

プログラミング基礎 第3回

藤江 真也
2021年5月7日

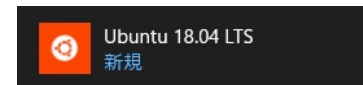
本日の内容

- Linux(Ubuntu)操作の基本
 - ターミナル
 - テキストファイル編集
 - ディレクトリ
- C言語のプログラミング
 - はじめの一步
- drawlibを使ったプログラミングの課題

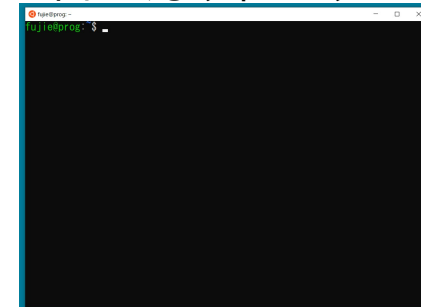
ターミナル

ターミナル

- 「Ubuntu 18.04 LTS」



を起動して出てくるウィンドウ



コマンドプロンプト

- ターミナルにコマンドを打ち込んで色々な処理を行う
- コマンドプロンプト
 - 設定で色々変えられるがデフォルト(初期状態)では以下のような情報が出ているはず



テキストファイル

テキストファイル

- ファイル ... データが記録されるもの
- テキストファイル ... 中身が文字(キャラクタ)だけのもの
- テキストエディタで作成・編集をする
 - 本講義では gedit (ジーエディット)を使う

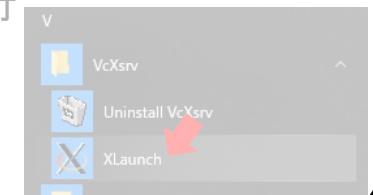


※情報処理第2回で使った「メモ帳」はWindows用のテキストエディタ
Linuxは異なるOSなので、別のテキストエディタを使うことになります

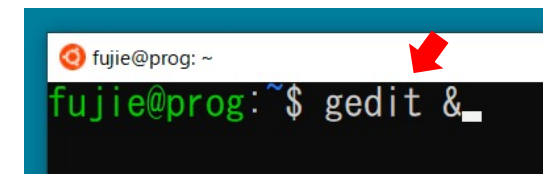
geditの起動

※自動的に実行されるはずなので不要

- VcXsrv (XLaunch) を実行

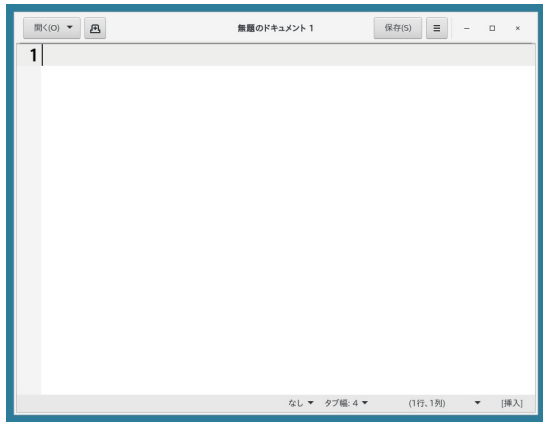


- ターミナルで「gedit &」と入力しエンター



こんな画面が出てくる

- 設定によって多少見た目が異なります
- ターミナルの裏に隠れてる場合もあるので注意



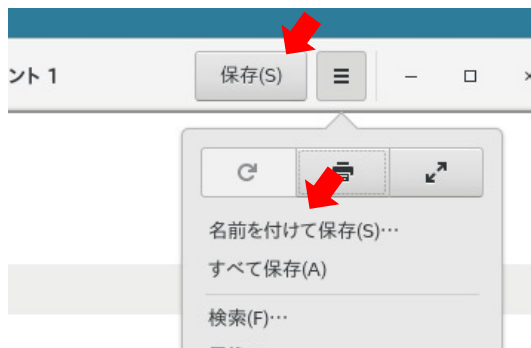
タイピング

- 学生番号, 氏名(漢字, ローマ字)を打ち込もう
 - 基本的な日本語入力の仕方はWindowsと同じ
 - ただし, Windowsの入力モードとは無関係なので注意



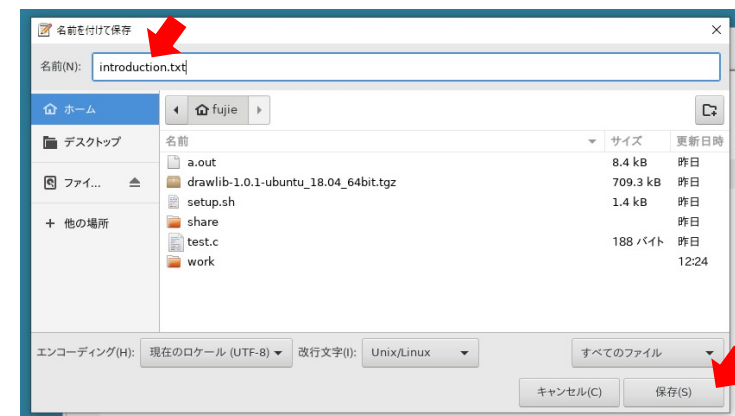
ファイルの保存

- 「保存」ボタンを押すか, メニューから「名前を付けて保存」を選択する



ファイルの保存

- 名前を introduction.txt にして, 「保存」ボタンを押す



ファイルを確認

■ ターミナルで「ls」と打ち込みエンター

```
fujie@prog:~$ ls
drawlib-1.0.1-ubuntu_18.04_64bit.tgz  setup
introduction.txt                      setup
fujie@prog:~$
```

- 先ほど作った introduction.txtがあることを確認

■ 続けて「cat introduction.txt」と打ち込みエンター

```
fujie@prog:~$ cat introduction.txt
2001987
藤江 真也
Shinya Fujie
fujie@prog:~$
```

- 先ほど保存した内容が表示されるか確認

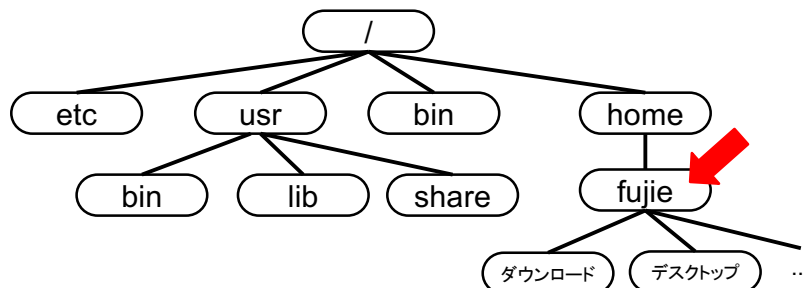
ディレクトリ

Linuxではディレクトリ
Windowsではフォルダ
と呼びます。

厳密には違うというはなしもありますが、同じものと考えてもらって問題ありません

カレントディレクトリ

■ カレントディレクトリ = 今いるディレクトリ



- ディレクトリの区切りは / (スラッシュ) で表す
- pwd コマンドでカレントディレクトリを確認できる
- コマンド実行時には常に意識する必要がある

ファイル・ディレクトリ操作に関するコマンド①

■ pwd

- カレントディレクトリがフルパスで表示される
- フルパスとは、ルートディレクトリからのパス (path, 道順) をすべて示したもの

■ cd [ディレクトリ]

- 指定された[ディレクトリ]へ移動する
 - ・ 移動先のディレクトリが**カレントディレクトリ**になる
- [ディレクトリ]が指定されなかった場合は**ホームディレクトリ**に移動する

ファイル・ディレクトリ操作に関するコマンド②

■ ls [ディレクトリ]

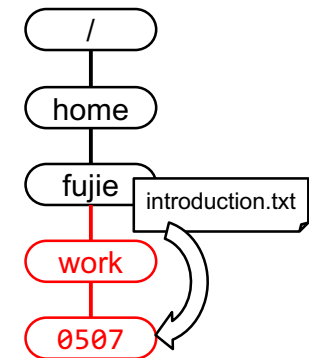
- 指定された[ディレクトリ]の内容を表示する
- [ディレクトリ]が指定されなかった場合はカレントディレクトリの内容を表示する
- よく使うオプション
 - -l ... 詳細情報(ファイルサイズ, 所有者)などを表示する
 - -a ... 隠しファイル(「.」で始まるファイル)も表示する

■ mkdir ディレクトリ

- 指定されたディレクトリを作成する
- よく使うオプション
 - -p ... 多階層のディレクトリを一度に作成する

練習

- ホームディレクトリの下に work, さらにその下に 0507 というディレクトリを作成する
- 作成したディレクトリに introduction.txt を移動する



特殊なディレクトリ名

■ ホームディレクトリ

- ホームディレクトリは ~ で表される

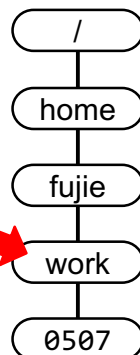
■ カレントディレクトリ

- カレントディレクトリは . で表される

■ 親ディレクトリ

- 親ディレクトリは .. で表される
- ここにいる場合

- ./ は /home/fujie/work/
- ../ は /home/fujie/
- ../../ は /home
- ../../../ は / を表す



練習

- ホームディレクトリに戻る
- work ディレクトリに cd する
- ../../.. に cd して, カレントディレクトリを確認する
- ~/work/0507 に cd する

注意とおすすめ

■ 注意

- 一つのディレクトリに同じ名前のファイルを複数作ることはできない
 - 上書きすると前の内容が失われる

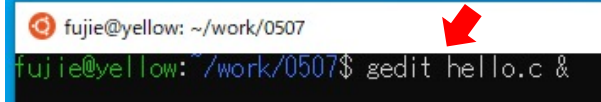
■ おすすめ

- 日付ごとや、課題ごとにディレクトリを分ける例)
5月7日の内容は ~/work/0507/ に入れる
第8回の課題は ~/kadai/08/ に入れるなど

C言語のプログラミング ～はじめの一步～

いよいよプログラミング

■ カレントディレクトリに注意して以下を実行



```
fujie@yellow: ~/work/0507  
fujie@yellow: ~/work/0507$ gedit hello.c &
```

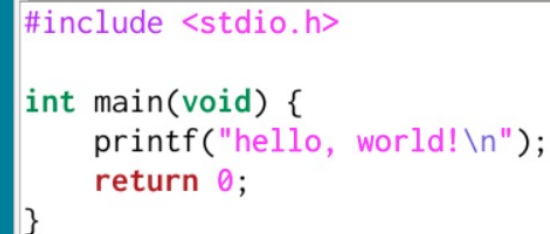
- gedit はコマンド名
- hello.c は作成・編集ファイルの名前

※既に gedit が起動している場合は、新しいタブが開かれる

プログラミング

■ 以下を入力して保存・終了

- 行番号は必要ない
- 「\」(バックスラッシュ)は、¥キーで入力可能



```
#include <stdio.h>  
  
int main(void) {  
    printf("hello, world!\n");  
    return 0;  
}
```

コンパイル

- まずファイルがあるか確認

```
fujie@prog:~/work/0605$ ls  
hello.c
```

- 続けてコンパイル

- gcc というコマンドを使う

```
fujie@prog:~/work/0605$ gcc hello.c -o hello
```

- 実行ファイル(hello)ができていることを確認

```
fujie@prog:~/work/0605$ ls  
hello hello.c
```

実行

- 実行はコマンドファイル名を指定すればできるはずだが、ディレクトリ(./)も含めなければならない

```
fujie@prog:~/work/0605$ ./hello  
hello, world!
```

練習

- hello, world! 以外の文字を表示するようにプログラムを変えてみる

- コンパイルをし直すのが必要なことに注意

余裕があったら考えてみよう

- ホームディレクトリに移動し、その状態で hello を実行してみる

- 実行できない... その理由は？
- ホームディレクトリにいながら hello を実行するにはどうすればよい？

プログラムの説明 基本単位

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

```
int main(void)
```

```
{  
    printf("hello, world¥n");  
    return 0;  
}
```

※スライド内ではバックスラッシュ(\\)が
円(¥)になっていますが読み替えてください

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

基本単位その1
“文(statement)”

```
int main(void)
```

```
{  
    printf("hello, world¥n");  
    return 0;  
}
```

文は基本的に
“;(セミコロン)”で終わる

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

基本単位その2
“ブロック(block)”

```
int main(void)
```

```
{  
    printf("hello, world¥n");  
    return 0;  
}
```

ブロックは,
“{} (波カッコ)”で囲む

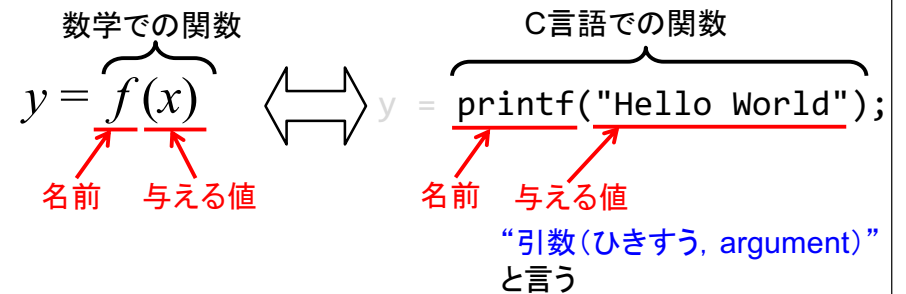
関数呼び出し

文にはどんなものがあるか？

- 変数宣言文 (variable declaration statement)
 - 代入文 (assignment statement)
 - **関数呼び出し文 (function calling statement)**
 - if文 (if statement)
 - switch文 (switch statement)
 - for文 (for statement)
 - while文 (while statement)
- などなど

C言語における関数 (function)

- 表し方は数学における関数と同じ



C言語では、関数を呼び出すことで
関数の内容に応じた処理が行われる

```
#include <stdio.h>

int main(void)
{
    printf("hello, world\n");
    return 0;
}
```

文に関数名と引数の内容を
与えることで関数呼び出し
ができる

ポイント!

- 関数の中身がどうなっているかを知る必要はない (=自分でプログラムする必要はない)
- 関数の仕様(名前, 引数の型)だけ知っていれば呼び出し可能

どんな関数があるか

- C言語標準ライブラリ (Wikipedia) などを見るとたくさん掲載されている
 - 標準なので、準拠している環境であればどれも使える
- ライブラリとは ... 関数の集合体
- ライブラリを使うには ... 対応したヘッダを読み込む必要がある

```
#include <stdio.h>

により, 関数 printf が使えるようになる
```

- 通常ヘッダの読み込みは最初に行う

関数定義

関数の型 引数の型
型(type)については後ほど詳しくやる
voidは特殊な型(なにも無いという意味)

```
#include <stdio.h>
int main(void)
{
    printf("hello, world¥n");
    return 0;
}
```

関数宣言 ブロック 関数名

関数宣言に続くブロックで、
その関数の処理内容を定義できる
(関数定義)

C言語のプログラムを実行すると
main関数が呼ばれる
という決まりがある

```
#include <stdio.h>
int main(void)
{
    printf("hello, world¥n");
    return 0;
}
```

従って、このプログラムは実質
printf("hello, world¥n");
を実行して終了する単純なもの

文字列

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

```
int main(void)
{
    printf("hello, world\n");
    return 0;
}
```

文字列 (string)

文字列は必ず
" (ダブルクォーテーション)
で囲む

printfは、引数に与えられた文字列を画面 (標準出力) に書き出す
処理をする

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

```
int main(void)
{
    printf("hello, world\n");
    return 0;
}
```

文字列の中の一つ一つは
文字 (character)

“\ (バックスラッシュ)” または “¥ (エン)” と組みになった
文字は、特殊文字 (special character) である。
¥n は改行を表す
¥n が無かったらどうなるか... ?

printf関数

printf関数

■ printf についてさらに詳しく...

1. printf("書式文字列");
2. printf("書式文字列", 引数1, 引数2, ...);

■ printf("%s¥n", "C is a programming language.");

- %s ... 変換指定子といい、引数として受け取ったデータを文字列として表示する。
- %d ... 整数として表示するための、変換指定子
- %f, %e ... 実数の変換指定子

■ エラー時のコンパイラの動作を確認しよう。

drawlib

参考資料のtest.c

```
1 #include <drawlib.h>
2
3 int main(void) {
4     dl_initialize(1.0);
5
6     dl_line(10, 10, 100, 100, dl_color_from_name("blue"), 2);
7
8     while (1) {
9         dl_wait(0.01);
10    }
11
12    return 0;
13 }
```

一見複雑だが実は単純

```
1 #include <drawlib.h>
2
3 int main(void) {
4     dl_initialize(1.0);
5
6     dl_line(10, 10, 100, 100, dl_color_from_name("blue"), 2);
7
8     while (1) {
9         dl_wait(0.01);
10    }
11
12    return 0;
13 }
```

関数呼び出し文①

関数呼び出し文②

関数呼び出し文③

一見複雑だが実は単純

```
1 #include <drawlib.h>
2
3 int main(void) {
4     dl_initialize(1.0);
5
6     dl_line(10, 10, 100, 100, dl_color_from_name("blue"), 2);
7
8     while (1) {
9         dl_wait(0.01);
10    }
11
12    return 0;
13 }
```

初期化をする関数で、drawlibを使うときは、必ず最初に呼び出す必要あり

プログラムを即座に終了しないために繰り返しを行う部分

表示されるウィンドウが小さすぎる人は、dl_initialize関数にあたる1.0を適当な大きい値にしてください(実数可)。

一見複雑だが実は単純

```
1 #include <drawlib.h>
2
3 int main(void) {
4     dl_initialize(1.0);
5
6     dl_line(10, 10, 100, 100, dl_color_from_name("blue"), 2);
7
8     while (1) {
9         dl_wait(0.01);
10    }
11
12    return 0;
13}
```

何を描画するかは
この部分だけで決まっている

dl_line関数の詳細

引数① (X座標1) 引数② (Y座標1) 引数③ (X座標2) 引数④ (Y座標2)

関数名

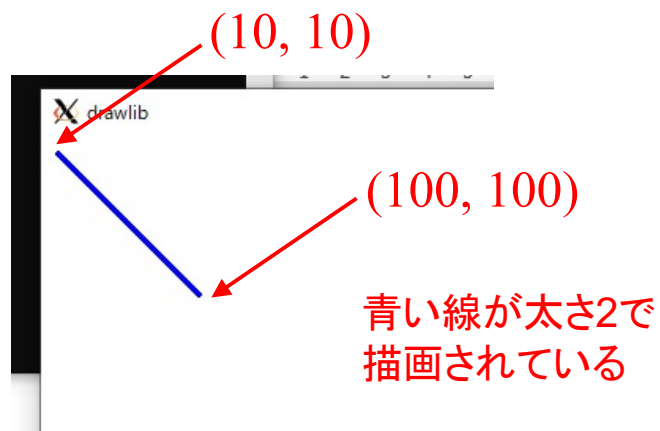
dl_line(10, 10, 100, 100, dl_color_from_name("blue"), 2);

引数⑤ (色)

引数⑥ (太さ)

色を指定する引数だけ特殊.
他は整数値で指定する.

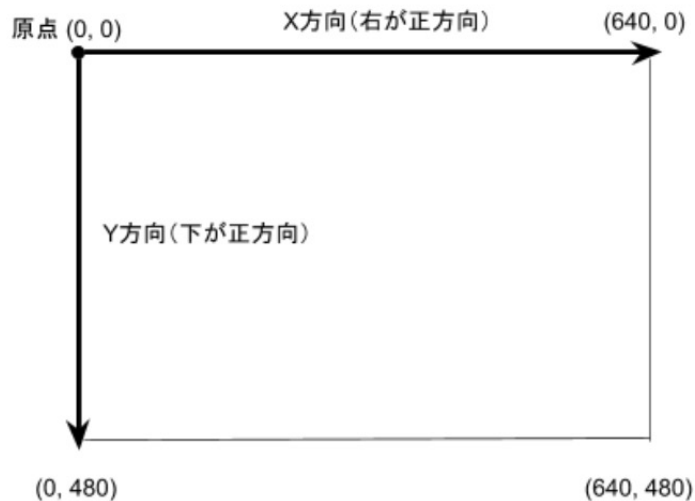
```
dl_line(10, 10, 100, 100,
        dl_color_from_name("blue"), 2);
```



練習

- 1. dl_line関数の呼び出しで、第1引数～第4引数の値を適当に変えて実行し、どのように描画のされ方が変わるか確認する.
- 2. 色の指定を変えてみる.
- 3. 更に別の線や、四角形などを加えてみる

drawlibの座標



色の指定方法(色の名前を使う場合)

`dl_color_from_name("blue")`

ここの文字列を指定したい色の名前に変える

- black … 黒
- red … 赤
- yellow … 黄
- magenda … マゼンタ(紫)
- green … 緑
- cyan … シアン(水色)
- blue … 青
- white … 白

これらの色以外を使いたい場合は
`dl_color_from_rgb`関数を使う

その他の描画関数

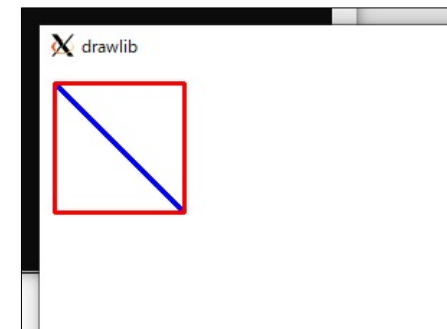
- `dl_line`は、2点を結ぶ線分を描画する。
- `dl_rectangle`は、2点を頂点とする矩形(長方形または正方形)を描画する。塗りつぶすこともできる。
- `dl_circle`は、円を描画する。中心点の座標と半径を与える。塗りつぶすこともできる。
- `dl_ellipse`は、楕円、もしくは円弧を描画する。中心点の座標、長径と短径、傾ける角度、開始角度、終了角度を与える。塗りつぶすこともできる。
- `dl_text`は、文字列を描画する。

それぞれ引数の数や意味が異なるので
気を付けよう

詳しくはリファレンスを見よう

例えば...

```
6 dl_line(10, 10, 100, 100, dl_color_from_name("blue"), 2);
7 dl_rectangle(10, 10, 100, 100, dl_color_from_name("red"), 2, 0);
```



レポート

レポート

- 提出期限 5月11日(火) 23:59
※手元にPCが無く作業できない人はその旨TAに伝えておいてください
- 提出先と提出物
 - manabaの指定場所
 - プログラムのソースコード(〇〇.c)と, 説明を書いたテキストファイル(〇〇.txt)を添付
- 課題内容
 - drawlibを使ったプログラムで, 描画関数を5回以上呼び出して適当な描画をするプログラムを作成して提出しなさい

レポートの提出方法

- ① Ubuntu 上でウェブブラウザを使う
 - ※firefoxが使えない場合は, epiphany を使う

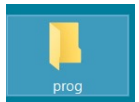
```
$ sudo apt install epiphany-browser
```


でインストール. 使う際は

```
$ epiphany
```
- ② gedit からメモ帳にコピー & ペーストして保存し, アップロードする
- ③ Windows を経由する(難易度高)
 - Windows側のフォルダにファイルをコピーする
 - パスが複雑なのでリンクを張る(次頁で説明)

Windows経由で課題を提出

- ① デスクトップに prog というフォルダを作成
- ② コマンドラインで prog の存在を確認
 - Windowsのユーザフォルダを確認



```
fujie@prog:~$ ls /mnt/c/Users
ls: シンボリックリンク '/mnt/c/Users/All Users' を読み
      取できません
ls: シンボリックリンク '/mnt/c/Users/Default User' を読
      取できません
'All Users'  'Default User'  desktop.ini  prfujie
Default     Public          fujie
```

prfujie はWindowsユーザ名. 人によって異なる.

```
fujie@prog:~$ ls -al /mnt/c/Users/prfujie/Desktop/prog/
合計 0
drwxrwxrwx 1 fujie fujie 512  6月  2 13:28
drwxrwxrwx 1 fujie fujie 512  6月  2 13:28
```

Windows経由で課題を提出

■ ③ リンク(シンボリックリンク)をはる

```
fujie@prog:~$ ln -s /mnt/c/Users/prfujie/Desktop/prog ~
```

以上で

Ubuntu側で ~/prog というフォルダにコピーされたファイルは
Windows側で デスクトップのprogフォルダ内にあることになる
※以上の作業は1度行えば以降は必要ない

■ 課題提出の際は以下のような作業

- 提出するファイルを Ubuntu 上で作成する
- ~/prog に提出するファイルをコピーする
- Windowsのブラウザ(Edgeなど)でmanabaにアクセスし、デスクトップのprogフォルダ内のファイルを添付する