C言語の学習 ファイル処理関数

山本昌志*

2006年6月27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方法を学ぶ．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，大量の数値を扱うことが多い．いちいち紙に計算結果を書き写すことは不可能なので，普通，ハードディスクに保存する．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拡がる．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．ここでは，`fprintf()` でファイルに文字を書くことができる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．

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，ファイルを読み込んで表示するプログラムである．

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．ただ，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せねばならない．

- 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．

3　ファイル処理の方法

3.1　ファイル処理の流れ

先ほどの例でファイル処理の大体の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．

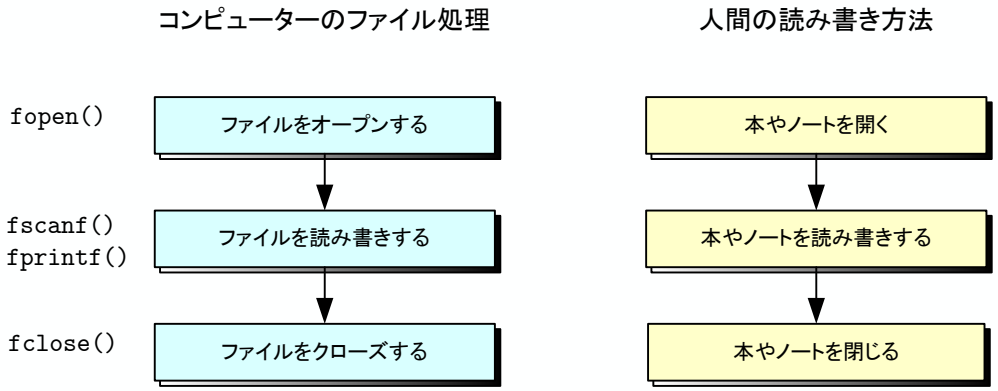

図 1: コンピューターのファイル処理と人間の読み書き方法

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．

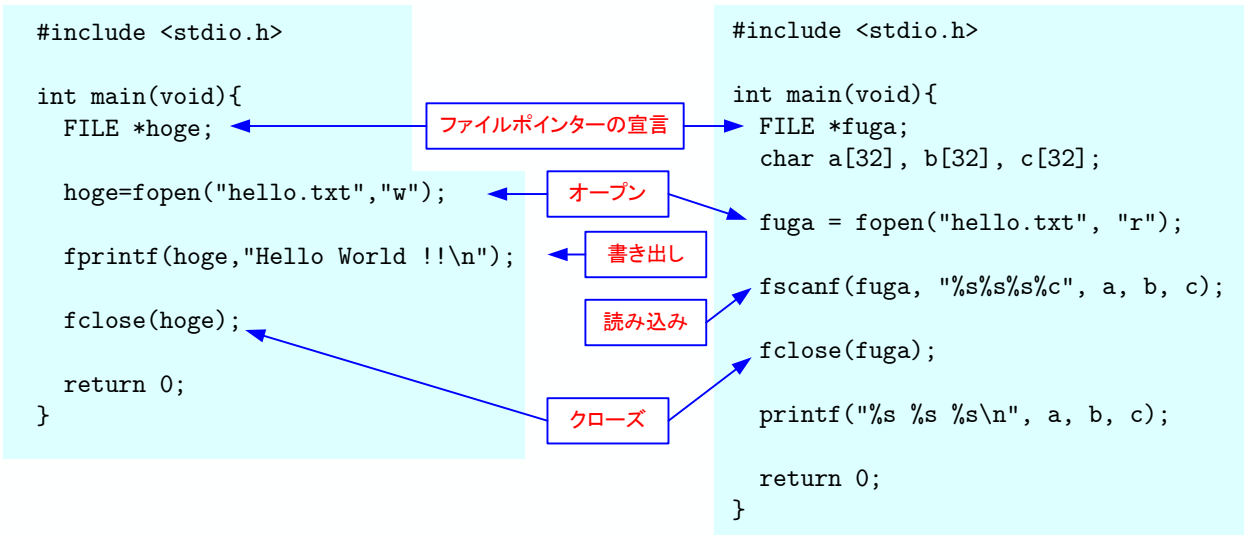

図 2: C 言語でのファイル処理の例

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，ファイルポインターの宣言が必要である．ファイルポインターについては，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．

3.2 ファイルのオープンとクローズ

3.2.1 ファイルポインターの宣言

ファイル処理にはそれに関わる多くの情報が必要である．そのために，ファイルの情報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ーである．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
- バッファー領域の先頭位置
- ファイルの状態
- ファイル番号

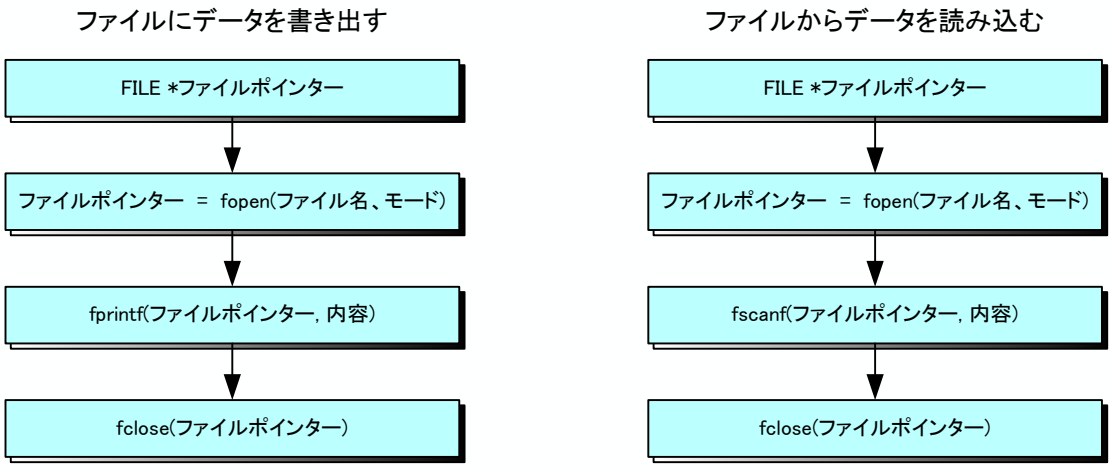

図 3: C 言語でのファイル処理の流れ

このファイルポインター (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 (ポインター) 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 (ポインター)foo を宣言できる．ただし，foo は変数名なのでプログラマーが適当な名前をつける．

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ーに代入する．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 型のポインター，ファイル名を表す第一引数は char 型のポインター，オープンモードを表す第 2 引数は char 型のポインターと言うことである．もし，オープンに失敗すると，NULL という戻り値になる．char 型のポインター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

3.3.1 入出力の方法

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．

まず，入力であるが，一般のファイルと標準入力の場合を並べて書くと

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

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．

出力もまったく同じである．

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

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．

3.4 ファイル出力の実際

計算結果などを大量のデータはハードディスクに保存しなくてはならない．ファイル出力のコツは，

- 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[1]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．

である．

リスト 3 に三角関数の値をファイル出力するプログラムを示す．

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
4  int main(void){
5    FILE *out_file;
6    double x, y1, y2, y3;
7    double dphi;
8    int i, n;
9
10   n = 360;
11
12   dphi = 2*M_PI/n;
13
14   out_file = fopen("trifunc.txt","w");
15
16   for(i=0; i<=n; i++){
17     x=i*dphi-M_PI;
18     y1 = sin(x);
19     y2 = cos(x);
20     y3 = tan(x);
21     fprintf(out_file, "%e\t%e\t%e\t%e\n", x, y1, y2, y3);
22   }
23
24   fclose(out_file);
25
26   return 0;
27 }
```

3.5 ファイル入力の実際

コンピューターを使ったデータ処理では，対象となる大量のデータをハードディスクから読み込むことが多い．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，

- 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．

[1]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．

- 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[2]，データの読み込みを止める.

である.

リスト 4 に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示す.

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    FILE *in_file;
5    double x[500], y1[500], y2[500], y3[500];
6    int i, j;
7
8    in_file = fopen("trifunc.txt","r");
9
10   for(i=0; i<=499; i++){
11     if(
12       EOF == fscanf(in_file, "%lf%lf%lf%lf",&x[i], &y1[i], &y2[i], &y3[i])
13       ) break;
14   }
15
16   fclose(in_file);
17
18   for(j=0; j<i; j++){
19     printf("%f\t%f\t%f\t%f\n",x[j], y1[j], y2[j], y3[j]);
20   }
21
22   return 0;
23 }
```

4 数値計算のファイル入出力

C 言語のファイル入出力は，用途に応じた処理ができるように，様々な機能が用意されている．この辺りについては，すぐに理解できなくても良いが，ファイル処理を多用するプログラムを作成する場合には各自勉強する必要がある．ここでの講義では，以下のようにすれば良いだろう.

- 高水準ファイル処理を使え

  教科書に書いてあるとおり，高水準ファイル処理と低水準ファイル処理がある．通常は前者を使う．前者はバッファーを文字や数値単位，あるいは文字単位で処理する関数が用意されており，容易にプログラムができる．それに対して，後者はバイト単位で処理する．そのため，細かい処理ができる反面，取り扱い—プログラム—が面倒である.

- テキストモードを使え

  改行コードの処理により，テキストモードとバイナリーモードを使い分ける．Linux(UNIX) では C 言語と同じ改行コードを使う．したがって，テキストモードもバイナリーモードも

[2] EOF は end of file の略でファイルの終わりを示す.

同じ結果が得られる．Windows の場合は，テキストモードだと改行コードの変換が行われるので，注意が必要である．この辺の所は教科書を見て欲しい．

- シーケンシャルファイル処理を使え

  ファイルのデータへのアクセス方法には，シーケンシャルファイル処理とランダムファイル処理がある．前者はデータを先頭から順にアクセスし，後者は任意の場所からアクセスできる．本講義ではランダムアクセス処理を使うことはない．

- ファイル出力には，`fprintf()` 関数を使え

  `fprintf()` 関数は，標準出力 (ディスプレイ) と同じ使い方ができる．画面に出力するのと同じイメージでファイルに書き込むことができるので，容易にプログラムが書ける．実際，出力したファイルを emacs のようなエディターで見ると画面出力と同じである．

- ファイル入力には，`fscanf()` 関数を使え

  `fscanf()` 関数は，標準入力 (キーボード) と同じ使い方ができる．キーボードからデータを入力するのと同じイメージでファイルからデータを読み込むことができる．諸君は，その方法になれているので，容易にプログラムが書けるだろう．

5 練習問題

[**練習 1**] 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．

```
もうすぐ，夏休みだ
暑いし，かったるい授業なんか，受けたくないよなー
Everybody, enjoy 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．

[**練習 2**] リスト 3 を参考に，x の値を 0〜2π 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．

[**練習 3**] リスト 4 を参考に，[練習 2] 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．

6　付録

6.1　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 型のポインターの宣言，(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．コンソール入力で述べたように，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	ファイルポインター	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ーとして `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ーとして `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.309 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.