

システムプログラミング序論

コマンドライン引数と ファイル入出力（１）

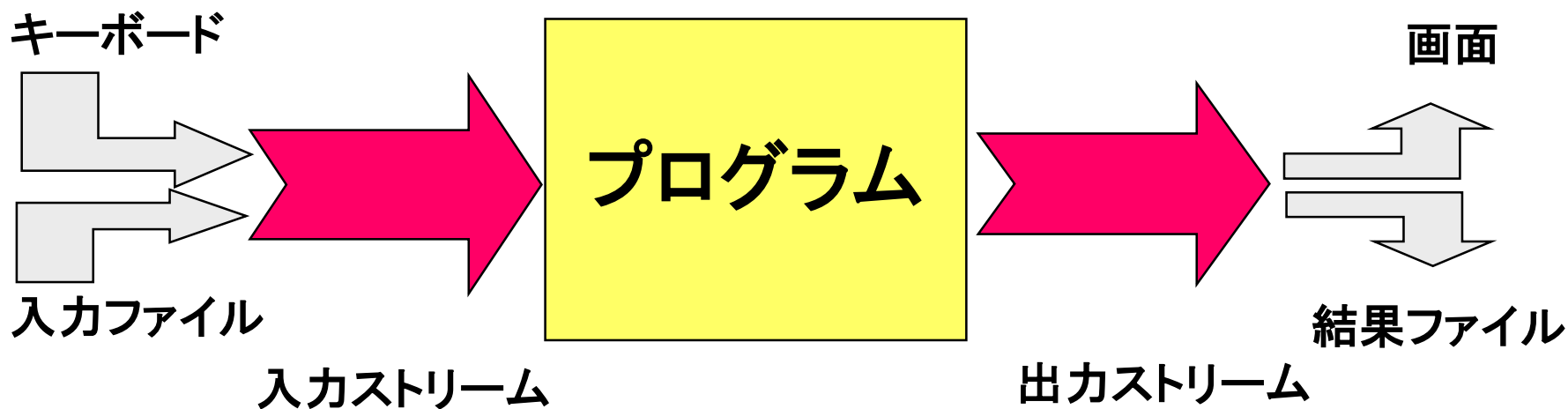
これまでのおさらい（入出力）

- ◆ これまでの入出力は
 - 入力 `scanf`
 - 出力 `printf`
 - キーボードと画面（端末）
- ◆ `scanf/printf`は、書式つき入出力
 - フォーマットを指定する
- ◆ 標準入出力を対象とする
 - 何もしなければ、標準入出力は、キーボードと画面

ストリームという考え方

◆ ストリーム(stream) = データの列

- キーボードから打つ文字列
- 画面に出力される文字列
- ファイル



ファイルのストリーム

◆ ファイルのオープン

- 操作を始める前に、ストリームとファイルを結びつける

◆ ファイルの操作

- 読み／書き

◆ ファイルのクローズ

- 操作が終わった後に、ストリームを切り離す

ファイルのオープン

- ◆ ストリームは、ファイルポインタ (FILE *)であらわされている。
- ◆ fopen関数でファイルをオープンし、ストリームを返す

```
FILE *fp;  
...  
fp = fopen("test.txt", "w");
```

fopen(ファイル名, オープンモード)

- ◆ ファイル名: オープンしたいファイル名
- ◆ オープンモード: 読みこみ" r", 書き込み" w"

ファイルのオープン

- ◆ オープンに失敗した時には、0 (= NULL)が返る
 - 読み込み “r”でオープンしようとして、ファイルがなかった場合
 - 書き込み “w”で、ファイルが書き込み禁止になっていた場合
- ◆ ファイルが正常にオープンされたかチェックすること

```
fp = fopen("test.txt", "r");  
if(fp == NULL) printf("error!");
```

- ◆ 書き込みの場合は既存の内容はクリアされるので注意

ファイルの読み書き

- ◆ ストリームに対する書式付きのscanf/printf

`fscanf` (ストリーム、フォーマット、引数...)

`fprintf` (ストリーム、フォーマット、引数...)

- ◆ ストリームは、`fopen`で得たファイルポインタ

ファイルのクローズ

- ◆ ストリームをファイルから切り離す

`fclose` (ストリーム)

- ◆ ストリームは、`fopen`で得たファイルポインタ
- ◆ 成功した場合は 1、失敗した場合は 0 を返す
- ◆ 書き込んだ内容は、`fclose`しないと全部かきこまれないので注意。

stdio.h

- ◆ 以上の入出力ストリームを扱う関数は、stdio.hに定義されているので、stdio.hをincludeすることをわすれないこと

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

- ◆ NULL (= 0) も定義されている

例

```
#include <stdio.h>

main()
{
    FILE *fp;
    fp = fopen("test.txt", "w");
    if(fp == NULL){
        printf("error!!");
        exit(1);
    }
    fprintf(fp, "this is sample¥n");
    fclose(fp);
    return 0;
}
```

例

```
#include <stdio.h>

main()
{
    FILE *fp; int x;
    fp = fopen("test.txt", "r");
    if (fp == NULL) {
        printf("error!!");
        exit(1);
    }
    fscanf(fp, "%d", &x);
    printf("file contains %d\n", x);
    fclose(fp);
    return 0;
}
```

いろいろなストリーム入出力関数

- ◆ 一文字の入力 `fgetc`
`char fgetc(FILE *fp)`
- ◆ 一文字の出力 `fputc`
`int fputc(int c, FILE *stream);`
- ◆ 行ごとの入力 `fgets`
`char *fgets(char *s, int size, FILE *stream);`
- ◆ 行ごとの出力 `fputs`
`int fputs(const char *s, FILE *stream);`
- ◆ `getchar`, `getc`, `gets`, `putchar`, `putc`, `puts`との違いを調べておくこと。

いろいろなストリーム入出力関数

- ◆ データ（バイナリ、構造体）の入力

```
int fread(void *ptr, int t size,  
          int nmemb, FILE *stream);
```

- ◆ データ（バイナリ、構造体）の出力

```
int fwrite(const void *ptr, int size,  
           int nmemb, FILE *stream);
```

- ◆ Read, write との違いを調べておくこと。

標準入出力

- ◆ 実は、defaultの入出力はプログラムの起動時にオープンされている
- ◆ 標準入力 `FILE *stdin`
 - なにもしなければ、キーボード
- ◆ 標準出力 `FILE *stdout`
 - なにもしなければ、画面
- ◆ これらの変数 `stdin`, `stdout` は、`stdio.h` に定義されている。使う時には `stdio.h` を `include`.

scanf/printfとfscanf/fprintf

- ◆ scanfは、stdinにfscanfをする関数

```
scanf (フォーマット、引数) ==  
fscanf (stdin, フォーマット、引数)
```

- ◆ printfは、stdoutにfprintfをする関数

```
printf (フォーマット、引数) ==  
fprintf (stdout, フォーマット、引数)
```

標準入力の変え方

◆ コマンドラインで標準入力を変えることができる

◆ 標準入力の変え方

- Shellでリダイレクト>する
- ファイルinput.txtを標準入力にする
a.out < input.txt

◆ 標準出力の変え方

- Shellで、リダイレクト>する
- ファイルoutput.txtを標準出力にする
a.out > output.txt

◆ パイプ

コマンドラインの引数の使い方

- ◆ 実行するときにファイル名などを指定したい
- ◆ main関数の引数として、アクセスする

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

- ◆ argcは、コマンドラインの引数の数
- ◆ argvは、コマンドラインの引数（文字列）の配列
- ◆ 1 番目（0）は、プログラム名

例

```
int main(int argc, char *argv[]) {  
    int i;  
  
    printf("number of argument is %d\n", argc);  
    for(i = 0; i < argc; i++)  
        printf("arg %d is %s\n", i, argv[i]);  
  
    return 0;  
}
```

演習1

```
#include <stdio.h>
int main(int argc, char *argv[])
{
    int ch, count;
    FILE *fp;
    /* コマンドライン引数が1つ指定されているかどうか確認する */
    if (argc != 2) {
        printf("missing file argument¥n");
        return 1;
    }
    if ((fp = fopen(argv[1], "r")) == NULL) {    /* ファイルのオープン */
        printf("can't open %s¥n", argv[1]);
        return 1;
    }
    count = 0;
    while ((ch = fgetc(fp)) != EOF) {    /* 1文字ずつ読み込む */
        if (ch == '¥n')
            count++;
    }
    printf("%d lines¥n", count);
    fclose(fp);    /* ファイルのクローズ */
    return 0;
}
```

演習2

```
#include <stdio.h>

int main(int argc, char *argv[])
{
    int ch;
    FILE *sfp, *dfp;
    /* コマンドライン引数が2つ指定されているかどうか確認する */
    if (argc != 3) {
        printf("missing file argument¥n");
        return 1;
    }
    if ((sfp = fopen(argv[1], "r")) == NULL) {    /* コピー元ファイルのオープン失敗 */
        printf("can't open %s¥n", argv[1]);
        return 1;
    }
    if ((dfp = fopen(argv[2], "w")) == NULL) {    /* コピー先ファイルのオープン失敗 */
        printf("can't open %s¥n", argv[2]);
        fclose(sfp);    /* コピー元ファイルのクローズ */
        return 1;
    }
    while ((ch = fgetc(sfp)) != EOF) {
        fputc(ch, dfp);    /* コピー先ファイルにコピー元ファイルの内容を書き込む */
    }
    fclose(dfp);    /* コピー先ファイルのクローズ */
    fclose(sfp);    /* コピー元ファイルのクローズ */
    return 0;
}
```

今日の内容

- ◆ 入出力とは
- ◆ ストリーム
- ◆ 入出力関数
- ◆ 標準入出力

- ◆ コマンド引数