディレクトリまたは通常ファイルを 0 個以上置くことができる．

すなわちルートディレクトリを根とし，通常ファイルあるいは空のディレクトリを葉とする木構造となる．中間ノードは空ではないディレクトリとなる．木構造における親ノード，子ノードの用語をそのまま適用して，親ディレクトリ，子ディレクトリという言い方をすることがある．

以下の記法を用いる．

/ ルートディレクトリ もしくは パスにおけるディレクトリの区切り

. (ピリオド 1 つ) カレントディレクトリ

.. (ピリオド 2 つ) 親ディレクトリ

~ ユーザのホームディレクトリ

~foo ユーザ名 foo のホームディレクトリ

例 2

`/` ルートディレクトリ ファイルシステムの「根」
`/usr` ルートディレクトリの下にある `usr` という名前のディレクトリ
`/usr/bin` ルートディレクトリの下にある `usr` という名前のディレクトリの下にある `bin` という名のディレクトリ

通常のシステムでは /usr/bin や/bin の下に重要な実行ファイルが置かれている．bin はおそらくは binary の略

通常最初に terminal を起動したときはユーザのホームディレクトリがカレントディレクトリ (今いる作業中のディレクトリ) となる．

5 カレントディレクトリの変更

```
$cd  DirectoryPath
```

無引数の場合は ユーザのホームディレクトリに移動する．引数があるときは DirectoryPath で指定したディレクトリに移動する．

DirectoryPath は相対パス指定と絶対パス指定の両方が可能

- パス指定が / から始まらない場合は相対パス指定
- 相対パス指定では 現在のディレクトリ/DirectoryPath のディレクトリに移動
- 絶対パス指定では DirectoryPath のディレクトリに移動

例 3
現在/etc にいるとして以下のように入力すると/etc/rc.d をあらたなカレントディレクリとする *(そこに移動する)*.

```
$cd rc.d
```

```
$cd /etc/rc.d
```

前者は相対パス指定，後者は絶対パス指定.

例 4
現在 /etc にいるとして以下のように入力すると/home が存在する場合，/home をあらたなカレントディレクリとする *(そこに移動する)*.

```
$cd /home
```

以下は Print Working Directory，すなわち現在作業している (現在いる) ディレクトリ名を表示する．

```
$pwd
```

やや特殊な Directory の指定方法として以下がある．

```
$cd ~
```

は自分のホームディレクトリに移動する.

```
$cd ~user
```

は user という名前のユーザのホームディレクトリに移動する.

例 5
組み合わせた実行例

```
$cd
$pwd
/home/hanako
$cd /usr
$cd bin
$pwd
/usr/bin
$cd ~hanako
$pwd
/home/hanako
$cd /usr/local/bin
$pwd
/usr/local/bin
```

例 6
組み合わせた実行例 *2* 次のように .. を用いることもできる.

```
$cd /usr/bin
$pwd
/usr/bin
$cd ../local
$pwd
/usr/local
$cd ../../home/hanako
$pwd
/home/hanako
```

問 1
いま */usr* にいるとして

```
$cd ./bin
$cd bin
```

の違いは何だろうか？

問 2
いま */usr* にいるとして

```
$cd  ../../home
```

を実行するとどこに行くだろうか？

```
$cd  ../../../home
```

を実行するとどこに行くだろうか？

問 3
`pushd`, `popd` について調べよ．実際に使ってみよ．何が便利だろうか？

6 ファイル一覧表示

```
$ls FilePath
```

FilePath で指定したファイルのファイル名を表示する．FilePath がディレクトリを指定している場合はそのディレクトリに含まれるファイル名やディレクトリ名を表示 (ただし . で始まるファイルやディレクトリは無視).

ファイルパスが指定されない場合はカレントディレクトリが指定されているとする．

7 コマンド入力のお作法

ls に限らずコマンド入力はコマンド名を入力し，必要に応じて引数をスペースで区切って記述する．またコマンドによっては オプションを記述することができる．通常，オプションはコマンド名の直後にスペースのあとに − に引続いて記述する．

なお GNU が開発したコマンドは従来の − から始まるオプション以外に −− (ハイフン 2 つ) から始まるオプションもサポートしていることが多い．また，従来のオプション形式ではオプションの区別がアルファベット 1 文字だけなので覚えにくい・識別しにくい (複数のオプションが使われると，初学者にとっては呪文以外の何物でもないであろう) のに対し，GNU のオプション形式はフルスペルの単語が使われており，オプションの意味が比較的判別しやすい傾向にある．ただし，後者は入力の手間が増えるし，類義単語の存在があるため覚えやすいとも限らない．

例 7
`ls` のオプション例

ls -a . で始まるファイルやディレクトリも表示する

ls -l ロング形式 (さまざまな情報を表示する形式) で表示する

ls -F ファイルやディレクトリなどの区別を表示する

ls にはいろいろなオプションがあり表示を変えることができる．
man コマンドは引数のコマンドの詳細 (manual) を表示する．

問 4
次を実行して `ls` のオプションについて調べよ．どのようななオプションでどのような表示 (情報) が得られるだろうか？大量のオプションのなかで，平時，使ってみたいと思うオプションはどれだけあるだろうか？用途が不明 (何の役に立つか分からない) なオプションはあるだろうか？以上を簡単にまとめよ．

```
$man ls
```

8 alias

さきほどの ls コマンドは Ubuntu の標準設定では正確には `ls --color=auto` にエイリアス設定がされている (ハイフン２つ)．すなわち ls を打つだけで `ls --color=auto` と入力したのと同じ効果があるように内部で読み替えられている．

問 5
`--color=auto` オプションの効果について調べよ．

素の ls コマンドを実行するには

```
$\ls
```

と入力する．

問 6
素の *ls* の表示を確認してみよ．

現在設定されているエイリアスの一覧を見るには以下のように実行する．

```
$alias
```

9 シェル

これらのコマンドを解釈するインタプリタをシェルと呼ぶ．Ubuntu の標準シェルは bash である．
シェルは一般に複数種類存在し，ユーザが好みで変更できる (どうやったら変えられるだろうか?)．
コマンドは一般に実行ファイルとして所定のディレクトリに入っている

典型的な置き場所は

`/bin /usr/bin`

など

シェルはこれらのディレクトリを検索しながらコマンドを特定し，実行する．どのディレクトリをどの順で見ているかは環境変数 `PATH` に格納されている．環境変数の内容をみるには echo コマンドが使える．

```
$echo $PATH
```

注意: 環境変数の名前は `PATH`, 変数の値を参照するときには先頭に$をつける

問 7

次を実行し，内容を確認せよ．いくつのディレクトリが読み取れるだろうか？なおパスの区切りは*:* が使われる．*PATH* に列挙されたディレクトリのリストの先頭，うしろどちらのディレクトリが *(*コマンドの検索順として*)* 優先されるであろうか？それはどのように調べたらよいだろうか？

```
$echo $PATH
```

ヒント: 今，仮に~/bin/foo ~/bin2/foo と同一の名前で違う動作をする 2 つの foo という実行ファイルがあったとする．もし~/bin のパスのほうが /bin2 より優先するのであれば foo とだけ入力したとき /bin/foo の実行ファイルが実行される．

ことなった動作をする foo という実行ファイルは後でまなぶ shell script で簡単に作れるが，以下のように ps コマンドと ls コマンドをコピーするのも一つの方法であろう．

```
$cd ~
$mkdir bin
$mkdir bin2
$cp /bin/ls bin/foo
$cp /bin/ps bin2/foo
$PATH=~/bin:~/bin2:$PATH
$echo $PATH
$hash
$foo
```

注意: PATH=∼/bin:∼/bin2:$PATH は変数 PATH に左辺を設定してる.「=」の左右に空白をいれてはいけない.

問 8

上のコマンド系列で何を実行しているのか解説せよ．また実際に動かしてみよ．

echo コマンドは引数をそのまま出力する．ただし上述の$PATH のようにいくつかの式は解釈してその値を出力する．

問 9
仮に*/bin/foo /usr/bin/foo* と同一の名前で違う動作をする *2* つの *foo* という実行ファイルがあったとする．このとき /usr/bin にある foo コマンドを実行したいときどのようにすればよいだろうか*?* 解を *2* つ以上考えよ．

問 10
環境変数の値を設定・変更するには具体的にどうすればよいだろうか？その変更を常に有効にする*(*毎ログイン時に自動で設定する*)* にはどのようにすればよいだろうか*?* ヒント ~/.profile

参考 1
hash コマンド
コマンドをタイプしたとき，シェルは実行可能ファイルを見つけるために，毎回 変数 PATH で指定されたディレクトリを検索するわけではない．シェルコマンドの内部データとしてハッシュテーブルを保持しており，そこに実行可能なコマンドをシェル起動時にあらかじめ *1* 回だけ変数 PATH を探索して登録してき，その情報をもとにコマンドを呼び出している．

このため，新しくシェルスクリプトを作成したり，実行バイナリをコンパイルした場合，実際はパスが指定しているディレクトリに実行可能ファイルがあるにもかかわらず，そのコマンドがハッシュテーブルに登録されていないため，コマンドを見付けることができない場合がある．このようなときは hash でハッシュテーブルの再構築をすることで，シェルに新しいコマンドを認識させることができる．

参考 2
カレントディレクトリは PATH に設定しない．
昔の教科書では PATH としてカレントディレクトリ *(.)* を含むと，主にプログラム作成をするユーザは，作成したプログラムを即時実行できて便利であるということが記述されていたが，現在ではカレントディレクトリを PATH に設定することを奨励する記述はない．おもにセキュリティ上の理由からである．カレントディレクトリにある実行ファイルを自分の責任で実行する場合は./コマンドのようにコマンド名の前に *./* をつければよい．*(*相対パス指定の応用*)*

10 入力の省力化

過去に入力したコマンドは上向きカーソルキーで検索できる．選択した後，左右のカーソルキーで部分編集することもできる．

またカレントディレクトリ下のファイルやディレクトリは先頭の文字を入力したあと tab キーで補完できる．

これらの機能をうまく活用すると CUI での作業効率化が図れる．

11 新規ディレクトリの作成

```
$mkdir  DirectoryPath
```

DirectoryPath で指定した名前のディレクトリを作る.
注意: 2 階層以上の新規ディレクトリを一度に作ることはできない.

例 8
現在 `foo` というディレクトリがない状態で `mkdir foo/bar` を実行してもエラーになる

問 11
ホームディレクトリ下に OSEX という名のディレクトリを作成せよ. *cd* でそこに移動してみよ. そこで以下のコマンドを実行したのち `ls -F` で確認してみよ. それは期待していた結果だろうか？

```
$mkdir temp0 temp1 os0 os1
```

問 12
この状況で以下を実行するとどうなるだろうか?

```
$mkdir temp0/bar
```

問 13
この状況で以下を実行するとどうなるだろうか? 理由は?

```
$mkdir temp2/bar
```

12 ワイルドカード

通常シェルはワイルドカードを使える.

***** 任意の文字列 ただし.(ドット) から始まる文字列にはマッチしない

? 1 文字の任意の文字 ただし.(ドット) から始まる文字列にはマッチしない

[···] [ ] 内で特定の記法によって指定された文字群

例 9
[*0-9a-zA-Z*] 1 文字の英数字
マッチする文字列は標準ではカレントディレクトリの下のファイル名などになる.

問 14
カレントディレクトリに次のファイルが存在するとき
`main.tex main.txt main.dvi main2.tex`
以下を実行したときに何をしたことになるだろうか？なお　空ファイル *main.txt* を作成したいときは *touch main.txt* を実行するとよい.

```
$ls *.tex
```

問 15
カレントディレクトリに次のファイルのみが存在するとする
`main.txt main.log .main.txt .main.log`

1. 次の実行結果はどうなるだろうか?

   ```
   $ls *.txt
   ```

2. *.txt* で終わるファイルをすべてかつそれらのみを表示したければどのようにワイルドカードを使えばよいだろうか? 引数にワイルドカードを複数指定してもよいものとする.

3. . で始まるファイルをすべてかつそれらのみを表示したければどのようにワイルドカードを使えばよいだろうか? 引数にワイルドカードを複数指定してもよいものとする．このとき親ディレクトリの内容を表示させないようにするにはどうすれば良いだろうか?

注意: main.txt がファイルではなくディレクトリであり，かつ，その下に main.tex という名のファイルがある状況で ls *.txt を実行すると main.tex が出力される．この状況になる理由を説明できるだろうか？一般にディレクトリ名はファイルと紛らわしい名前にすべきではない.

注意: ワイルドカードを解釈し展開するのはシェル自身であって，個々のコマンドではない.

先にシェルがワイルドカードにマッチする具体的な文字列のリスト (ファイル名のリスト) を作成し，そのリストをコマンドに渡す.

13　ファイルの消去

```
$rm FilePath
```

FilePath で指定したファイルを消去する．複数指定することもできる.

今，仮にカレントディレクトリの下に 100 個を超える不要なファイルがあったとする．それらのファイルをすべて一度に消去したければ

```
$rm *
```

と入力すればよい.

問 16
今上述と同じ状況で * という名のファイルが存在して，そのファイルだけを消去したければどのように入力すればよいだろうか?

そもそも*という名前の空ファイルはどのように作成すればよいだろうか？なお、*touch fileName* で *fileName* の名前をもつ空ファイルが作成できる．

なお，*という名前のファイルは ls コマンドで表示すると例えば'*' のように表示されるようである．これは３文字からなるファイルではなく*１文字からなるファイルである．

純粋なファイル名を表示するには例えば以下のように ls に　- -escape オプションをつける (ハイフン 2 つ).

```
$ls --escape
```

14　ディレクトリの削除

```
$rmdir DirectoryPath
```

DirectoryPath で指定した名前のディレクトリを削除する．DirectoryPath に子のファイルやディレクトリがある場合は削除されない．

問 17
子ディレクトリやファイルの有無にかかわらず，指定ディレクトリ以下を一命令ですべて削除したいときにはどうすればよいだろうか？

15　ファイルやディレクトリの移動

```
$mv src dist
```

src で指定したファイルあるいはディレクトリを dist で指定したファイルあるいはディレクトリに移動させる．

問 18
mv コマンド *dist* にファイル名ではなくディレクトリを指定することができる．このときの振る舞いを説明せよ．

mv コマンドは *dist* がディレクトリであれば *src* のファイルを複数指定することができる．このときの振る舞いを説明せよ．

16　ファイルのコピー

```
$cp src dist
```

src で指定したファイルの内容を dist で指定したファイルにコピーする．

問 19
ln コマンドについて説明せよ．また *ln -s* オプションについて説明せよ．

問 20
以下の実行系列について説明せよ．

```
$echo aaaa > a.txt
$cp a.txt b.txt
$ln a.txt c.txt
$ln -s a.txt d.txt
$cat a.txt
$cat b.txt
$cat c.txt
$cat d.txt
$echo bbbb >> a.txt
$cat a.txt
$cat b.txt
$cat c.txt
$cat d.txt
$rm a.txt
$cat b.txt
$cat c.txt
$cat d.txt
$ls d.txt
```

問 21
ディレクトリを対象に cp を実行するとどうなるだろうか？

17　cat コマンド

```
$cat FilePath
```

conCATenate

FilePath で指定したファイルの中身を標準出力に表示する．複数のファイルを指定した場合は順次その中身を表示する．ファイルを指定していない場合は標準入力から読み込む．この場合 1 行単位で出力されるのでオーム返しのような効果が観測できる．標準入力から読み取っている状況で終了するには Ctl-C を用いる．ファイルはバイナリであっても機能する．もちろん標準出力に表示されていても読めないが．

18 標準ストリーム

標準出力

デフォルトの出力: 通常はターミナルの画面**標準入力**

デフォルトの入力: 通常はターミナルのキーボード入力**標準エラー出力**

デフォルトのエラー出力: 通常はターミナルの画面

エラー出力と標準出力は論理的に区別されている．たまたま両方ともターミナルに出力される設定であるがリダイレクトで標準出力のみをファイルに取り込むことができる．

```
$grep Pattern File
```

File から Pattern に一致する行のみとりだし，標準出力に表示する．File を指定しない場合は標準入力を対象とする．

19 リダイレクション パイプ

コマンド < ファイル

コマンドの標準入力を指定ファイルに切り替える．

コマンド > ファイル

コマンドの標準出力を指定ファイルに切り替える．

パイプ 前後のコマンドの入出力をつなげる．

コマンド 1 | コマンド 2

コマンド 1 の標準出力結果をコマンド 2 の標準入力に渡す．

パイプは多段階に設定できる．

問 22

以下は何をしているのか?

```
$cat *.txt| grep tanaka > tanaka.data
```

問 23

`grep` に指定できるパターン *(Pattern)* について調べよ

問 24

1. 次の *test* のようなデータが入っているようなテキストファイルに対してメールアドレス *(*ぽい*)* もののみを含む行を取り出したファイル *result1* を作成したい．このときにどのように *grep* を使えばよいかを示しなさい．

2. 同様に日本の電話番号 *(*ぽい*)* もののみを含む行を取り出したファイル *result2* を作成したい．このときにどのように *grep* を使えばよいかを示しなさい．

3. *grep* のパターンとして以下のパターンを表す正規表現を示し，理由も併せて説明しなさい．

   (a) 日本の携帯電話番号表記は原則 *0* から始まる *11* 桁の数字である。ただし区切り記号として*-*が使われているかもしれないし，区切り記号がないかもしれない（区切り記号として空白や（）は認めないとする）．また日本を表す国際電話の識別子*+81-* が先頭にあるかもしれないし，ないかもしれない．また，国際番号識別子のあとは先頭の *0* はつけず下 *10* 桁が続く．

   (b) 日本の郵便番号にマッチする正規表現 *(*ハイフン付きの *7* 桁のみを対象とする*)*

```
$cat test
a-okano okano@shindai-u.ac.jp
u-tanaka 034-567-6789
nakata nakata@jmail.con
t.yasuda 090-1234-5678
k.okamoto ok@yahho.co.jp
u.aizawa 070-1111-0000
hanako.yamada +81-70-5545-5678
ann.g ann.dona@cc.cc.dd.ee
```

20 ファイルパーミッション

ファイルには読み取り，書き込み，実行の 3 つの権限を 3 つの階層，ユーザ，グループ，他人にわけて設定できる．

ユーザにつけられた権限はそのファイルの持ち主に与えられ，グループにつけられた権限はそのファイルが属するグループに属する任意のユーザに対して与えられる．他人につけられた権限は，任意のユーザに対して与えられる．

chmod 権限を変える．

```
$chmod u+r file
```

file のユーザ (持ち主) にそのファイルの読み取り権限を与える．

```
$chmod u+w file
```

file のユーザ (持ち主) にそのファイル書き込み権限を与える．

```
$chmod  u+x  file
```

file のユーザ (持ち主) にそのファイルの実行権限を与える．
グループに対して変更するときは u の代わりに g を使う．同様に他人に対しては o を使う．
権限の剥奪は + の代わりに - を用いる．
ファイルの代わりにディレクトリに対しても指定できる．この場合 rwx の意味は少し異なる．

問 25
chmod を適当なファイルやディレクトリに対して様々なパーミッションで設定してその設定がきちんと設定されていることと，その設定によりどのような効果があるかを確かめよ．特にディレクトリに対して設定をかえたときにそのパーミッションの変化がどのように影響を与えるか実際に確かめてみよ．例えば，実行パーミッションをなしくしたディレクトリに *cd* で移動できるだろうか？など．

+, − ではなく，直接数値で指定する方法もある．4,2,1 はそれぞれ読み取り可能，書き込み可能，実行可能を意味する．複合権限はこれらの値の加算値を用いる．3 桁の数値の 100 位がユーザ，10 位がグループ 1 位が他人となる．

例 10
以下はファイル *file* に対して，オーナーは読み取り，書き込み，実行が可能，グループは読み取りと書き込みのみ可能，他人は読み取りのみ可能となる．

```
$chmod 764 file
```

一般にセキュリティを重視して他人にそのファイルの内容を書き換えられないよう他人に対しては書き込み権限ははずすべきであるし，さらに秘匿すべきファイルであれば読み取り権限も落とすべきであろう．
権限関係ついては他にも chown, chgrp など (ルート権限でないと実行できない).
任意のユーザは複数のグループに属することができる．

問 26
chgrp の具体的な利用例を説明しなさい．

問 27
あるユーザー *foo* さんをグループ　*wheel* のメンバーに追加したい場合どういう処理をすればよいか説明しなさい．

more ページャ
大量の行数が表示される場合に，1 画面分ごと確認をとりながら読むことができる．

```
$ps  auxg  |more
```

21 プロセス管理

X11 のアプリを実行するために (まだ実行していなければ) 以下のコマンドを実行し X11 をインストールしよう

```
$sudo apt install x11-apps
```

いつものように自分の password を入れてインストールを進めていく.

インストール後，以下の実行をしてみよう.

コマンド実行時に最後に&をつけるとそのコマンドはバックグラウンドプロセスとして実行される．バックグラウンドプロセスとして実行されるのでシェルのプロンプトがそのコマンドの実行時間に無関係にすぐ提示される.

たとえば時計を起動したくて xclock 以下のように通常通り起動する (これをフォアグランドプロセスと呼ぶ).

```
$xclock
```

確かに時計が実行されるが，シェルのプロンプトが帰ってこないので次の作業ができない．ここで Ctl-C を押すと，時計の実行が終了し，プロンプトが戻ってくる．時計を起動し，かつ，次の仕事をそのシェル上で行いたいときには

```
$xclock &
```

とバッググラウンドで実行するとよい．このときは，プロンプトが戻ってきて，次の作業にとりかかれる.

この状態で

```
$ps
```

を実行するとその端末から起動し，現在実行中のプロセスの一覧が表示される．先ほどの時計のプロセスも表示されているはずである.

このプロセス ID を仮に 1005 だとする.

この状態で

```
$kill 1005
```

を実行すると先ほど起動した xclock が終了する.

kill コマンドはこのようにプロセスの強制終了に使われるが他の機能として目的のプロセスにシグナルを送るという機能がある．先ほどの強制終了はこのシグナルを送るという機能を実は使っている．デフォルトのシグナルでは多くのプロセスは終了というように解釈される.

問 28

`kill` コマンドがデフォルトで送るシグナルは何か調べよ.

ps コマンドで自分の terminal で実行したプロセス以外の現在起動しているすべてのプロセスを確認したい場合は

ps auxg というようなオプション群を指定して実行する．

注意: Linux では伝統的に ps のこのオプションは-をつけない．

問 29

`top` コマンドを調べよ．

あるコマンドが実際にどのパス上に存在しているコマンドであるかを調べるには which コマンドを用いる．

たとえば

```
$which ls
```

を実行すると

/bin/ls

が表示され，ls コマンドは実際には /bin/ls が実行されていることがわかる．

一方

```
$which cd
```

を実行すると何も表示されない．これは cd が bash の内部コマンドであることを意味する．

シェルの種類によっては ls なども内部コマンドになることがある．

22 管理者コマンド

UNIX では管理者は root と呼ばれている．

管理者でなければ実行できないコマンドが複数ある．これらは大部分がシステムの設定を変更するようなコマンドである．

root になるには以下の方法がある．

1. root で直接ログインする
2. su コマンドで root になる
3. sudo コマンドを使う．

最初の方法は現在ではセキュリティ上の観点からあまり推奨されない．

2 つ目の su を実行するためには，su コマンドを実行したいそのユーザが wheel グループに登録されている必要がある．

また 3 つめの sudo コマンドを使うためには事前に visudo コマンドを用いて登録する必要がある．

詳しくは以下を参照．

http://www.atmarkit.co.jp/ait/articles/0311/05/news001.html

なお, Ubuntu では sudo コマンドが使える状況でセットセットアップされている．

sudo コマンドでは上述のように sudo の後の引数に本来 root 権限で実行したいコマンドとその引数を記述する.

たとえば電源 off のシャットダウンを行いたいときはは

```
$shutdown -h now
```

というように -h now のオプションを指定して実行するが，一般ユーザーの権限でこのコマンドを実行しても「ルートである必要があります」というエラーメッセージが表示され実行されない.

```
$sudo shutdown -h now
```

というように sudo コマンドを経由して実行するとパスワードを聞かれ，パスワードを入力すると shutdown が無事実行される．パスワードは今実行しているユーザのものを入れる．スーパーユーザーのパスワードを要求されているわけではないことに注意.

同様にリブートをしたければ

```
$sudo reboot
```

を入力するとよい.