Shin Yoshizawa: shin@riken.jp

画像情報処理論及び演習I

第1回講義・演習の復習＆補足

www.riken.jp/brict/Yoshizawa/Lectures/Lec01.pdf

www.riken.jp/brict/Yoshizawa/Lectures/ex01.pdf

www.riken.jp/brict/Yoshizawa/Lectures/Ex01.zip

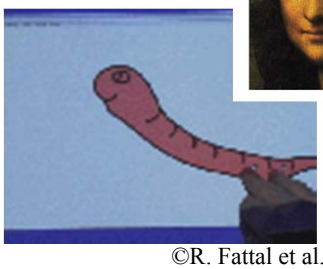
www.riken.jp/brict/Yoshizawa/Lectures/ex01_2.pdf

吉澤　信

Shin Yoshizawa: shin@riken.jp

復習：第一回講義まとめ

- ✓ **画像処理は信号(音声)処理・CG (Computer Graphics) /CV(Computer Vision)/パターン認識の分野と密接な関連がある:**
  - 情報学ではCG と並んで花形の分野.
  - 目に見える結果、綺麗、技術的面白さ.



©T. Igarashi et al.

©R. Fattal et al.

©V. Blanz et al.

Shin Yoshizawa: shin@riken.jp

復習：第一回講義まとめ

- ✓ **様々な応用分野がある（データが画像）:**
  - デジタルカメラの爆発的普及により…
  - エンターテイメント産業：映画・ゲーム等.
  - 自然科学：天文学・生物学・化学・物理学等の観察・観測データ解析等.
  - 工業・工学：現実世界の製品データ解析等.
  - 医療：CT、MRI等の画像診断等.

© New Line Productions, Inc.

©journal.mycom.co.jp

©heritage.stsci.edu

Shin Yoshizawa: shin@riken.jp

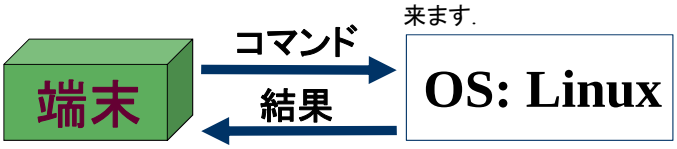
なんでLinuxなんかでやるの？

- ✓ **Windowsでいいじゃん、Visual Studio（VC++）とかのビルダーでいいじゃん！**
  - 端末&エディターを使ってのプログラミングはどんなコンピュータの環境でも使える基本！
- **例えば，…**
  - 1私企業のマイクロソフト依存は危険！マイクロソフトが潰れたら？主流じゃなくなったら？
  - Visual Studioって結構高いよ(10万～200万).
  - スマートフォン等の次世代携帯機器はAndroid OSやMac OS（共にUNIX/Linuxベース）が主流.
  - 画像処理アルゴリズムやC/C++言語とは関係が無いビルダー固有の開発方法を覚えなければいけない.
  - 就活等で「Linuxでのプログラミングも出来ます！」.

Shin Yoshizawa: shin@riken.jp

コンソール(端末) とエディター(emacs)

- ✓ コンソール(端末):cd, mv, pwd等のLinuxコマンドを入力し「Enter」キーを押す事でOS(オペレーティングシステム)にファイル操作やプログラムのコンパイル等の命令を与える.
  - ✓ 「Tab」キーでパスやコマンドを補間出来ます.

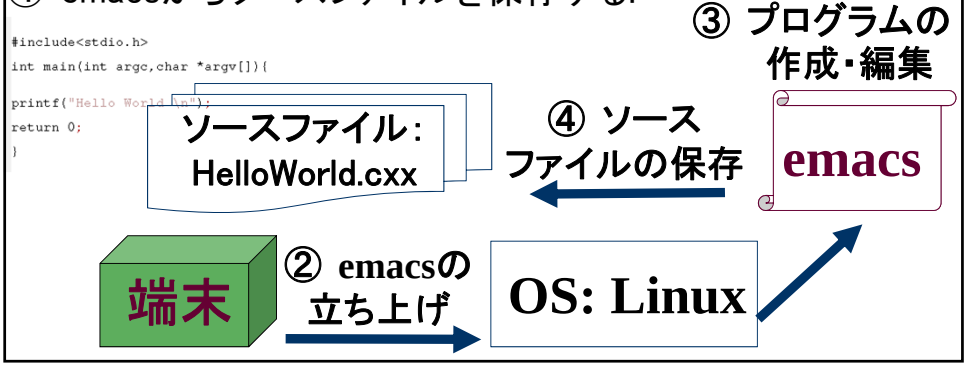


- ✓ エディター(emacs): プログラムや文章を書くツール. テキストでプログラムを入力→ファイルとして保存.

Shin Yoshizawa: shin@riken.jp

Linuxでのプログラミングの流れ:1

① コンソール(端末)を立ち上げる:画面左下のコンソールアイコンをクリック.
② 端末:でemacsを立ち上げる: 端末に「emacs &」と打ち込み「Enter」キーを押す.
③ emacsでプログラム(ソースファイル)を書く.
④ emacsからソースファイルを保存する.

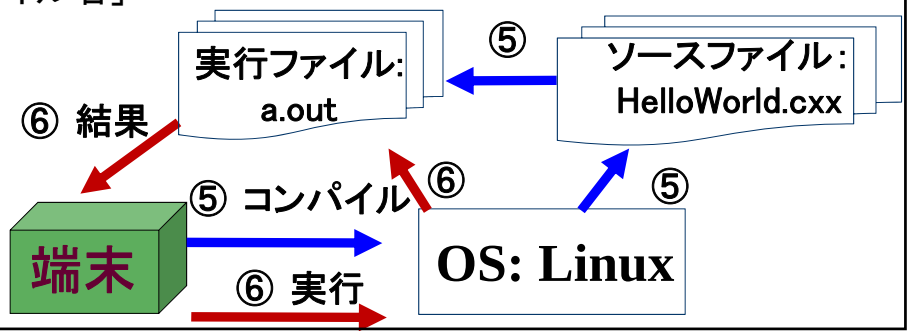


Shin Yoshizawa: shin@riken.jp

Linuxでのプログラミングの流れ：2

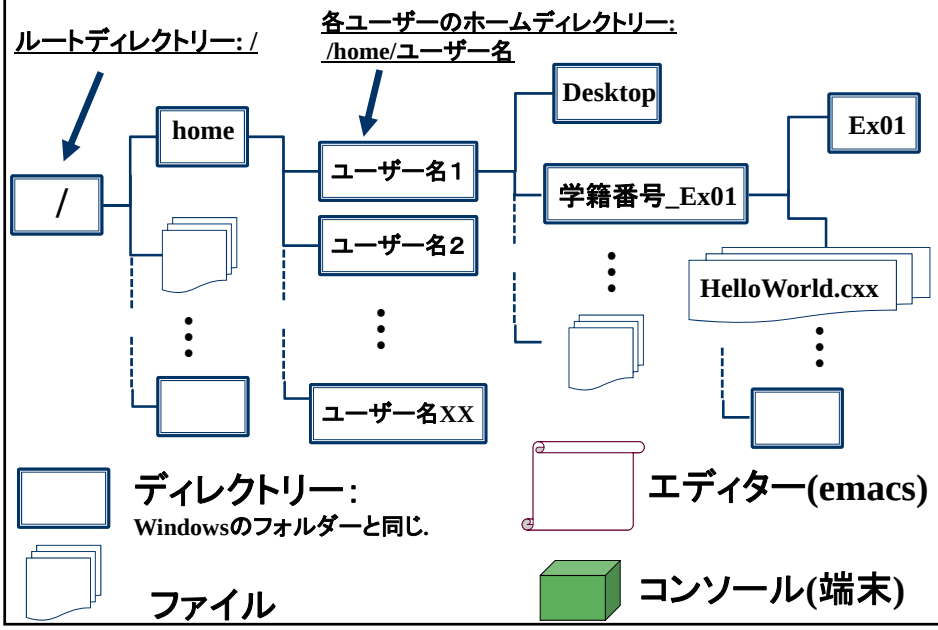
⑤ 端末でソースファイルをコンパイルして実行ファイルを作る:「g++ ソースファイル名」or
「g++ −o 実行ファイル名 ソースファイル名 」.
実行ファイル名を指定しない場合は「a.out」という名前の実行ファイルが作成される.

⑥ 端末で実行ファイルを実行する:「./a,out」or「./実行ファイル名」



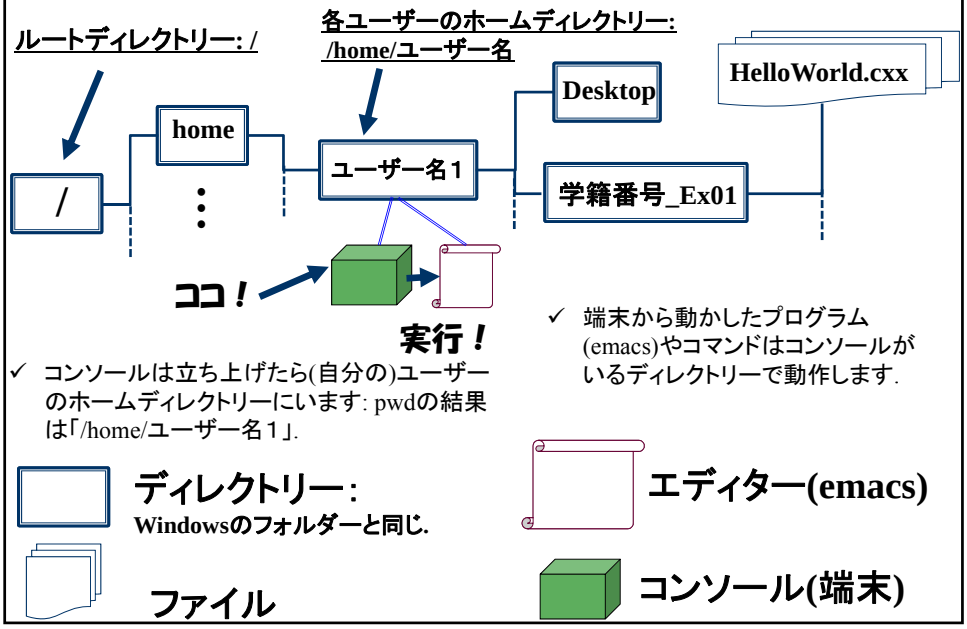
Shin Yoshizawa: shin@riken.jp

ディレクトリー構造：絶対パス・相対パス



Shin Yoshizawa: shin@riken.jp

ディレクトリー構造



- ✓ コンソールは立ち上げたら(自分の)ユーザーのホームディレクトリーにいます: pwdの結果は「/home/ユーザー名1」.
- ✓ 端末から動かしたプログラム(emacs)やコマンドはコンソールがいるディレクトリーで動作します.

ディレクトリー:
Windowsのフォルダーと同じ.

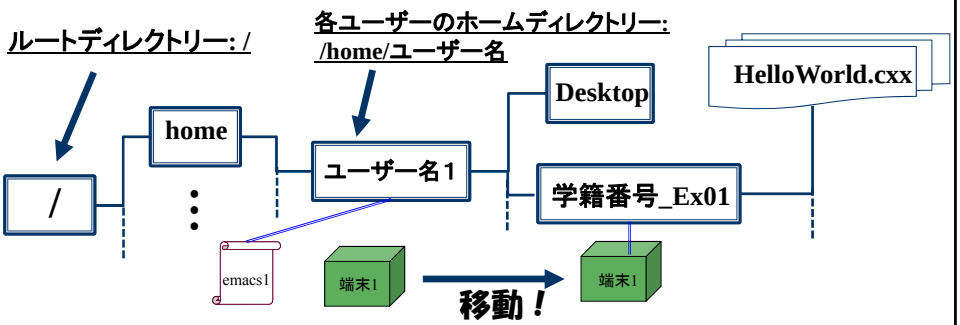
ファイル

エディター(emacs)

コンソール(端末)

Shin Yoshizawa: shin@riken.jp

ディレクトリー構造



- ✓ cdで今のコンソールの位置を移動します.
- ✓ 最初に立ち上げたemacsの位置はそのままです.
- ✓ 「cd ディレクトリー名」で移動します.
- ✓ : pwdの結果は「/home/ユーザー名1/学籍番号_Ex01」

ディレクトリー:
Windowsのフォルダーと同じ.

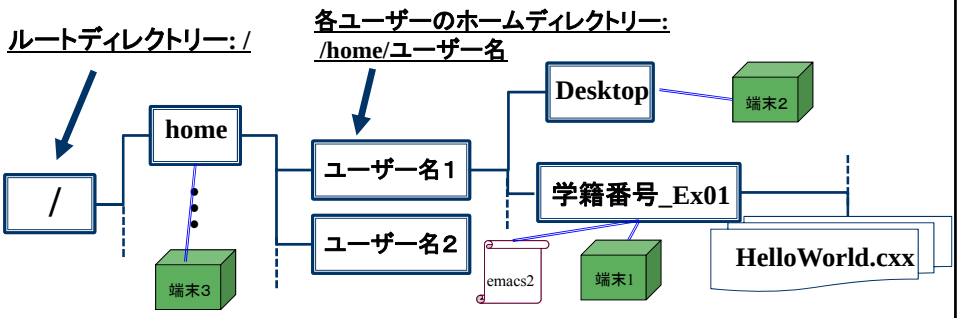
ファイル

エディター(emacs)

コンソール(端末)

Shin Yoshizawa: shin@riken.jp

ディレクトリー構造：絶対パス・相対パス



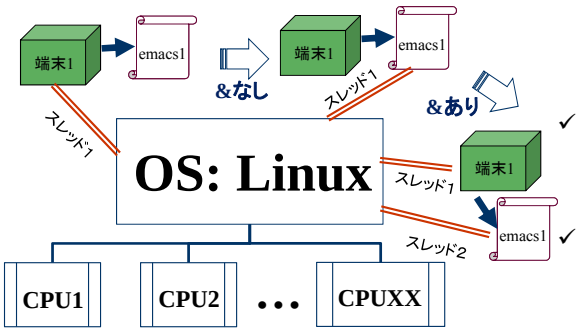
- ✓ 絶対パス：ルートディレクトリーから全てのディレクトリーを含んだパス.
- ✓ 相対パス：端末の自分の位置からの相対的パス.
- ✓ cd、g++、emacsの後に絶対パスを入れる事(引数として実行)で端末の今の位置(pwd)とは関係なしでコマンドを実行したりファイルを開けたりします.
- ✓ 「./」は今の、「../」は一つ上. 「cd /home/ユーザー名1/学籍番号_Ex01」
  「emacs /home/ユーザー名1/学籍番号_Ex01/HelloWorld.cxx」
- ✓ 相対パスは端末やemacs(動かしているプログラム)の今の位置からパスを指定します.

端末1：「cd ../../ユーザー名2」, 端末2：「cd ../学籍番号_Ex01」, 端末3「cd ./ユーザー名1」.
端末1：「emacs ./HelloWorld.cxx」、端末2：「emacs ../学籍番号_Ex01/HelloWorld.cxx」.

Shin Yoshizawa: shin@riken.jp

マルチタスク vs シングルタスク

- ✓ プログラムは(古典的には)一つのCPU (Central Processing Unit:中央演算子)に一つしか動かせない！
- ✓ マルチタスク：OSの機能としてCPUの数よりも多くのプログラムをプロセス(スレッド)として動かす事.
- ✓ マルチスレッド:プログラミングとして複数のプロセスを管理して動かす事.



- ✓ 「emacs」と「&」を付けないで端末から実行した場合は端末に割り当てられていたスレッドがemacsに渡される→emacsからスレッドが戻ってこないと端末は動かない.
- ✓ 「emacs &」と実行する事で端末に割り当てられていたスレッドとは別のスレッドが割り当てられて端末もemacsも両方とも使える.
- ✓ 「&」なしで動かしていたプログラムは端末にて「Control-Z」の後に「bg」コマンドでそのプログラムをバックグランド化しできる.

Shin Yoshizawa: shin@riken.jp

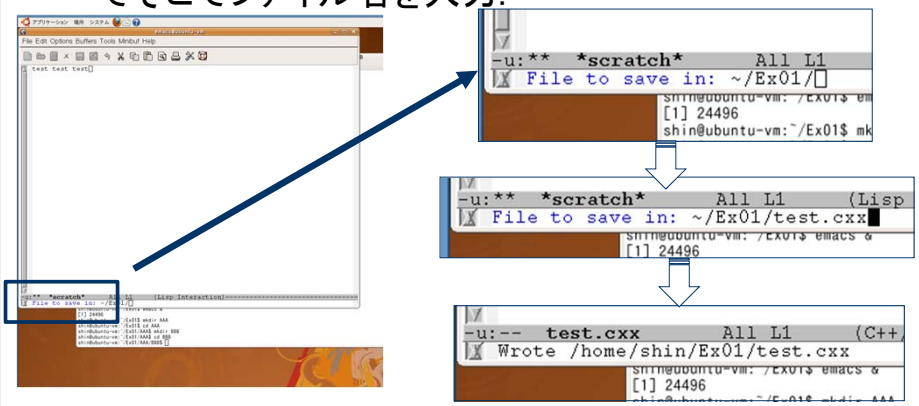
前回質問があったところの補足

- ✓ Ex01.pdfで「¥」はバックスラッシュ「 \ 」でキーボード右上の「¥」キーに対応しています. 以後スライドに¥が出てきたら「 \ 」に読み替えてください.
- ✓ Ex01.pdfでの「Control-X」や「Control-S」の「Control」はキーボード左下の「Ctrl」キーを押しながらの意味です。
- ✓ Ex01.pdfの「int main(int argc,char *argv[])」の「[]」はEnterキーの横にある括弧「[ ]」です。
- ✓ emacsでファイルの保存や入力のときにおかしくなったら「Control-G」を押してみてください.

Shin Yoshizawa: shin@riken.jp

前回質問があったところの補足

- ✓ emacsについて:
  - ファイルの入出力:「Control-X」、「Control-S」又は、左上のFile(ファイル)→save buffer as(名前を付けて保存)でemacsの下にファイル名を入力する所が現れるのでそこでファイル名を入力.



Shin Yoshizawa: shin@riken.jp

前回質問があったところの補足

- ✓ emacsについて:
  - 日本語（全角）入力・英語(半角英数)入力の切り替え「Control−¥」.

✓ プログラムはいつも半角英数で入力してください.

  - 下の部分でセーブモード等のコマンド入力状態からの脱出は「Control－G」.
  - プログラムは「Tab」キーを押すと自動的に構造化して見やすくなります.
  - Control−Kで一行削除.
  - Undoは「Control−/」

Tabなし

```
/* Only Copy */
for(i=0;i<in->sy;i++)
for(j=0;j<in->sx;j++){
out->img[i][j] = in->img[i][j];
}
```

Tabあり

```
/* Only Copy */
for(i=0;i<in->sy;i++)
  for(j=0;j<in->sx;j++){
    out->img[i][j] = in->img[i][j];
  }
```

Shin Yoshizawa: shin@riken.jp

C: Hello Worldの説明

stdio.hはprintf等の標準入出力関数群

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

ヘッダーファイル(.h)の読み込み

mainはLinuxでは必ずint型の関数.

```
int main(int argc,char *argv[]){
```

argcにはこのプログラムが実行されたときの引数の数が入る.
*argv[]には引数の文字列が入る.

```
printf("Hello World \n");
```

printfは端末(標準入出力)に文字や数値を出力する関数,「¥n」は改行.

```
return 0;
```

プログラムの正常終了を表す0をOSに返す.

```
}
```

Shin Yoshizawa: shin@riken.jp

C: argvの説明

端末での実行結果

```
shin@ubuntu-vm:~$ ./a.out
argc = 1
argv[0] = ./a.out
shin@ubuntu-vm:~$ ./a.out abc
argc = 2
argv[0] = ./a.out
argv[1] = abc
shin@ubuntu-vm:~$ ./a.out abc efg
argc = 3
argv[0] = ./a.out
argv[1] = abc
argv[2] = efg
shin@ubuntu-vm:~$ ./a.out abc efg 1.234
argc = 4
argv[0] = ./a.out
argv[1] = abc
argv[2] = efg
argv[3] = 1.234
shin@ubuntu-vm:~$ ./a.out abc efg 1.234 HIJK
argc = 5
argv[0] = ./a.out
argv[1] = abc
argv[2] = efg
argv[3] = 1.234
argv[4] = HIJK
shin@ubuntu-vm:~$
```

ソースファイル

```c
#include<stdio.h>

int main(int argc,char *argv[]){


printf("argc = %d\n",argc);
 int i;
 for(i=0;i<argc;i++)
   printf("argv[%d] = %s\n",i,argv[i]);


return 0;

}
```

- ✓ %dはint、%sはchar []やchar *の文字列を表示するときに使う.
- ✓ floatは%f, doubleは%lf、charは%c.
- ✓ argcにはこのプログラムが実行されたときの引数の数が入る.
- ✓ *argv[]には引数の文字列が入る.
- ✓ プログラム内ではargv[1],argv[2]とかで使う.argv[0]には実行ファイル名が入る.
- ✓ 文字列な事に注意！数値として使いたい場合は、stdlib.hをインクルードしてatoi()やatof()を使う.

Shin Yoshizawa: shin@riken.jp

C: argvの説明

端末での実行結果

```
shin@ubuntu-vm:~$ ./a.out abc
argv[1] = abc
atoi(argv[1]) = 0
atof(argv[1]) = 0.000000
shin@ubuntu-vm:~$ ./a.out 1
argv[1] = 1
atoi(argv[1]) = 1
atof(argv[1]) = 1.000000
shin@ubuntu-vm:~$ ./a.out 2
argv[1] = 2
atoi(argv[1]) = 2
atof(argv[1]) = 2.000000
shin@ubuntu-vm:~$ ./a.out 1.2
argv[1] = 1.2
atoi(argv[1]) = 1
atof(argv[1]) = 1.200000
shin@ubuntu-vm:~$ ./a.out 0.567
argv[1] = 0.567
atoi(argv[1]) = 0
atof(argv[1]) = 0.567000
shin@ubuntu-vm:~$ ./a.out 123.456
argv[1] = 123.456
atoi(argv[1]) = 123
atof(argv[1]) = 123.456000
shin@ubuntu-vm:~$
```

ソースファイル

```c
#include<stdio.h>
#include<stdlib.h>

int main(int argc,char *argv[]){


  printf("argv[1] = %s\n",argv[1]);

  int a = atoi(argv[1]);
  printf("atoi(argv[1]) = %d\n",a);

  double b = atof(argv[1]);

  printf("atof(argv[1]) = %lf\n",b);

  return 0;
}
```

- ✓ 文字列な事に注意！数値として使いたい場合は、stdlib.hをインクルードしてatoi()やatof()を使う.

演習

1. C言語でpnm画像の入出力を書いてみよう:
   10. 端末でfirefoxを立ち上げて
   www.riken.jp/brict/Yoshizawa/Lectures/Ex01.zip
   を開いて、学籍番号_Ex01のディレクトリーにダウンロードしてください.
   Firefox:編集→設定→一般→ダウンロード→「ファイルごとに保存先を指定する」にチェックを入れてください.
   11. 端末で「unzip Ex01.zip」として展開後に「cd Ex01」、「emacs ex01.cxx &」でプログラムを開いてください. ex01.cxxはpgm画像を読み込んでそのままセーブするプログラムです.

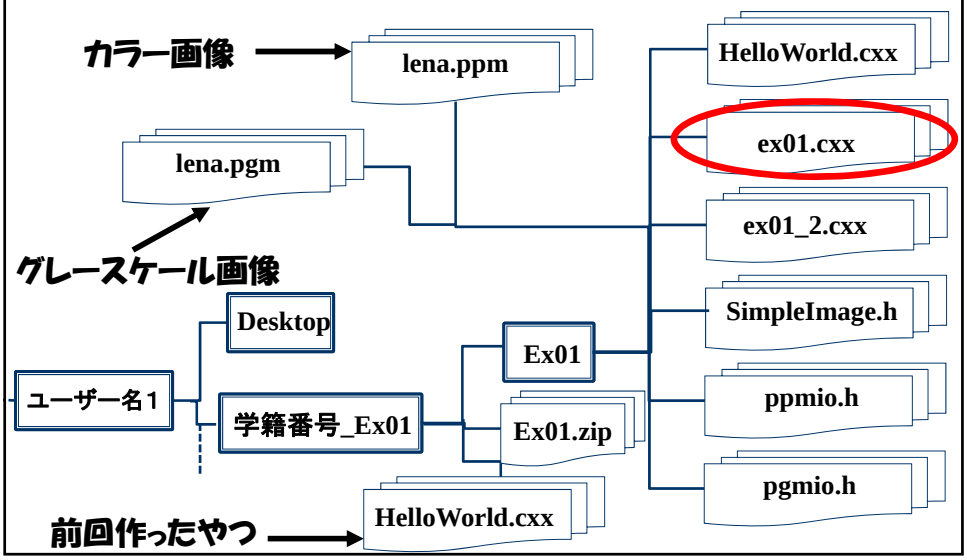
第一回演習:C: Hello World & pnm画像 & 閾値の続き↑からはじめましょう!

11**までいったら、まずは、**ex01.cxx**の解説から**…

Shin Yoshizawa: shin@riken.jp

演習

ex01.cxxはpgm画像を読み込んでそのままセーブするプログラムです.



Shin Yoshizawa: shin@riken.jp

演習：pgm入出力

ex01.cxxはpgm画像を読み込んでそのままセーブするプログラムです.

```
#include<stdio.h>
#include<math.h>
#include"SimpleImage.h"
#include"pgmio.h"

int main(int argc,char *argv[]){
  if(argc!=3){
    printf("Please use as ./a.out input.pgm output.pgm\n");
    return 0;
  }
  Image *in = new Image();

  getPGM(in,argv[1]);

  Image *out = new Image(in->sx,in->sy);

  /* main processing */
  int i,j;

  /* Only Copy */
   for(i=0;i<in->sy;i++)
    for(j=0;j<in->sx;j++){
      out->img[i][j] = in->img[i][j];
    }
   /* End Copy */

  /*
  double threshold = 128.0;
  for(i=0;i<in->sy;i++)
    for(j=0;j<in->sx;j++){
      if(in->img[i][j]>=threshold){
        out->img[i][j] = in->img[i][j];
      }else{
        out->img[i][j] = 0.0;
      }
    }

    */

  /* end main processing */

   savePGM(out,argv[2]);

  delete out;
  delete in;

  return 0;
}
```

```
#include<stdio.h>
#include<math.h>
#include"SimpleImage.h"
#include"pgmio.h"
```

入力用Imageクラスinの宣言・new.

argv[1]で渡されたファイル名のpgm画像を開いてImageクラスinに入れる.

出力用Imageクラスoutの宣言・new.

Inからoutへ画素の値をコピー.

SimpleImage.h

- ✓ SimpleImage.h: 2次元配列で一色(グレースケール)の画像を表すImageクラス

argv[2]で渡されたファイル名にoutの中身をpgm画像として保存.

pgmio.h

- ✓ pgmio.h: pgmファイルの入出力を行う2つの関数.

In、out領域の開放(delete).

Shin Yoshizawa: shin@riken.jp

演習：Imageクラス

SimpleImage.h: 2次元配列で一色の画像を表すImageクラス.

#include"SimpleImage.h"した後の使い方例：

宣言・メモリ確保 (allocation):

```
Image *in = new Image();
Image *out = new Image(in->sx,in->sy);
```

```
//2D Image class
class Image{
 public:
  int sx,sy;
  double **img;
  int gray;
  Image (){
    img=NULL;
    sx=sy=0;
    gray = 255;
  }
  Image(int dx,int dy){
    allocate(dx,dy,255);
  }
  void allocate(int dx,int dy,int dgray){
    sx = dx;
    sy = dy;
    int i;
    img = new double* [sy];
    for(i=0;i<sy;i++){
      img[i] = new double[dx];
    }
    gray = dgray;
  }
  virtual ~Image(){
    int i;
    for(i=0;i<sy;i++){
      delete [] img[i];
    }
    delete [] img;
  }
};
```

SimpleImage.h

処理：

```
getPGM(in,argv[1]);
```

```
/* Only Copy */
 for(i=0;i<in->sy;i++)
  for(j=0;j<in->sx;j++){
    out->img[i][j] = in->img[i][j];
  }
```

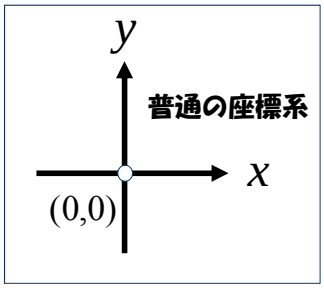
- ✓ 画像サイズ：縦：sy、横sx.
- ✓ (座標(i,j)での)画素値：img[i][j]

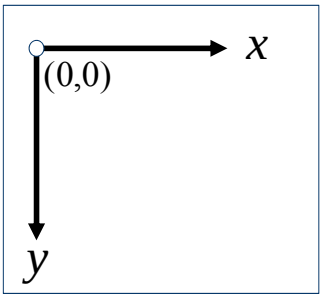
メモリの開放：

```
delete out;
delete in;
```

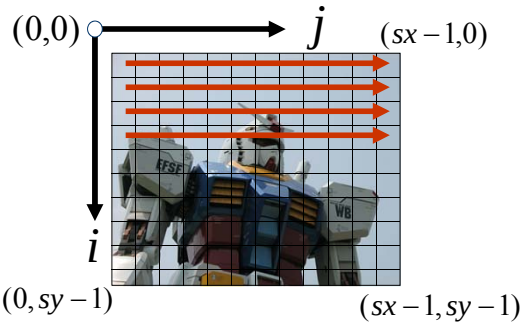
Shin Yoshizawa: shin@riken.jp

復習:デジタル画像の座標と配列





画像処理でよく使う座標系



輝度値の配列表現：

int $I[sy][sx]$;

double $I[sy][sx]$;

```
for(i = 0;i < sy;i++){
  for( j = 0; j < sx; j++){
    I[i][j] = ...
  }
}
```

Shin Yoshizawa: shin@riken.jp

演習:getPGM(), savePGM()

pgmio.h: pgmファイルの入出力を行う2つの関数.

画像入力： `void getPGM(Image *in,char *filename)`

画像出力： `void savePGM(Image *in,char *filename)`

```
void getPGM(Image *in,char *filename){
  if(in==NULL)return;
  FILE *fp = fopen(filename,"r");
  char P2[100];
  fscanf(fp,"%s",P2);
  int sx,sy,gray;
  fscanf(fp,"%d %d",&sx,&sy);
  fscanf(fp,"%d",&gray);
  in->allocate(sx,sy,gray);
  int i,j;
  for(i=0;i<in->sy;i++)
    for(j=0;j<in->sx;j++){

      int val;
      fscanf(fp,"%d ",&val);
      in->img[i][j] = ((double)(val));
    }
  fclose(fp);
}

void savePGM(Image *in,char *filename){
  FILE *fp = fopen(filename,"w");
  fprintf(fp,"P2\n");
  fprintf(fp,"%d %d\n",in->sx,in->sy);
  fprintf(fp,"%d\n",in->gray);
  int i,j;
  for(i=0;i<in->sy;i++)
    for(j=0;j<in->sx;j++){
      int val = ((int)(in->img[i][j]));
      if(fabs(val-in->img[i][j])>=0.5)val++;
      if(val<0)val=0;
      if(val>in->gray)val=in->gray;

      fprintf(fp,"%d ",val);

    }
  fprintf(fp,"\n");
  fclose(fp);
}
```

pgmio.h

#include"pgmio.h"**した後の使い方例:**

入力：

```
getPGM(in,argv[1]);
```

- ✓ argv[1]で渡されたファイル名のpgm画像を開いてImageクラスinに入れる.
- ✓ **注意:inは下記の様に画像サイズなしでnewされていないといけない！**

```
Image *in = new Image();
```

出力：

```
savePGM(out,argv[2]);
```

- ✓ argv[2]で渡されたファイル名にoutの中身をpgm画像として保存.
- ✓ **注意:outは下記の様に画像サイズありでnewされていないといけない！**

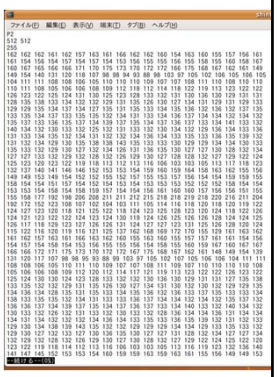
```
Image *out = new Image(in->sx,in->sy);
```

Shin Yoshizawa: shin@riken.jp

演習資料：pnm画像フォーマット

✓ 一番簡単な画像フォーマットです：
- グレースケール画像は「.pgm」、カラー画像は「.ppm」でテキスト形式とバイナリー形式があります.
- グレースケール(.pgm):

1行名: テキストで「P2」
2行目： 画像サイズ(横：width 縦：height)
3行目：画素の階調(最大値) 8bitの場合は255
4行目から: integerで画素値スペース画素値…



- カラー(.ppm):

1行名: テキストで「P3」
2行目： 画像サイズ(横：width 縦：height)
3行目：画素の階調(最大値) 8bitの場合は255
4行目から: integerでR G B R G B R G B…

Shin Yoshizawa: shin@riken.jp

演習資料：pgmを端末上でmoreしてみよう！

pgmの非圧縮形式はテキストファイルなのでmoreで中身が見れます.

1. Ex01.zipを展開したディレクトリーEx01に「cd」を使って端末の位置を動かす「cd /home/ユーザー名/学籍番号_Ex01/Ex01」.

2. 「ls」でディレクトリーの中身を確認(lena.pgmが入っていればOK). 入ってなかったら「pwd」で今の端末の位置を確認して、「cd」と「ls」を使ってEx01の場所を探す. 「cd」だけでエンターキーを押すと自分のホームディレクトリーに戻ります.

3. 端末にて「more lena_s.pgm」でエンターキーを押してみる.

✓ moreはスペースキーで先に進みます. 途中で終了するのはControl-Cです.

Shin Yoshizawa: shin@riken.jp

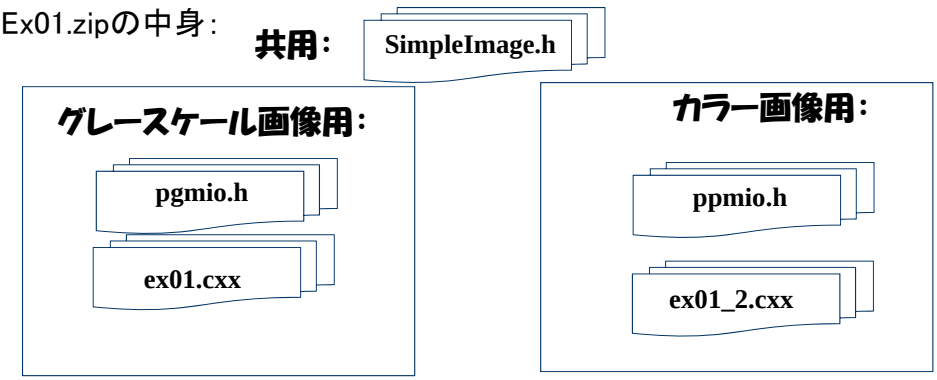
第一回演習：C: Hello World & pnm画像 & 閾値

1. C言語でpnm画像の入出力を書いてみよう：
   10. 端末でfirefoxを立ち上げて
   www.riken.jp/brict/Yoshizawa/Lectures/Ex01.zip
   を開いて、学籍番号_Ex01のディレクトリーにダウンロードしてください.
   11. 端末で「unzip Ex01.zip」として展開後に「cd Ex01」、「emacs ex01.cxx &」でプログラムを開いてください. ex01.cxxはpgm画像を読み込んでそのままセーブするプログラムです。
   12. 端末で「g++ ex01.cxx」として実行ファイルa.outを作成後に「./a.out lena.pgm test.pgm」としてください。その後に「display test.pgm &」と「display lena.pgm &」を実行して同じ画像である事を確認してください.
   13. 同様に「g++ ex01_2.cxx」、「./a.out lena.ppm test.ppm」、「display lena.ppm &」、「display test.ppm &」として同じカラー画像である事を確認してください.

Shin Yoshizawa: shin@riken.jp

演習

Ex01.zipの中身：



- ✓ 今後の全ての演習はこれらのファイル中のプログラム構造を雛形として使っていきますので中身をよく見ておいてください.
- ✓ カラー画像用ではImageクラスをR,G,B3つ使っているだけです.

Shin Yoshizawa: shin@riken.jp

第一回演習：C: Hello World & pnm画像 & 閾値

1. C言語でpnm画像の入出力を書いてみよう:
   14. ex01.cxxで画像をCopyしているところをコメントアウトして、その下の既にコメントアウトしてある部分をコメントアウトを外してください. 再度コンパイル→実行してどんな画像が生成されたか確認してみてください. そのときに「./a.out lena.pgm test1.pgm」と名前を変えてください。
   15. Threshould=128.0となっているところを32.0、64.0、160.0、192.0と変えた場合にどんな画像が生成されるか確認してみてください.同様に「test2.pgm,test3.pgm,test4.pgm,test5.pgm」違う名前でセーブしてください.
   16. 同様にex01_2.cxxの方でも閾値を変えて実行してみてください. ファイル名「test1.ppm,･･･test5.ppm」.

   以上で第一回演習は終了です.