C言語の基礎10

ファイル操作

ファイルポインタ

> https://ja.cppreference.com/w/c/io

- `<stdio.h>` にてサポートされている
  - ファイルには、標準ストリームを含む
    - `stdin` ：標準入力
    - `stdout` ：標準出力
    - `stderr` ：標準エラー出力
- ファイルの入出力はファイルポインタを利用して操作する

```
FILE *fopen(const char *filename, const char *mode);
```

- mode

モード	説明
`r`	読み込みモードで開く
`w`	書き込みモードで開く （ファイルが既に存在する場合は上書き。存在しない場合は新規）
`a`	追加（append）モードで開く（ファイルの末尾に追加される）
上記モード+`+`	更新モードで開く（読み込みと書き込みの両方が可能） `r+` 、 `w+` 、 `a+`
上記モード+`b`	バイナリモードを指定 `rb` 、 `wb` 、 `ab`

【サンプルデータ】example.txt

```
1-ABCDwxyz1234
2-ABCDwxyz1234
3-ABCDwxyz1234
4-ABCDwxyz1234
5-ABCDwxyz1234
6-ABCDwxyz1234
```

【`fopen`具体的な使い方 file01.c】

```c
#include <stdio.h>

int main(int argc, char const *argv[]){
    FILE *file;
    file = fopen("example.txt", "r");
    if (file == NULL) {
        printf("ファイルを開けませんでした。\n");
        return 2;
    }
    // ファイルの操作
    //   読み込み・書き込み等の処理
    //
    fclose(file); // ファイルを閉じる
    return 0;
}
```

【`fopen`具体的な使い方 file02.c】

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

int main(int argc, char const *argv[]){
    FILE *fp;
    if ((fp = fopen("example.txt", "r")) == NULL) {
        printf("ファイルを開けませんでした。\n");
        exit(2);
    }
    // ファイルの操作
    //   読み込み・書き込み等の処理
    //
    fclose(fp); // ファイルを閉じる
    return 0;
}
```

ファイル入出力関数

- 主によく使用される関数

関数名	書式	処理
`fgetc()`	int fgetc(FILE *fp);	`*fp` で示されるストリームから１文字を取得
`fgets()`	char *fgets(char *s, int n, FILE *fp);	`*fp` で示されるストリームから「n-1」文字を取得し `\0` を付加 その文字列の先頭アドレスを引数のchar型ポインタ `*s` に渡し、戻り値としても返す
`fscanf()`	int fscanf(FILE *fp, const char *format, ...);	`*fp` で示されるファイルからの入力値を書式指定した文字列として取得 第3引数以降は「const char *format」の変換指定子の数だけ変数を指定する `scanf()` の引数にファイルポインタが増えただけ

ファイル出力関数

- 主によく使用される関数

関数名	書式	処理
`fputc()`	int fputc(int c, FILE *fp);	キャラクタ（ASCIIコード）cをストリームに出力
`fputs()`	int fputs(const char *s, FILE *fp);	文字列sをストリームに出力
`fprintf()`	int fprintf(FILE *fp, const char *format,...);	書式指定文字列をストリームに出力。 `printf()` にファイルポインタが増えただけ

【`fgetc` 具体的な使い方 file03.c】

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

int main(int argc, char const *argv[]) {
    FILE *fp;
    if ((fp = fopen("example.txt", "r")) == NULL) {
        printf("ファイルを開けませんでした。\n");
        exit(2);
    }

    char ch;
    printf("ファイルexample.txtの最初の10文字\n");
    for (int i = 1; i <= 10; i++) {
```

```
        ch = fgetc(fp);
        printf("%02d => (%02x) %c\n", i, ch, ch);
    }

    printf("ファイルexample.txtの次の10文字\n");
    for (int i = 11; i <= 20; i++) {
        ch = fgetc(fp);
        printf("%02d => (%02x) %c\n", i, ch, ch);
    }

    fclose(fp); // ファイルを閉じる
    return 0;
}
```

【 `fgets` の具体的な使い方 file04.c】

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

int main(int argc, char const *argv[]) {
    FILE *fp;
    if ((fp = fopen("example.txt", "r")) == NULL) {
        printf("ファイルを開けませんでした。\n");
        exit(2);
    }

    char buff[100];
    printf("ファイルexample.txtの最初の2行\n");
    for (int i = 1; i <= 2; i++) {
        if (fgets(buff, 100, fp) != NULL) {
            printf("%02d => %s\n", i, buff);
        } else {
            printf("終端まで読み込みました\n");
        }
    }

    printf("ファイルexample.txtの次の10行\n");
    for (int i = 3; i <= 13; i++) {
        if (fgets(buff, 100, fp) != NULL) {
            printf("%02d => %s\n", i, buff);
        } else {
            printf("%02d : 終端まで読み込みました\n", i);
        }
    }

    fclose(fp); // ファイルを閉じる
    return 0;
}
```

練習

練習1

- プログラム名（prog1.c）
- `prog1.c`（自分自身のソースコード）を読み込み、画面に出力するプログラム
- ただし、先頭に行番号を表示する。
- ファイルの1行は最大で256文字とする。

【実行例】

- 例では、file01.cを出力している。

```
$ ./a.out
0001:#include <stdio.h>
0002:#include <stdlib.h>
0003:
0004:int main(int argc, char const *argv[]){
0005:    FILE *fp;
0006:    if ((fp = fopen("example.txt", "r")) == NULL) {
0007:        printf("ファイルを開けませんでした。\n");
0008:        exit(2);
0009:    }
0010:    // ファイルの操作
0011:    //   読み込み・書き込み等の処理
0012:    //
0013:    fclose(fp); // ファイルを閉じる
0014:    return 0;
0015:}
```

練習2

- 毎回、ファイル名を書き換える度に、コンパイルし直すのは手間である。
    - コマンドライン引数を使用し、出力するようにしてください。（prog2.c）
        - 複数のファイルを指定できるものとする。
        - ファイルごとに、行番号は1から開始するものとする。

【実行例1】

```
== filename:file01.c ==
0001: #include <stdio.h>
0002:
```

```
0003: int main(int argc, char const *argv[]) {
0004:     FILE *file;
0005:     file = fopen("example.txt", "r");
0006:     if (file == NULL) {
0007:         printf("ファイルを開けませんでした。\n");
0008:         return 2;
0009:     }
0010:     // ファイルの操作
0011:     //   読み込み・書き込み等の処理
0012:     //
0013:     fclose(file); // ファイルを閉じる
0014:     return 0;
0015: }

== filename:file02.c ==
0001: #include <stdio.h>
0002: #include <stdlib.h>
0003:
0004: int main(int argc, char const *argv[]) {
0005:     FILE *fp;
0006:     if ((fp = fopen("example.txt", "r")) == NULL) {
0007:         printf("ファイルを開けませんでした。\n");
0008:         exit(2);
0009:     }
0010:     // ファイルの操作
0011:     //   読み込み・書き込み等の処理
0012:     //
0013:     fclose(fp); // ファイルを閉じる
0014:     return 0;
0015: }
```

【実行例2】

```
$ ./a.out
使用方法：プログラム名 ファイル名1 ファイル名2 ...
```

【実行例3】

```
$ ./a.out test.txt file01.c
[ERROR] ファイル test.txt を開けませんでした。

0001: #include <stdio.h>
0002:
0003: int main(int argc, char const *argv[]) {
0004:     FILE *file;
0005:     file = fopen("example.txt", "r");
0006:     if (file == NULL) {
0007:         printf("ファイルを開けませんでした。\n");
0008:         return 2;
0009:     }
0010:     // ファイルの操作
```

```
0011:     //   読み込み・書き込み等の処理
0012:     //
0013:     fclose(file); // ファイルを閉じる
0014:     return 0;
0015: }
```

練習3

- ファイル名に「`-`」を指定した場合は、標準入力（デフォルトはキーボード）からの入力を行うようにしてください。（prog3.c）

【実行例1】

```
$ ./a.out
使用方法：プログラム名 ファイル名1 ファイル名2 ...
        ファイル名に '-' を使用した場合は標準入力
```

【実行例2】

```
$ ./a.out -
test string
0001: test string
abc
0002: abc
xyz
0003: xyz
      ←（ctrl+d で終了）
```

- 入力の終了は`ctrl`+`d`

Linux標準コマンド

- `cat`コマンドの使い方を調べてください。
    - 各オプションを良く確認し、利用してみてください。
- `nl`コマンドの使い方を調べてください。
    - 各オプションを良く確認し、利用してみてください。

（応用）CSVファイルを読み取るライブラリ

`strtok()`

> https://daeudaeu.com/c-strtok/#strtok-9

- 上記関数を利用した簡易的な方法は、多く存在する。
  - CSVのフォーマットルールはどうなっているか知っているだろうか？
- libcsvはLinuxで利用できる。

```
sudo apt install libcsv-dev
```

  - 上記コマンドでインストールした場合、`#include <csv.h>`と記述する
  - ドキュメントには`#include "libcsv/csv.h"`と書かれている
  - コンパイル時には、オプション`-lcsv`を付加する
- 使い方
  - コールバック関数を使用する必要あり

```
#include <csv.h>
#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct counts {
    long unsigned fields;
    long unsigned rows;
};

void cb1(void *s, size_t len, void *data) {
    ((struct counts *)data)->fields++;
    printf("(%s)", (char *)s);  // 吉村が追加
}

void cb2(int c, void *data) {
    ((struct counts *)data)->rows++;
    printf("\t1行終了\n");       // 吉村が追加
}

int main(int argc, char *argv[]) {
    FILE *fp;
    struct csv_parser p;
    char buf[1024];
    size_t bytes_read;
    struct counts c = {0, 0};

    if (csv_init(&p, CSV_APPEND_NULL) != 0)
        exit(EXIT_FAILURE);
    fp = fopen(argv[1], "rb");
    if (!fp)
```

```
        exit(EXIT_FAILURE);

    while ((bytes_read = fread(buf, 1, 1024, fp)) > 0)
        if (csv_parse(&p, buf, bytes_read, cb1, cb2, &c) != bytes_read) {
            fprintf(stderr, "Error while parsing file: %s\n",
                    csv_strerror(csv_error(&p)));
            exit(EXIT_FAILURE);
        }
    fclose(fp);
    csv_fini(&p, cb1, cb2, &c);
    printf("%lu fields, %lu rows\n", c.fields, c.rows);
    csv_free(&p);
    exit(EXIT_SUCCESS);
}
```