Linux 基本操作（本選競技時配布）

コマンドライン端末の操作方法

ユーザ joi のディレクトリ

ホームディレクトリ	/home/joi
デスクトップ	/home/joi/デスクトップ

端末からは以下のような操作で作業ディレクトリに移動できる：

```
$ cd ~/joi2020
$ cd ~/joi2020/2020-ho-t1
```

ディレクトリの指定例

カレントディレクトリ	. (ピリオド 1 つ)
カレントディレクトリの親ディレクトリ	.. (ピリオド 2 つ)
自分のホームディレクトリ	~ (チルダ)
ルートディレクトリ	/

基本コマンド・実行例

PWD (Print Working Directory; カレントディレクトリの表示)

```
$  pwd
```

cd (Change Directory; カレントディレクトリの変更)

```
$  cd  (ホームディレクトリに移動)
$  cd  ~/joi2020/2020-ho-t2
(ホームディレクトリにある joi2020/2020-ho-t2 に移動)
```

ls (LiSt contents of directory; ディレクトリやファイルの情報を表示)

```
$ ls      (カレントディレクトリにあるファイルやディレクトリの一覧)
$ ls ~/   (ホームディレクトリにあるファイルやディレクトリの一覧)
$ ls ..   (親ディレクトリにあるファイルやディレクトリの一覧)
$ ls ~/joi2020  (ホームディレクトリにある joi2020 の一覧)
$ ls ~/joi2020/2020-ho-t2  (ホームディレクトリにある joi2020/2020-ho-t2 の一覧)
$ ls -F  ~/joi2020  (-F オプションはファイルタイプを示す印を付加)
$ ls -l  test.c  (-l オプションはファイルの詳細情報を long format で表示)
$ ls -l  ~/joi2020
```

mkdir (MaKe DIRectory; ディレクトリの作成)

```
$ mkdir my_sol  (カレントディレクトリの my_sol というディレクトリを作成)
```

cp (CoPy: ファイルやディレクトリのコピー)

```
$ cp test.c test_old.c  (test.c を test_old.c というファイル名でコピー)
$ cp test.c my_sol  (my_sol がディレクトリの場合は，test.c を my_sol にコピー)
```

mv (MoVe: ファイルやディレクトリの移動)

```
$ mv test.c test_old.c  (test.c を test_old.c というファイル名に変更)
$ mv test.c my_sol  (my_sol がディレクトリの場合は，test.c を my_sol に移動)
```

rm (ReMove: ファイルやディレクトリの消去)

```
$ rm test.c  (test.c を消去)
```

コンパイル・リンク方法

本選競技で使用できるプログラミング言語は C (C のバージョンは C11) および C++ (C++ のバージョンは C++14).

本選競技システムで使われるコンパイルオプションは，本選競技の際に配布される Overview Sheet に記載する.

本選競技実施時に，競技システムにおけるコンパイルオプションを競技参加者が変更することはできない.

gcc (C 言語) でのコンパイル例

```
$ gcc -std=gnu11 -Wall -O2 -o test test.c -lm
  (手元でコンパイルしてテスト実行する場合)
$ gcc -std=gnu11 -DEVAL -static -O2 -o test test.c -lm
  (コンパイルオプション -std=gnu11 -DEVAL -static -O2 -lm でコンパイルする場合)
```

g++ (C++14) でのコンパイル例

```
$ g++ -std=gnu++14 -Wall -O2 -o test test.cpp
  (手元でコンパイルしてテスト実行する場合)
$ g++ -std=gnu++14 -DEVAL -static -O2 -o test test.cpp
  (コンパイルオプション -std=gnu++14 -DEVAL -static -O2 でコンパイルする場合)
```

実行方法

カレントディレクトリにある test という実行ファイルを実行する場合.

```
$ ./test
$ ./test < myinput.txt
$ ./test < in/sample1.in
```

ファイルの内容を標準入力に与えたい場合は，上記のようにリダイレクションできる.

コンパイルオプション (補足)

-std=gnu11 gcc において C11 に準拠した言語仕様でコンパイルする (GNU 拡張を有効にする).

-std=gnu++14 g++ において C++14 に準拠した言語仕様でコンパイルする (GNU 拡張を有効にする).

-o リンカにより生成される実行ファイル名を指定する.

-O2 コンパイルの最適化に関するオプション. このオプションの有無によって, プログラムの実行速度が大幅に変わることがある. 手元でコンパイルしてテスト実行する際には, このオプションを常に付けることを勧める.

-Wall コンパイル時の警告オプションをすべて有効にする．手元でコンパイルしてテスト実行する際には, このオプションを常に付けることを勧める.

-lm 本選競技で使用する gcc では, 数学に関する標準 C ライブラリ (math.h) に含まれている関数 (sin, cos など) を使う場合にコンパイルオプション -lm が必要. (本選競技で使用する g++ では, コンパイルオプション -lm はデフォルトのため, このオプションは不要.)

-DEVAL マクロ名 EVAL を 1 と定めるコンパイルオプション. ソースの先頭に #define EVAL 1 と記述することと同等の意味を持つ. マクロ名 EVAL を用いたソースを書くことで, 手元でのコンパイル結果と本選競技システム上のコンパイル結果を意図的に変えることができる. 例えば, ソース中に

```
#ifndef EVAL
printf("debug:%d\n",i);
#endif
```

と書くことで, 手元の実行時にはデバッグ情報を出力して, 本選競技システム上での実行時にはデバッグ情報を出力しない, といったことも可能となる.

-static リンカに対するオプション. リンク時に実行バイナリにライブラリが組み込まれる. このオプションの有無によって, プログラムの実行速度が大幅に変わることは基本的にない. 本選競技システムと同一条件でコンパイル・テスト実行するときは, このオプションを付けることを勧める.

-pipe コンパイラの実行方法に対するオプション. コンパイル時に一時ファイルを使わずパイプライン処理を行う. このオプションの有無によって, プログラムの実行速度が変わることは基本的にない.

-s リンカに対するオプション. リンク時に実行バイナリからシンボルテーブルと再配置情報を削除する. このオプションの有無によって, プログラムの実行速度が変わることは基本的にない.