今回は、ファイルの読み書きについて学ぶ。教科書の P.194 〜 P.202 の内容が対応する。３年次に作ったプログラムは stdin から入力（getchar()）し、stdout へ出力（putchar()、printf()）するものだった。今回は、任意のファイルを読み書きするプログラムを作る。

C 言語では、ファイルをバイトの１次元配列（バイトストリーム）と考える。stdin や stdout、stderr もバイトストリームだったので、これらと同様の手順でファイルも扱うことができる。

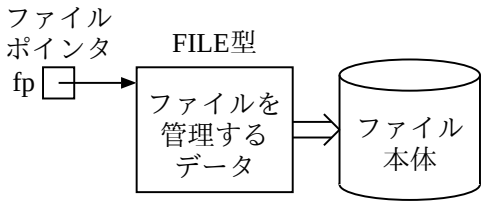
1. ファイルのオープン（fopen() 関数）

   stdin や stdout、stderr は、特別に予めオープンされたファイルである。普通のファイルはプログラム内で明示的にオープンする必要がある。fopen() がファイルをオープンする関数である。

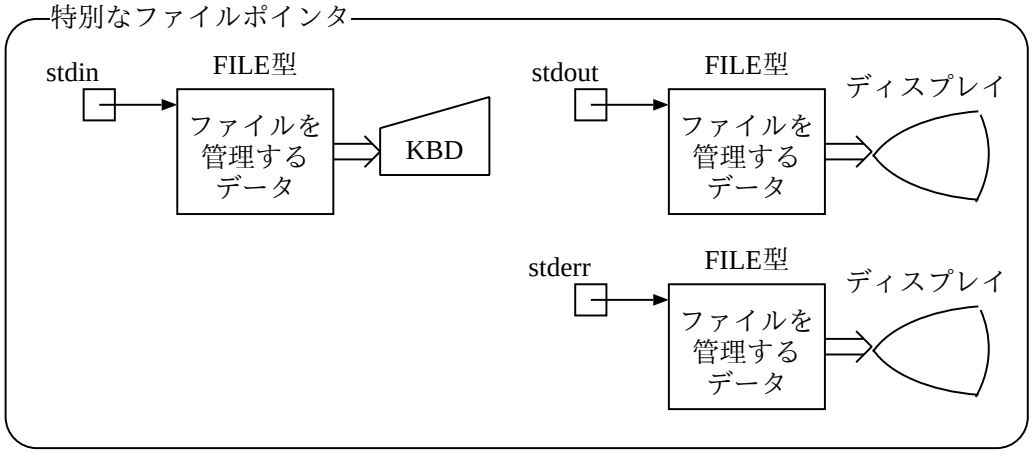
```
#include <stdio.h>
FILE *fopen(char *fname, char *mode);

例：// バイナリファイルをリードオープン
     FILE *fp;        // ファイルポインタ
     fp = fopen("a.dat", "rb");
     if (fp==NULL) {   // エラーチェック
```

fname：ファイル名
mode ：
　”r”：読み (read)
　”w”：書き (write)　＋　”b”：バイナリ
　”a”：追加 (append)
返り値：FILE 構造体を指すポインタ



fopenがFILE型構造体領域を作って初期化する。勿論、ファイル本体のオープンもする。



2. ファイル入出力

   (a) １文字入力（getc() 関数 — getchar() のファイル版 —）

```
#include <stdio.h>
int getc(FILE *fp);
// getchar() は次のように定義できる
// #define getchar()  getc(stdin)
```

getc は、ファイルから読んだ１バイトの値 (0〜255) を返す。EOF になると −1 を返す。（getchar() と同様）

```
例 :ファイルから１バイト読む
FILE *fp = fopen("a.txt", "r");
int ch;                    // int 型 ! !
ch = getc(fp);
```

左の例は、a.txt ファイルの最初の１バイトを変数 ch に読み込む。

(b) 1 文字出力（putc() 関数 — putchar() のファイル版 — ）

```
#include <stdio.h>
int putc(int c, FILE *fp);
// putchar() は次のように定義できる
// #define putchar()  putc(stdout)
```

```
例 :ファイルに文字「a」を書き込む
FILE *fp = fopen("a.txt", "w");
putc('a', fp);
```

(c) 1 行入力（fgets() 関数）

```
#include <stdio.h>
char *fgets(char *buf,
            int size, FILE *fp);
// fgets は EOF で NULL を返す
```

buf : 1 行入力バッファ
size : バッファの大きさ
fp : ファイルポインタ

```
例 :ファイルから 1 行読む
FILE *fp = fopen("a.txt", "r");
char buf[100];
fgets(buf, 100, fp);
```

左の例は、a.txt ファイルの最初の 1 行を buf に入力する。buf の最後には、"\n\0" が格納される。

(d) 1 行出力（fputs() 関数 — puts() のファイル版 —）

```
#include <stdio.h>
int fputs(char *buf, FILE *fp);
// エラー時に EOF を返す
// buf は'\0'で終わる文字列
```

```
例 :ファイルに文字列「abc\n」を書き込む
FILE *fp = fopen("a.txt", "w");
fputs("abc\n", fp);
```

(e) 書式付き出力（fprintf() 関数 — printf() のファイル版 — ）

```
#include <stdio.h>
int fprintf(FILE *fp,
            char *format, ...);
```

```
例 :ファイルにデータを書き込む
FILE *fp = fopen("a.txt", "w");
fprintf(fp, "%d,%d\n", x, y);
```

3. ファイルのクローズ（fclose() 関数）

```
#include <stdio.h>
int fclose(FILE *fp);
```

4. エラーメッセージの出力（perror() 関数）

直近に発生したエラーを説明するメッセージを stderr に出力する。

```
#include <stdio.h>
void perror(char *msg);  // msg はエラーメッセージの先頭に付加する文字列

例 :ファイルオープンに失敗のエラー処理
     fp = fopen("a.txt", "r");
     if (fp==NULL) {
       perror("a.txt");  // "a.txt: No such file or directory" などと表示
       exit(1);
     }
```

5. プログラム例

```
// mycat.c : 複数のファイルを stdout へ連続出力
#include <stdio.h>
#include <stdlib.h>  // exit のため

// ifp から ofp にコピー
void fileCopy(FILE *ifp, FILE *ofp) {
  int c;
  while ((c=getc(ifp))!=EOF)
    putc(c, ofp);
}

int main(int argc, char *argv[]) {
  if (argc==1)                          // ファイルが指定されていない
    fileCopy(stdin, stdout);            //   stdin から stdout へコピー
  else                                  // ファイルが指定されている
    for (int i=1; i<argc; i++) {        //   全てのファイルについて
      FILE *fp = fopen(argv[i], "r");   //     オープンして
      if (fp==NULL) {                   //     エラーチェックして
        perror(argv[i]);
        exit(1);
      }
      fileCopy(fp, stdout);             //     ファイルから stdout へコピー
      fclose(fp);                       //     忘れずクローズ
    }
  return 0;
}
```

実行例：

```
$ mycat                     <--- ファイルが指定されていない場合
aaaa                        <--- 入力
aaaa                        <--- 出力
bbbb                        <--- 入力
bbbb                        <--- 出力
^D                          <--- EOF
$ mycat a.txt
hello                       <--- a.txt の内容を表示
$ mycat b.txt
world                       <--- b.txt の内容を表示
$ mycat a.txt b.txt
hello                       <--- a.txt の内容を表示
world                       <--- b.txt の内容を表示
$
```

6. 演習（＃１）　〆切　４月１１日

   (a) mycp コマンド

   ファイルをコピーするプログラム mycp を作りなさい。mycp は、cp コマンドの簡易版です。次のように使います。エラー処理もすること。

   ```
   $ mycp file1 file2          <--- file1 の内容を file2 にコピー
   ```

   (b) mydiff コマンド

   二つのファイルを比較して、違っている最初の行と行番号を印刷するプログラムを書きなさい。処理できる１行の最大の長さは自分で決めてい良い。次のように使います。エラー処理もすること。

   ```
   $ mydiff a.txt b.txt        <--- a.txt と b.txt を比較
   2 行目
   > world
   < World
   $
   ```