

# これまでの復習 (前期中間試験に向けて)

山本昌志\*

2004年6月9日

これまでの、学習をまとめる。中間試験には、このプリントに書いてある内容を理解して臨むこと。中間テストの実施要領は、以下の通りである。

- 教科書は持ち込み可とする。
- 配布したプリントは持ち込不可とする。教科書にプリントのコピーを貼り付けたものは、カンニングと見なす。ただし、教科書への書き込みは可とする。

## 1 重要な UNIX コマンド

- UNIX のファイル構造は、ツリー (木) 構造である。
  - ツリーは、ファイルとディレクトリーからできている。ファイルはデータやプログラムである。ディレクトリー<sup>1</sup>は、入れ物でファイルやディレクトリーを入れる。
  - 今、自分が居るディレクトリーを、カレントディレクトリーと言う。カレントディレクトリーへの絶対パス (位置) を調べるコマンドは、「pwd」である。
  - パスを表す場合、ディレクトリーの区切りには「/」(スラッシュ) を使う。
  - カレントディレクトリーを明示したい場合は、1つのピリオド「.」で表す。
  - カレントディレクトリーの1つ上のディレクトリーを親ディレクトリーと言う。親ディレクトリーへ移動するコマンドは、「cd ..」である。2つのピリオド「..」が、親ディレクトリーを表す。
  - カレントディレクトリーの直ぐ下のディレクトリーを子ディレクトリー、あるいはサブディレクトリーと言う。例えば、子ディレクトリー hogehoge に移動するコマンドは、「cd hogehoge」である。
  - ユーザー各個人が使用 (読み、書き、実行) を許されている最上位のディレクトリーをホームディレクトリーと言う。移動するコマンドは、「cd」である。
  - 2通りのパスの表し方がある。例えば、私のホームディレクトリー (/home/user/yamamoto) にサブディレクトリー (sub\_1)、そのサブディレクトリー (sub\_11) は、次のように表すことができる。相対パスはカレントディレクトリーからの相対位置を表し、ここではホームディレクトリー

\*国立秋田工業高等専門学校 電気工学科

<sup>1</sup>Windows や Macintosh ではフォルダーと言う。

とする。

絶対パス /home/user/yamamoto/sub\_1/sub\_11

相対パス sub\_1/sub\_11 あるいは ./sub\_1/sub\_11

- カレントディレクトリーにあるファイルやサブディレクトリーの名前を調べるコマンドは、「ls」である。
- サブディレクトリーを追加する場合、コマンド「mkdir」を使う。例えば、サブディレクトリーとして hogehoge の追加する場合、コマンド mkdir hogehoge とターミナルに入力する。
- 空っぽのサブディレクトリー (hogehoge) を削除するコマンドは、「rmdir」である。空っぽのディレクトリー hogehoge を削除する場合、「rmdir hogehoge」とする。また、サブディレクトリーに中身がある場合、「rm -rf hogehoge」とするとサブディレクトリーに含まれるもの全て削除される。
- ファイルを削除するコマンドは、「rm」である。例えば、「rm hogehoge」とすると、hogehoge というファイルが削除される。
- ファイルやディレクトリーをコピーするコマンドは、「cp hogehoge hugahuga」である。これで、hogehoge というファイルあるいはディレクトリーの hugahuga という名のコピーが作成される。
- ファイルやディレクトリーを移動させるコマンドは、「mv」である。例えば、「mv hogehoge ../hugahuga」とすると、親ディレクトリー (..) に hugahuga が無い場合、hogehoge が親ディレクトリーに移動して、名前は hugahuga に変更される。もし、親ディレクトリー (..) にサブディレクトリー hugahuga がある場合、その中に hogehoge という名前でも移動する。また、ファイルやディレクトリーの名前の変更にも使われる。
- 以前使用したコマンドを呼び出す機能をヒストリー機能と言う。キーボードの「↑」や「↓」でその機能が使える。同じような長いコマンドを何回も打ち込む手間が省け便利である。
- 重要なコマンドをまとめると、以下のようになる。

---

pwd	現ディレクトリー (カレントディレクトリー) のパス (位置) の表示
ls	ファイルとディレクトリーの表示
cd	ワーキングディレクトリーの移動
mkdir	ディレクトリーの作成
rmdir	空のディレクトリーの削除
cp	ファイルやディレクトリーの複製
mv	名前変更や移動
rm	ファイルやディレクトリーの削除
cat	ファイルの表示や連結
more	ファイルの内容を一画面単位で出力
man	コマンドのオンラインマニュアル
↑ 又は ↓	history。以前のコマンドの表示を行う。編集可能である。
[ctrl]+c	プロセスの強制終了
[Tab]	補完機能

---

## 2 コンパイルと実行

- C 言語のプログラムが書かれたソースファイルには、「hogehoge.c」のように拡張子「.c」が必要である。
- C 言語のソースファイル「hogehoge.c」をコンパイルして、実行ファイル「hugahuga」を作成するコマンドは、次の通りである。

```
数学関数が無い場合 gcc -o hugahuga hogehoge.c
数学関数がある場合 gcc -lm -o hugahuga hogehoge.c
```

- ターミナルに「ディレクトリーを指定して、実行ファイル名 (例えば、./hugahuga)」を打ち込んで [Enter] キーを押せば、プログラムは実行される。

## 3 C 言語

今後、C 言語を用いての数値計算を学習する上で、C 言語の重要な事項をまとめる。

### 3.1 C 言語の基礎

- C 言語では、大文字と小文字は、区別される。変数名 hogehoge と HogeHoge、hoGehoge は異なる。
- コメント文は、プログラムの内容をわかりやすくするために記述するものである。これは、人間のためのもので、コンパイラーは無視する。/\*~\*/で囲まれた部分が、コメント文となる。行をまたいでも、それは有効である。
- 識別子とは、変数、記号定数、関数などにつける名前のことである。名前に用いることができる文字は決まっている。英大文字「A~Z」と英小文字「a~z」、数字の「0~9」とアンダースコア「\_」である。
- C はフリーフォーマットで記述でききるので、文の区切りの記号が必要である。その区切りの記号にセミコロン「;」を用いる。
- コンピューター内部では、\ (バックスラッシュ) と ¥ (円マーク) の取り扱いは全く同じです。
- { と } は対応しており、{ と } で囲まれた部分は、一つの処理のまとまり (ブロック) を表す。
- プログラムは main() 関数から実行される。
- 基本的には、プログラムは上から下へと処理の動作が行われる。

### 3.2 データの型

- 変数は、値を入れておく箱のようなものである。1 つの変数に 1 個の値を入れておく (記憶) ことができる。

- 変数を使う場合、実行文に先立って、その宣言を行う必要がある。
- 変数は定義してから用いなくてはならない。型と変数名を指定することにより、変数の定義ができる。これは、プログラムを実行するときに必要な領域を確保するために必要で、コンパイラーが実行ファイルを作るときに使う。
- 使用頻度が高い型は、以下の通りである。

型名	型指定子	変数宣言例
文字型	char	char a, b;
整数型	int	int i, j;
倍精度実数型	double	double x, y;

- C 言語では、変数の適用範囲は厳密に決められている。ローカル変数とグローバル変数があり、適用範囲が異なる。

ローカル変数      関数の中で定義され、その関数の中だけで使用できる。関数がコールされるとメモリー上に変数が配置される。その関数の処理が終わるとその変数は消滅する。通常、使うのはこれである。

グローバル変数    関数の外で定義され、どの関数でも使用できる。プログラムが起動されるとメモリー上に変数が配置される。プログラムが終了するまで、変数は維持される。

### 3.3 演算子

- C 言語で使われる演算子で分かりにくいものを表 1 にまとめておく。

表 1: 分かりにくい演算子

種類	演算子	機能	使用例	備考
算術演算子	%	剰余 (余り)	c=a%b	除算の余りを計算
関係演算子	==	等しい	if (a==b)	a==b の演算結果は、0 or 1
	!=	等しくない	if (a!=b)	a!=b の演算結果は、0 or 1
論理演算子	!	否定	if (!a)	!a の演算結果は、0 or 1
		論理和 (or)	if (a    b)	a    b の演算結果は、0 or 1
	&&	論理積 (and)	if (a && b)	a && b の演算結果は、0 or 1
代入演算子	=	代入	b=a	右辺の式の値を左辺の変数に代入
	a+=b		b=a	a=a+b と同じ
	a-=b		b=a	a=a-b と同じ
	a*=b		b=a	a=a*b と同じ
	a/=b		b=a	a=a/b と同じ
その他	++	インクリメント	a++	a=a+1 と同じ
	--	デクリメント	a--	a=a-1 と同じ

- 論理演算では、0 が偽 (誤り) で 1 が真 (正しい) となる。
- 0 と 1 以外で論理演算を行った場合、0 のみが偽として取り扱われ、非 0 は真となる。

## 3.4 キーボード入力、ディスプレイ出力

### 3.4.1 scanf() 関数

- キーボードから、値 (文字や数値) を読み込むために、scanf() 関数を使う<sup>2</sup>。読み込んだ値は、変数に格納される。
- scanf() 関数の記述の仕方は、次の通りである。

```
scanf("変換仕様の並び ", &変数名, &変数名, ..., &変数名);
```

- 変換仕様とは、キーボードから入力されたものがどのような値か判断するために使う。主なものは以下の通りである。

1 文字	%c
整数	%d
小数	%lf
指数形式	%e

<sup>2</sup>他にもあるが、これが簡単である。

- 変数名は、先頭に&をつける必要がある<sup>3</sup>。

### 3.4.2 printf() 関数

- ダブルクォーテーションで囲まれた部分 ("出力の並び") に、ディスプレイに表示したい文字列や変数の変換仕様を記述する。変換仕様は対応する変数の出力方法を決めるものである。

```
printf("出力の並び", &変数名, &変数名, ..., &変数名);
```

- 使用頻度の高い変換仕様は、以下の通りである。

1 文字	%c
整数	%d
小数	%f
指数形式	%e

- 整数の表示桁数を調整するときは、変換仕様%dの%とdの間に、表示したい桁数を記述する。
- 少数部の桁を調整するときは、変換仕様%fの%とfの間に、「. 桁数」と記述する。
- 改行したい場合は、改行したい場所に「\n」を記述する。
- データの区切りにタブ (Tab) が使われることが多い。タブを入れたいときには、「\t」を記述する。タブとは適当な空白のこと。

## 3.5 制御文

- 通常、プログラムは上から下へと実行される。しかし、条件に従い実行の流れを変えたい場合がある。そのような場合、制御文をつかう。制御文には、分岐と繰り返しがある。
- 分岐には if 文と switch 文がある。ただし、switch 文は滅多に使われない。

1. if(条件 1){ 文 1}else if(条件 2){ 文 2}else{ 文 3}

- 条件 1 が真の時、文 1 が実行されます。条件 1 が偽の場合、次の条件 2 の真偽を判断し、真ならば文 2 を実行します。else if 文はいくらでも書くことができます。
- 最初に真である条件に続く文を実行すると、if 文から抜けます。
- 全ての if 又は else if の条件が偽ならば、else の文 3 を実行します。

2. switch(式)

- 余り使われないので、説明は省く。

- 使用頻度の高い繰り返し文は、次の 3 個である。

---

<sup>3</sup>データを書き込むアドレスを指定している

1. for(初期値; 継続条件式; 再設定式){ 文 }

– 実行順序は、以下の通り。

- (1) 初期値の設定
- (2) 継続条件が真ならば、続く文を実行し、偽ならば for 文は終了する。
- (3) 再設定式を実行
- (4) 再び、(2) から実行する。

リスト 1: for 文による 1~100 までの和の計算

```
1 #include <stdio.h>
2 int main(void){
3     int a, b;
4
5     a = 0;
6     b = 0;
7
8     for(a=1; a<=100; a++){
9         b += a;
10    }
11
12    printf("b = %d\n",b);
13
14    return 0;
15 }
```

2. do{ 文 }while(継続条件式);

– 実行順序は、以下の通り。

- (1) 文を実行
- (2) 継続条件式が真ならば (1) へ戻り、偽ならば do 文は終了

リスト 2: do-while 文による 1~100 までの和の計算

```
1 #include <stdio.h>
2 int main(void){
3     int a, b;
4
5     a = 1;
6     b = 0;
7
8     do{
9         b += a;
10        a++;
11    }while(a<=100);
12
13    printf("b = %d\n",b);
14
15    return 0;
16 }
```

3. while(継続条件式){ 文 };

– 実行順序は、以下の通り。

- (1) 継続条件式が偽ならば、while 文は終了。新ならば、(2) へ
- (2) 文を実行
- (3) (1) へ戻る

リスト 3: while 文による 1~100 までの和の計算

```
1 #include <stdio.h>
2 int main(void){
3     int a, b;
4
5     a = 1;
6     b = 0;
7
8     while(a<=100){
9         b += a;
10        a++;
11    }
12
13    printf("b = %d\n",b);
14
15    return 0;
16 }
```

### 3.6 配列

- 配列は、同じようなデータが多くある場合に使います。多くのデータに一つずつ名前をつけると大変です。1万個のデータがあった場合、1万個の名前を付けた変数を用意しますか?。下の例で、変数を用いての大量のデータ処理が不可能ということが分かるでしょう。

– 1万個の変数で領域を用意する場合の宣言

```
double aaa, aab, aac, aad, aae, aaf;
      ⋮
double oun, ouo, oup, ouq;
```

– 配列で1万個の領域を用意する場合の宣言

```
double a[10000]
```

- 配列を使う場合も宣言が必要です。宣言の例は、以下の通りです。

配列の次元	要素数	宣言例
1次元	100	double x[100]
2次元	100×100	double x[100][100]
3次元	100×100×100	double x[100][100][100]



- 配列添字は 0 から始まります。したがって、「double x[1000]」と宣言した場合、使える配列は、x[0] ~ x[999] までです。
- 添字である数字でデータの指定ができるため、メモリからのデータの読み書きが単純化できます。

### 3.7 ポインター

この辺はテストにでないが、以下について理解して欲しい。

- 通常の変数には整数や実数の値を格納するが、ポインターにはアドレスを格納する。
- ポインターの宣言には、型名とアスタリスク (\*) を付ける。

```
int *pi;
double *px;
```

- 変数のアドレスを取り出すには、変数名の前にアンパサンド (&) をつける。&はアドレス演算子である。

```
pi=&i;
px=&x;
```

- ポインターが示しているデータの値を取り出すためには、ポインター変数の前にアスタリスク (\*) をつける。\*は間接参照演算子である。

```
j=*pi;
y=*px;
```

### 3.8 関数

- C 言語は、関数の集まりです。その中で main 関数は特別で、そこから実行されます。
- プログラマーが関数を作成する場合、以下のように記述します。ここでは、hogehoge がプログラマーが作成した関数です。

リスト 4: 関数の書き方

```
1 #include <stdio.h>
2
3 戻り値の型  hogehoge(引数の型と名前);
4
5 /*----- main function -----*/
6 int main(void){
7     文;
8     a = hogehoge(実引数)
9     文;
10    return 0;
11 }
```

```

12 |
13 | /* ----- user defined function ----- */
14 | 戻り値の型 hogehoge(仮引数の型と名前){
15 |     文;
16 |     return(式);
17 | }

```

- 1行目が関数のプロトタイプ宣言です。関数名、戻り値や引数の型を宣言しています。
- 8行目で関数 hogehoge を呼び出しています。
- 関数 hogehoge の定義は、14~17行で行っています。
- 関数の戻り値は、16行目の式の値です。

- 関数へのデータの渡し方に、2種類あります。

値渡し            呼び出す側と呼ばれる側の関数が各々変数を用意します。仮引数は、実引数のコピーとなります。呼ばれた関数が処理をしても、呼び出した側の実引数の変数は、影響がありません。

アドレス渡し    呼び出す側の実引数は、アドレスです。呼ばれる側は、ポインターを用意して、実引数のアドレスを受け取ります。呼ばれた関数が処理をすると、呼び出した側の実引数の変数にも影響があります。

### 3.9 プリプロセッサ

- コンパイルに先立って、ソースプログラムを整形する機能をプリプロセッサと言う。プリプロセッサは、必ず#から始まる。
- #include 文は、引き続くファイルがそこに展開される。
- #define 文は、文字の置き換えに使われる。