物理学情報処理演習

8. 計算処理

Cでプログラム

UNIXで処理

可視化

大久保晋

E-mail: buturi-johoshori@tiger.kobe-u.ac.jp

http://extreme.phys.sci.kobe-u.ac.jp/staffs/okubo/lectures/Programming/index.html

参考文献

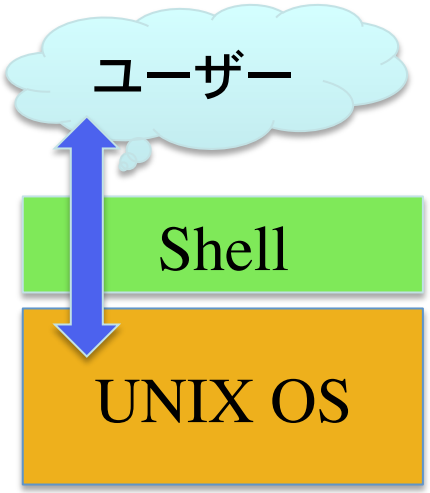
“実習 UNIXシェル”　プレンティスホール出版　エリー・クイグリー著

ISBN 4-89471-031-5

Unix shell その1

シェル(Shell）はUNIXオペレーティングシステム(OS)とユーザーとの間のインターフェースをとりもつ特別なプログラムである。

UNIX OSでは様々なプログラムが動作しているが、その根幹をなすプログラムをカーネル(kernel)と呼ぶ。



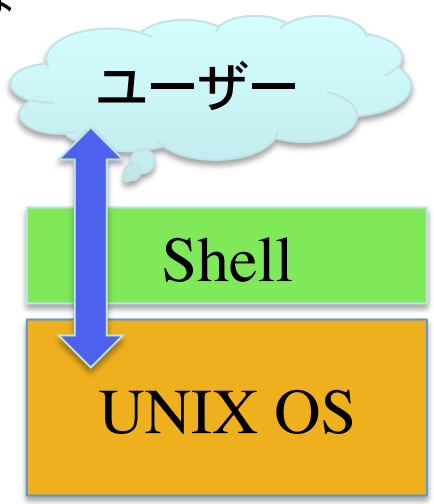
kernelでは、プロセス(プログラム)の生成と制御、メモリ、ファイルシステム、通信などを管理する。

ユーザーは直接kernelを操作することは出来ないので、仲介となるshellプログラムが必要である。些細なプロセスの呼び出しにもshellを通じて行う。

Unix shell その2

ユーザーにとって、ShellはほとんどUNIX OSそのものに見えるであろう。実際に、ユーザーがUNIXにログインすると、対話型Shellが立ち上がりユーザーからの命令を待ち受け、Shellは

(a) コマンド行を解釈
(b) ワイルドカード、リダイレクション、
　　パイプ、ジョブ制御を操作
(c)コマンドを探索し、コマンドを起動



を行う。

このような作業で、定型作業を行う場合、
Shellにはスクリプトと呼ばれる、作業手順を示したプログラムを作ることができる。
以下では、リダイレクションを使った入出力の代替を学ぶ。

ユーザーインターフェースのプログラムなので、Shellには複数の種類があり、それぞれに特徴がある。

Unix shell その3

Shallには大別してbourneシェルとCシェルというインターフェースの異なるシェルがある。

MacOSX Leopard で使われているのは bourneシェル系のbashと呼ばれるものである。

bourneシェル：
bsh, bash, ksh, zsh, etc.... コマンドプロンプトが %

Cシェル：
csh, tcsh, etc .... コマンドプロンプトが $

Unix shell その4

Shellが提供している機能には以下のようなものがある

`$ プログラム　> [outputファイル名]`

`>`　はプログラムによって画面に出力される内容を`[outputファイル名]`で示されたテキストファイルに出力する。

`$ プログラム　< [inputファイル名]`

`<`　は`[inputファイル名]`で示されたテキストファイルの内容を、あたかもキーボードから入力したかのように、プログラムに渡す。

`bash`や`tcsh`では、

- 入力した行をその行に限って矢印キーを使って編集可能
- 上下キーで過去に入力したコマンドを呼び出せる
- `tab`キーで、ファイル名ディレクトリなどの補完をする

演習８–1：Cでプログラム, unixで出力, 可視化

- C言語で0から4πまでのθ, sin(θ)の２列のデータを生成し、表計算ソフトでプロットしよう

以下は、0から4πまで、0.01π刻みでsin(θ)を計算し、タブ区切りで θ　sin(θ)　を標準出力するプログラム、これを8-1.cとして入力しよう

8-1.c ソースプログラム

```
#include <stdio.h>
#include <math.h>

#define PI 3.141592653589793
#define MAX_STEP 100
#define MIN_THETA 0
#define MAX_THETA 4
```

つづく

演習８－１：Cでプログラム, unixで出力, 可視化

つづき

```
int main(void)
{
  int i, j;
  double step, theta;

  step = 1.0/MAX_STEP;
  for (i=MIN_THETA; i<MAX_THETA*MAX_STEP; i++)
  {
   theta = PI * i * step;
   printf("%lf¥t%lf¥n", theta, sin(theta));
  }

  return (0);
}
```

演習８−１：Cでプログラム, unixで出力, 可視化

コンパイル＆出力

```
gcc -lm -o 8-1 8-1.c
```
コンパイル

```
./8-1 > data.txt
```
unixのshellコマンド、リダイレクト『 > 』を使って、標準出力に出力に出される文字をディスク上にテキストファイルとして保存する

保存先ファイル名は data.txt

OpenOffice 表計算ソフトで「開く」を指定する。

新しいwindowが開く、ファイルの種類として

「テキスト CSV(*.csv; *.txt, *.xls」を指定すること。

θ, sin(θ)の２列のデータが読めて、横軸 θ として、縦軸 sin(θ) として、グラフを書いてみよう。原点を通る sin(θ) 関数になっているか確認せよ。

演習８−２：Ｃでプログラム, unixで入力

- １０人分のテストの成績を読み込んで、その平均と標準偏差を求めるプログラムを作ろう

演習7-1で行ったプログラムを使い（そのまま！）、10人分のデータをエディターで作成し、読み込ませて答えを出そう！

8-2.cソースコード これを入力

```
#include <stdio.h>
#define NUMBER 10

int main(void){
    int i;
    int seiseki[NUMBER];
    double sum=0.0, ave = 0.0, sigma2 = 0.0, sigma = 0.0;

    for (i=0; i<NUMBER; i++){
        printf("%d-th point ?",i);
        scanf("%d", &(seiseki[i]) );
    }
```

演習８−２：Cでプログラム, unixで入力

続き

```
for (i=0; i<NUMBER; i++)
   {
    sum = sum + seiseki[i];
   }
   ave = sum/NUMBER;

   for (i=0; i< NUMBER; i++)
   {
    sigma2 += (seiseki[i] - ave)*(seiseki[i] - ave)/NUMBER;
   }
   sigma = sqrt(sigma2);

   printf(“¥n”);
   printf("average= %lf¥n",ave);
   printf("standard deviation= %lf¥n",sigma);

   return (0);
}
```

演習８−２：Cでプログラム, unixで入力

・ mi エディターを開き、以下のデータを入力し”seiseki1.txt”, “seiseki2.txt”として保存しよう

seiseki1.txt

```
0
1
2
3
4
5
4
3
2
1
```

seiseki2.txt

```
5
5
5
5
5
5
5
5
4
6
```

数字いれて、すぐに改行をいれること。最後の行は改行を入れた後、余計な改行は入れない

演習８−２：Cでプログラム, unixで入力

- 入力した 8-2.c をコンパイルする

  gcc -lm -o 8-2 8-2.c

- UNIXのshellの機能、リダイレクト「＜」を使って、標準入力から “seiseki1.txt”, “seiseki2.txt” を読み込ませよう

  ./8-2 < seiseki1.txt

  以下の様に出力されているだろうか？

```
0-th point ?1-th point ?2-th point ?3-th point ?4-th point ?5-th
point ?6-th point ?7-th point ?8-th point ?9-th point ?
average= 2.500000
standard deviation= 1.500000
```

演習８−２：Cでプログラム, unixで入力

- “seiseki2.txt” を読み込ませた結果はどうだろうか？

  ./8-2 < seiseki2.txt

  以下の様に出力されているだろうか？

```
0-th point ?1-th point ?2-th point ?3-th point ?4-th point ?5-th
point ?6-th point ?7-th point ?8-th point ?9-th point ?
average= 5.000000
standard deviation= 0.447214
```

- これで、エディターで成績を入力できるので、入力ミスがあってもやり直しが利く！

演習８−３：行列計算その２

・ 演習7−２の行列計算で行列A, 行列Bをファイルから読ませるプログラムに変えよう！ 行列A,Bを与える matrixAB.txtは、

```
1 0 1
0 0 0
1 0 -1
0 1 -1
1 0 1
-1 1 0
```

の形であたえる

Hint!:

標準入力から複数の整数を読み込ませるのは

scanf(“%d %d %d”, &(a[0]) , &(a[1]) , &(a[2]));

等とする。

・ 整数行列入力の見本プログラム

8-3test.cソースコード

```
#include <stdio.h>
#include <math.h>
#define SIZE 3
```

演習８−３：行列計算その２

つづき

```
int main(void)
{
   int a[SIZE][SIZE];
   int i, j;

   for (i=0; i<SIZE; i++)
   {
    printf("%d-th columns of matrix A", i);
    scanf("%d %d %d", &(a[i][0]), &(a[i][1]), &(a[i][2]));
   }

   printf("¥n matrix A = ¥n");
   for (i=0; i<SIZE; i++){
    for (j=0; j<SIZE; j++){
     printf("%d¥t", a[i][j]);
    }
    printf("¥n");
   }
   return (0);
}
```

演習８−３：行列計算その２

行列A として以下のファイル(matrixA.txt)を作成し

```
1  0  1
0  0  0
1  0  -1
```

```
gcc -lm -o 8-3test 8-3test.c
```

```
./8-3test < matrixA.txt
```

として実行してみよう

実行結果

```
0-th columns of matrix A1-th columns of matrix A2-th
   columns of matrix A
matrix A =
1        0        1
0        0        0
1        0        -1
```

となっているだろうか？

演習８−３：行列計算その２

与えるMatrixAB.txtを以下とすると

```
1 0 1
0 0 0
1 0 -1
```
行列A

```
0 1 -1
1 0 1
-1 1 0
```
行列B

実行結果は

```
matrix A+B =
1       1       0
1       0       1
0       1       -1
matrix A-B =
1       -1      2
-1      0       -1
2       -1      -1
```

となっているだろうか？