

# C言語の学習 ファイル処理関数

山本昌志\*

2007年7月4日

## 概要

ファイルの取り扱い，とくにハードディスクにデータを保管する方法とそこからデータを読み出す方法を学習する．

## 1 本日の学習内容

本日は，教科書 [1] の 18 章のファイル処理について学習する．ハードディスクにデータを書き込んだり，ハードディスクからデータを読み込んだりするプログラムの書き方を学ぶ．本日の学習のゴールは，以下の通りである．

- ディスプレイに文字を打ち出すのと同じイメージで，ハードディスクにデータを保存できる．そのことを理解して，データを保存するプログラムが書ける．
- キーボードからデータを読み込むのと同じイメージで，ハードディスクのデータを読み込むことができる．そのイメージを理解して，データを読み込むプログラムが書ける．

## 2 ファイル処理の概要

### 2.1 ファイル処理とは

本講義のメインテーマである数値計算では大量の数値を扱い，それはハードディスクに保存することが多い．いちいち紙に計算結果を書き写すことは不可能なためである．数値計算に限らず，実験で取得したデータもハードディスクに保存することが多い．なにせ，データの取得にコンピューターを利用するのは普通のことである．そのデータを処理して，意味のある量に加工するのである．数値計算や実験など，理科系の仕事に限らず，現代社会での大量の情報は全てハードディスクに保存される—と言っても過言ではない．

このようなことから，ファイル処理—ここではハードディスクへのデータの入出力—は，コンピューターを上手に使う必須のテクニックである．ファイル処理の技術を習得し，コンピューターを自在に活用して欲しい．これができるようになると，コンピューターの応用がかなり広がる．卒研などで，大変重宝するだろう．

---

\*独立行政法人 秋田工業高等専門学校 電気工学科

## 2.2 ファイル処理の体験

### 2.2.1 ファイル出力

ごちゃごちゃと説明するよりも、実際にファイル処理のプログラムを示した方がわかりやすいだろう。いままで、ディスプレイに出力していた”Hello World !!”をファイルに書き出す。リスト 1 のプログラムでそれが実現できる。ディスプレイに表示する `printf()` という関数の代わりに、`fprintf()` 関数を使うのである。`fprintf()` 関数がファイルに文字を書き出すのである。前後に少し、おまけがついているが、ほとんど同じである。

リスト 1: ファイル出力の例

```
1 #include <stdio.h>
2
3 int main(void){
4     FILE *hoge;
5
6     hoge=fopen("hello.txt", "w");
7
8     fprintf(hoge, "Hello World !!\n");
9
10    fclose(hoge);
11
12    return 0;
13 }
```

#### 実行結果

リスト 1 を実行すると、`hello.txt` というファイルができあがる。そのファイルの中には、以下のような文字が書かれている。

```
Hello World !!
```

このプログラムの各行の内容は、次の通りである。文法の詳細については、後で説明する。`fprintf()` 関数でファイルに文字を書くことができる—ということを理解せねばならない。

- 4 行目 `FILE *hoge;`  
ファイルを識別するためのデータを入れる変数を準備する。
- 6 行目 `hoge=fopen("hello.txt", "w");`  
`hello.txt` というファイルを書き込み (write) モードで開いている。そして、ファイルの情報— 正確には情報が書かれたアドレス—をポインター `hoge` に代入する。
- 8 行目 `fprintf(hoge, "Hello World !!\n");`  
ファイルに `Hello World !!` と書く。書き込むべきファイルを `hoge` で示している。
- 10 行目 `fclose(hoge);`  
使い終わったファイルを閉じる。

[練習 1] 自分の名前を、ファイルに書き出すプログラムを作成せよ。そして、実行してファイルのデータを確認せよ。

[練習 2] ファイルに、以下の内容を書き出すプログラムを作成せよ。

```
=====
Akita National College of Technology
  Dept. of Electrical and Computer Engineering

Yamamoto Masashi

TEL/FAX 018-847-6044
e-mail yamamoto@akita-nct.jp
=====
```

### 2.2.2 ファイル入力

ファイルに文字を書く方法は分かった。次に、ファイルから文字を読み込んでみよう。先ほど作成したファイル (hello.txt) の内容を読み、それを画面に出力する。リスト 2 が、そのプログラムである。

リスト 2: ファイル入力の例

```
1 #include <stdio.h>
2
3 int main(void){
4     FILE *fuga;
5     char a[32], b[32], c[32];
6
7     fuga = fopen("hello.txt", "r");
8
9     fscanf(fuga, "%s%s%s", a, b, c);
10
11    fclose(fuga);
12
13    printf("%s %s %s\n", a, b, c);
14
15    return 0;
16 }
```

#### 実行結果

このプログラムを実行すると、hello.txt というファイルから文字を読み込み、ディスプレイに、以下のように表示される。

```
Hello World !!
```

このプログラムの各行の内容は、次の通りである。ここでの文法の詳細についても後で説明する。fscanf() でファイルから文字を読み取ることができる—ことを理解しなくてはならない。

- 4 行目 FILE \*fuga;  
先ほど同様、ファイルを識別するためのデータを入れる変数である。
- 7 行目 fuga = fopen("hello.txt", "r");

hello.txt というファイルを読み込み (read) モード開いている。戻り値であるファイルの情報は、ポインター hoge に代入している。

- 9 行目 `fscanf(fuga, "%s%s%s", a, b, c);`  
ファイルからデータ—文字列—を読み込んでいる。空白は文字列の区切りを表すので、文字列を入れる 3 つの配列を用意している。配列 a に”Hello”，配列 b に”World”，配列 c に”!!”が格納される。
- 11 行目 `fclose(fuga);`  
使い終わったファイルを閉じている。

[練習 3] 自分の名前が書かれたファイルを読み込んで、ディスプレイに表示するプログラムを作成せよ。

### 3 ファイル処理の方法

#### 3.1 ファイル処理の大まかな流れ

先ほどの例でファイル処理の大体的な方法を理解したと思う。C 言語に限らず、ほとんどのプログラム言語のファイルの処理はどれも似かよっている。それは、図 1 のようになっている。人間がデータを記録している本やノートなどを見る動作と同じようにしている。オープンと読み書き、クローズは約束事と理解する必要がある。

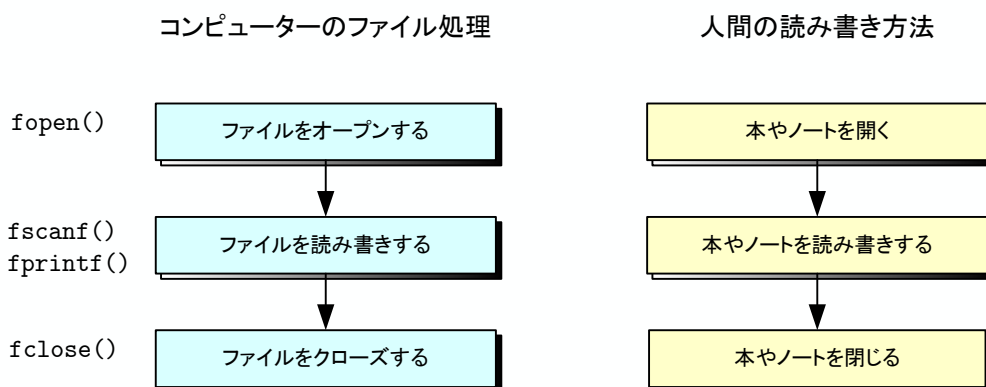


図 1: コンピューターのファイル処理と人間の読み書き方法

リスト 1 や 2 のプログラムの中で、これらの約束事の記述の例を図 2 に示す。人間の動作を考えれば、取り立ててその流れは難しくない。

- `fopen()` 関数でファイルをオープンしている。

- fprintf() や fscanf() 関数で、ファイルのデータを読み書きしている。
- fclose() 関数で、ファイルをクローズしている。

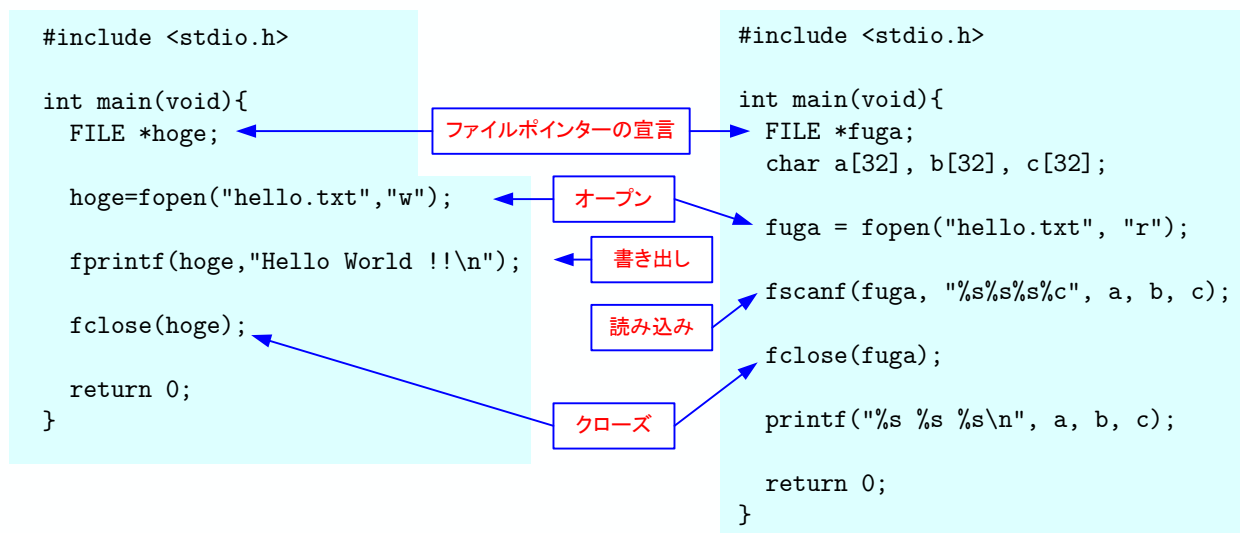


図 2: C 言語でのファイル処理の例

図 2 を見て分かるように、実際の C 言語ではオープンと読み書き、クローズの他に、ファイルポインタの宣言が必要である。ファイルポインタについては、次の節で述べることにする。C 言語のプログラムでファイル処理をする場合は、図 3 に示す手順に従えば良い。ただし、実際のプログラムではエラー処理を書かなくてはならないが、ここでは示していない。

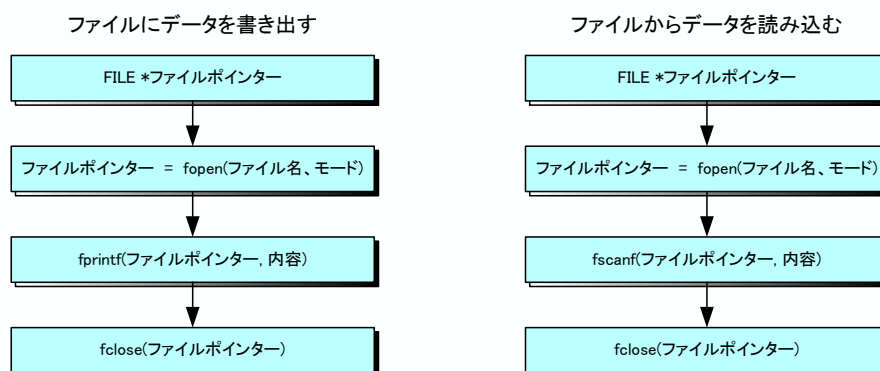


図 3: C 言語でのファイル処理の流れ

## 3.2 ファイルのオープンとクローズ

### 3.2.1 ファイルポインタの宣言

ファイル処理には多くの情報が必要で、それをメモリーの一部に格納する必要がある。その格納場所を示すものがファイルポインタである。それは構造体になっており、`stdio.h`というヘッダーファイルに以下のような内容が定義(教科書 p.377)されている。

- ファイルの読み書きをする場合の位置
- 残っている文字数
- バッファ領域の先頭位置
- ファイルの状態
- ファイル番号

このファイルポインタ (`fp`) を使って、全てのファイル関係の処理を行う。なにせ、ファイルに関する情報が全て書かれているので、これを指定すれば、あとはコンピューターが勝手に処理してくれる。面倒くさい処理はコンピューター任せにして、プログラマーは楽をしようということである。この `FILE` 型の変数(ポインタ)を使うためには、次のように宣言する。

```
FILE *foo;
```

これで、`FILE` 型の変数(ポインタ) `foo` を宣言できる。ただし、ここの `foo` は変数名なのでプログラマーが適当な名前をつけることができる。`FILE *hoge` でもよい。

### 3.2.2 ファイルのオープン

通常は、`fopen()` という関数の戻り値をこのポインタに代入する。このファイルをオープンする関数 `fopen()` の書式は、次の通りである。

```
FILE *fopen(char *filename, char *openmode)
```

戻り値は `FILE` 型のポインタ、ファイル名を表す第一引数は `char` 型のポインタ、オープンモードを表す第二引数は `char` 型のポインタとすることである。もし、オープンに失敗すると、`NULL` という戻り値になる。`char` 型のポインタと難しいことを言っているが、先のプログラムの例(リスト 1, 2)でも分かるように、文字列をダブルクォーテーションで囲めば良いのである。例えば、`hoge.txt` というファイルを読み込みモードでオープンする場合

```
foo=fopen("hoge.txt", "r");
```

と書く。

オープンモードについては、いろいろ用意されており、教科書の p.382 にまとめてある。細かいファイル処理をする場合は、これらのモードを巧みに使う必要があるが、本講義では、

- ファイルからデータを読み込む場合は、オープンモードの部分は `"r"`
- ファイルへデータを書き込む場合は、オープンモードの部分は `"w"`

とすればよい。バイナリーモードも UNIX(Linux) では関係ないので使わない。

### 3.2.3 ファイルのクローズ

ファイルをクローズする関数 `fclose()` の書式は、簡単で、次の通りである。

```
int fclose(FILE *filepointer)
```

戻り値は、`int` 型で、クローズに成功すると 0、失敗すると EOF が返される。引数は、ファイルポインターのみである。ファイルを開いたら閉じるのが礼儀だと心得て、処理の最後に

```
fclose(foo);
```

と必ず書く。

## 3.3 ファイルの入出力関数

いよいよ、ファイルのデータを読み書きするファイル入出力関数について説明する。難しと思うだろうが、実は非常に簡単である。いままで、標準入力(キーボード)と標準出力(ディスプレイ)に使ってきた関数、`printf()` と `scanf()` とほとんど同じである。付録に示すようにキーボードやディスプレイもファイルとして取り扱われるので、同じ手法がハードディスクにも使える。

### 3.3.1 入力 (`fscanf()`) 関数

ファイル入力であるが、標準入力(キーボード)とほとんど同じように取り扱える。これら二つの入力関数を並べて書くと

ファイル入力	<code>int fscanf(ファイルポインター, 書式指定, 引数並び)</code>
標準入力	<code>int scanf(書式指定, 引数並び)</code>

となる。ファイルポインターを指定する以外、すべて標準入力の場合と同じである。非常に単純で簡単である。実際の動作もキーボードからデータを入力するのも、ファイルから読み込むのも同じイメージで取り扱える。戻り値の整数は入力したデータの個数である。もし、ファイルの最後あるいは読み込みに失敗すると EOF を返す。

### 3.3.2 出力 (`fprintf()`) 関数

ファイル出力も、標準出力(ディスプレイ)とほとんど同じように取り扱える。これら二つの出力関数を並べて書くと

ファイル出力	<code>int fprintf(ファイルポインター, 書式指定, 引数並び)</code>
標準出力	<code>int printf(書式指定, 引数並び)</code>

となる。ハードディスクのファイルにデータを書き込むのは、ディスプレイにデータを出力するのと全く同じイメージである。実際、ファイル出力されたデータを見ると、ディスプレイと同じであることが分かる。戻り値の整数は出力した文字数である。書き込みに失敗すると負の値を返す。

これまでから、コンソール入出力とファイル入出力は同じ取り扱いができることが理解できたであろう。

## 4 数値計算のファイル入出力

数値計算を行うと、大量のデータが出力される。また、大量の数値を読み込むことも多い。そのため、ファイル—ハードディスクを使って、データの保管を行う。ここでは、今後の数値計算の講義でつかう基本的なテクニックをプログラムを例にとって、示す。

### 4.1 ファイル出力の実際

数値計算の出力は大量の数値データのことが多い。そのデータをファイルに保管する基本的なテクニックは、次の通りである。

- 表をイメージして、データをファイルに書き出す。
- 1行に複数のデータがある場合は、“\t”を用いてタブ区切り<sup>1</sup>とする。
- ループ文(for, while, do while)を用いて、fprintf() 関数を繰り返し使う。

これらを使った例をリスト 3 に示す。これは、三角関数の値をファイル出力するプログラムである。

リスト 3: 三角関数の値のファイル出力プログラム

```
1 #include <stdio.h>
2 #include <math.h>
3 #define N 360
4
5 int main(void)
6 {
7     FILE *out_file;
8     double x, dphi;
9     int i;
10
11     dphi = 2*M_PI/N;
12
13     out_file = fopen("trifunc.txt", "w");
14
15     for(i=0; i<=N; i++){
16         x=i*dphi-M_PI;
17         fprintf(out_file, "%15.7e\t%15.7e\t%15.7e\t%15.7e\n", x, sin(x), cos(x), tan(x));
18     }
19
20     fclose(out_file);
21
22     return 0;
23 }
```

#### 実行結果

このプログラムを実行すると「trifunc.txt」というファイルができる。このファイルの中は、次のようになっている。

```
-3.1415927e+00  -1.2246064e-16  -1.0000000e+00  1.2246064e-16
-3.1241394e+00  -1.7452406e-02  -9.9984770e-01  1.7455065e-02
-3.1066861e+00  -3.4899497e-02  -9.9939083e-01  3.4920769e-02
```

<sup>1</sup>データの区切りとして、タブは良く使われる。ファイルの内容ををエディターで見るときに、データの並びがそろっているため、視認性がよくなるからである。



```

-3.0892328e+00 -5.2335956e-02 -9.9862953e-01 5.2407779e-02
-3.0717795e+00 -6.9756474e-02 -9.9756405e-01 6.9926812e-02
-3.0543262e+00 -8.7155743e-02 -9.9619470e-01 8.7488664e-02
-3.0368729e+00 -1.0452846e-01 -9.9452190e-01 1.0510424e-01
-3.0194196e+00 -1.2186934e-01 -9.9254615e-01 1.2278456e-01
-3.0019663e+00 -1.3917310e-01 -9.9026807e-01 1.4054083e-01
-2.9845130e+00 -1.5643447e-01 -9.8768834e-01 1.5838444e-01
-2.9670597e+00 -1.7364818e-01 -9.8480775e-01 1.7632698e-01
-2.9496064e+00 -1.9080900e-01 -9.8162718e-01 1.9438031e-01

```

途中は省略．長いので全て記述すると紙の無駄．

```

3.0892328e+00 5.2335956e-02 -9.9862953e-01 -5.2407779e-02
3.1066861e+00 3.4899497e-02 -9.9939083e-01 -3.4920769e-02
3.1241394e+00 1.7452406e-02 -9.9984770e-01 -1.7455065e-02
3.1415927e+00 1.2246064e-16 -1.0000000e+00 -1.2246064e-16

```

[練習 4] 次の関数の値 (実数部と虚数部) を,  $0 \leq x \leq 2\pi$  を 100 等分してファイルに書き出せ．

$$f(x) = e^{ix} \qquad g(x) = \cos x + i \sin x \qquad (1)$$

## 4.2 ファイル入力の実際

また, 数値計算では大量の数値データをハードディスクから読み込むことも多い．ここでの講義でも, データを読み込むことがある．また, 卒研ではなおさらである．数値計算や実験でのファイルからのデータ入力のコツは, 次のとおりである．

- データ数が多い場合, 読み込んだデータは配列 (あるいは構造体) に格納する．
- ループ文を用いて, `fscanf()` 関数を繰り返し使い, データを読み込む．
- `fscanf()` 関数の戻り値が EOF の場合<sup>2</sup>, データの読み込みを止める．

これらを使った例をリスト 4 に示す．先ほど作成したファイル内容を読み込み, ディスプレイに出力するプログラムをである．

リスト 4: ファイルの内容を読み込むプログラム

```

1 #include <stdio.h>
2
3 int main(void)
4 {
5     FILE *in_file;
6     double x[500], y1[500], y2[500], y3[500]; //ちょっと多めの要素を確保
7     int i, j;
8
9     in_file = fopen("trifunc.txt", "r");
10
11     for(i=0; i<=499; i++){
12         if(EOF == fscanf(in_file, "%lf%lf%lf%lf",&x[i], &y1[i], &y2[i], &y3[i])) break;
13     }
14
15     fclose(in_file);

```

<sup>2</sup>EOF は end of file の略でファイルの終わりを示す．

```

16
17     for(j=0; j<i; j++){
18         printf("%10.7f\t%10.7f\t%10.7f\t%10.7f\n",x[j], y1[j], y2[j], y3[j]);
19     }
20
21     return 0;
22 }

```

### 実行結果

このプログラムはファイル (trifunc.txt) からデータを読み込み、その値をディスプレイに出力する。ディスプレイは以下ようになる。

```

-3.1415927    -0.0000000    -1.0000000    0.0000000
-3.1241394    -0.0174524    -0.9998477    0.0174551
-3.1066861    -0.0348995    -0.9993908    0.0349208
-3.0892328    -0.0523360    -0.9986295    0.0524078
-3.0717795    -0.0697565    -0.9975641    0.0699268
-3.0543262    -0.0871557    -0.9961947    0.0874887
-3.0368729    -0.1045285    -0.9945219    0.1051042
-3.0194196    -0.1218693    -0.9925462    0.1227846
-3.0019663    -0.1391731    -0.9902681    0.1405408
-2.9845130    -0.1564345    -0.9876883    0.1583844
-2.9670597    -0.1736482    -0.9848077    0.1763270
-2.9496064    -0.1908090    -0.9816272    0.1943803

```

途中は省略。長いので全て記述すると紙の無駄。

```

3.0892328    0.0523360    -0.9986295    -0.0524078
3.1066861    0.0348995    -0.9993908    -0.0349208
3.1241394    0.0174524    -0.9998477    -0.0174551
3.1415927    0.0000000    -1.0000000    -0.0000000

```

[練習 5] web ページに載せているファイルには、4 列の大量の整数が書かれている。各列の整数の最大値と最小値、そして平均値を求めよ。

## 5 課題

### 5.1 内容

以下の課題を実施し、レポートとして提出すること。

[問 1] (復)教科書 [1] の第 18 章 (pp.376–405) を 3 回読め。レポートには「3 回読んだ」と書け。

[問 2] (復)本日配布したプリントを 2 回読め。レポートには「2 回読んだ」と書け。さらに、誤字・脱字、表現の悪いところ、間違いを指摘せよ。

[問 3] (復)リスト 1 を参考に、ファイルに以下の文字を書き込むプログラムを作成せよ。

夏休み、まだかなー。

その前に、前期末試験がある。いやだなー。

暑いし、かつたるい授業なんか、受けたくないよなー。

I look forward to a summer vacation.

プログラムを実行した後、作成したファイルを中身を確認せよ。

[問 4] (復)リスト 3 を参考に、 $x$  の値を  $0 \sim 2\pi$  を 360 等分して、以下の計算結果をファイルに書き込め。

$$x \qquad \sin x \qquad x - \frac{x^3}{6} + \frac{x^5}{120} - \frac{x^7}{5040}$$

プログラムを実行した後、作成したファイルを中身を確認せよ。(おまけ)3 つめはどのような関数か?

[問 5] (復)リスト 4 を参考に、前問で作成したファイルを読み込み、値をディスプレイに表示せよ。

[問 6] ここでの学習内容でわからないところがあれば、具体的に記述せよ。

### 5.2 レポート 提出要領

期限 7月11日(水) AM 8:45

用紙 A4のレポート用紙。左上をホッチキスで綴じて、提出のこと。

提出場所 山本研究室の入口のポスト

表紙 表紙を1枚つけて、以下の項目を分かりやすく記述すること。

授業科目名「計算機応用」

課題名「ファイル処理関数」

提出日

5E 学籍番号 氏名

内容 2ページ以降に問いに対する答えを分かりやすく記述すること。

## 付録 A 標準入出力

### 付録 A.1 特別なファイル (標準入力、標準出力、標準エラー出力)

C 言語でファイルを取り扱う場合、(1)FILE 型のポインタの宣言、(2) ファイルのオープン、(3) ファイルの読み書き、(4) ファイルのクローズの処理が必要である。しかし、特別な 3 個のファイル (標準入力、標準出力、標準エラー出力) は、いきなりファイルの読み書きができる。(1) と (2)、(4) の処理が不要なのである。コンソール入力述べたように、通常、標準入力はキーボード、標準出力はディスプレイを示す。C 言語では (UNIX では)、キーボードやディスプレイもファイルとして扱われ、読み書きする。それどころか、すべてのデバイスがファイルとして扱われる。そうすると、シンプルな取り扱いが可能となる。

これら、特別な 3 個のファイルについて、表 1 にまとめる。

表 1: 標準入出力ファイル

ファイル	ファイルポインタ	デバイス (通常)
標準入力	stdin	キーボード
標準出力	stdout	ディスプレイ
標準エラー出力	stderr	ディスプレイ

fscanf() 関数でファイルポインタとして stdin を指定した場合、scanf() と同じ動作をする。同様に、fprintf() 関数で stdout を指定した場合、printf() と同じ動作をする。これを上手に使うと、プログラムのデバッグのときに便利である。

最後に標準エラー出力について述べる。標準エラー出力とは、エラーが発生した場合のメッセージなどを出力先のことを言う。プログラム中で処理にエラーが発生した場合、そのメッセージの出力先に指定する。printf() 関数を使うよりも、fprintf() 関数でファイルポインタとして stderr を指定した方が後々都合が良い。

```
fprintf(stderr, "ファイルの読み込みに失敗しました\n");
```

見た目の動作は printf 関数と同じ動作をする。

しかし、こうするとエラーメッセージのみ、リダイレクトすることができプログラムの保守性が上がる。本当は、教科書 [1]p.395 に書かれている perror() 関数を使うのがもっとも良いだろう。

## 参考文献

- [1] 林春比古. 新訂 C 言語入門 シニア編. ソフトバンク パブリッシング, 2004.