ファイル入出力

次のプログラム upper.c は、指定されたファイルを開いて読み込み、アルファベットの小文字を大文字に変換し、指定されたファイルに書き込む。プログラムの説明は後述するので、まずは続く upper1.c、実行例に目を通してほしい。

```
/* upper.c: translate all alphabet characters to uppercase */

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

int main(int argc, char *argv[])
{
   FILE *fpin,*fpout;
   int c;

   if (argc < 3){
      fprintf(stderr, "Please Specify Input and Output Filenames¥n");
      exit(1);
   }

   fprintf(stderr, "Opening input file, '%s'¥n",argv[1]);

   if ((fpin = fopen( argv[1], "r")) == NULL) {
      fprintf(stderr, "Can't Open File¥n");
      exit(2);
   }

   fprintf(stderr, "Opening output file, '%s'¥n",argv[2]);

   if ((fpout = fopen( argv[2], "w")) == NULL) {
      fprintf(stderr, "Can't Open File¥n");
      fclose(fpin);
      exit(2);
   }

   while ((c = fgetc(fpin)) != EOF){
      if (c >= 'a' && c <= 'z')
         c += 'A' - 'a';
      fputc(c, fpout);
   }

   fclose(fpout);
   fclose(fpin);
}
```

入力ファイル、出力ファイルがそれぞれ一つでよいならば、リダイレクトを使って同じ事が出来る。次の upper1.c はリダイレクトを使って upper.c と同じ機能を実現したものである。

```
/* upper1.c: translate all alphabet characters to uppercase */
#include <stdio.h>

int main(void)
{
   int c;

   while ((c = getchar()) != EOF){
      if (c >= 'a' && c <= 'z')
         c += 'A' - 'a';
      putchar(c);
   }
}
```

実行例

```
ux101$ gcc upper.c
ux101$ ./a.out    upper.c upper.txt
Opening input file, 'upper.c'
Opening output file, 'upper.txt'
ux101$ cat upper.txt
/* UPPER.C: TRANSLATE ALL ALPHABET CHARACTERS TO UPPERCASE */
#INCLUDE <STDIO.H>

・・・　中略　・・・

   FCLOSE(FPIN);
}
ux101$ gcc upper1.c
ux101$ ./a.out < upper1.c > upper1.txt
ux101$ cat upper1.txt
/* UPPER1.C: TRANSLATE ALL ALPHABET CHARACTERS TO UPPERCASE */
#INCLUDE <STDIO.H>

・・・　中略　・・・

   FCLOSE(FPIN);
}
ux101$
```

２つのプログラムが同じ動作をしていることが確認できただろうか。
upper1.c はこれまでに学んできた事柄だけを使って書かれている。キーボードから１文字を読み込む関数 getchar と端末画面に 1 文字を出力する関数 putchar が使用されている。

```
while ((c = getchar()) != EOF){
          ・・・
      putchar(c);
   }
```

getchar 関数により 1 文字が読み込まれ、その文字コードが変数 c に保存される。読み込まれた文字が入力の完了を示す ctrl-d(EOF)で無い場合、while 文により{}の中が繰り返し実

行される。putchar 関数は読み込まれた文字を画面に出力する。

```
    if (c >= 'a' && c <= 'z')
      c += 'A' - 'a';
```

この部分で小文字を大文字に変換している。以前に学習した文字コードの表を思い出してほしい。計算機では１つの文字を１つの数字として表現する。たとえば'a'は 10 進数で 97、'b'は 98、'A'は 65、'B'は 66 である。このように、小文字の文字コードに'A'-'a'、すなわち-32 を足すと小文字を大文字に変換することが出来る。

upper1.c はこのまま実行するとキーボードからの入力を待つ状態になる。実行例にあるように、

```
ux101$ ./a.out < upper1.c > upper1.txt
```

などとして実行すると、キーボードからの入力や、画面への出力の代わりにファイルが用いられるようになる。「< upper1.c」によってキーボードからの入力の代わりにファイル upper1.c が読み込まれ、「> upper1.txt」によって画面への出力の代わりにファイル upper1.txt に書き込まれるようになる。こうした操作は「リダイレクト」と呼ばれる。

一方、upper.c はより複雑になっている。リダイレクトではなく、実行時にファイル名を指定できるようになっている。実行例にあるように、

```
ux101$ ./a.out    upper.c upper.txt
```

などとして実行すると、"upper.c"や"upper.txt"という文字列がプログラムに引き渡される。こうした方法で実行時にプログラムに引数を与えてやることを「コマンドライン引数」と呼ぶ。

与えられたコマンドライン引数は、main 関数の引数として利用することが出来る。コマンドライン引数を使用するときには、main 関数を以下のように定義しておく。

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

このようにすると、argc に引数の数（コマンド名自体が最初の引数となるので、コマンドライン引数の数+1）が、argv にそれぞれの引数文字列へのポインタの配列が渡される。プログラムではまず、argc が 3 以上、すなわちコマンド名以外に２つのファイル名が指定されていることを確認している。

```
  if (argc < 3){
    fprintf(stderr, "Please Specify Input and Output Filenames¥n");
    exit(1);
  }
```

ここで、exit 関数は、プログラムを終了させる関数であり、引数の不足があった場合にはその場で実行が終了する。exit 関数の引数として 1 を与えているが、これはプログラムの実行が失敗した事をコマンドを実行したシェルにつたえるためである。正常終了の場合は 0 を与える。実行が失敗した原因に応じて 0 以外の数字（エラーコード）を割り当てておけ

ば、シェルの側でエラーに対応した処理を行うことが出来る。fprintf 関数については後述する(printf 関数とほとんど同じだと思ってよい)。

```
fprintf(stderr, "Opening input file, '%s'¥n",argv[1]);
```

一つ目のコマンドライン引数を表示している。%s は文字列を表示させるときに使用する。

```
if ((fpin = fopen( argv[1], "r")) == NULL) {
   fprintf(stderr, "Can't Open File¥n");
   exit(2);
}
```

fopen 関数によって、argv[1]に格納されたファイル名でファイルをオープンしている。fopen 関数の 2 番目の引数の”r”はファイルを読みとり専用モードで開くことを意味している。他に”w”書き込み専用モード、”a”追加モードなどがある。ファイルを開くことに成功すると、fopen の返値としてファイルポインタにファイル情報が格納された領域へのアドレスが返される。プログラムの冒頭で、

```
FILE *fpin,*fpout;
```

と宣言されているのがファイルポインタである。プログラムが開始されたときにはこれらのポインタは無効な値を示しているが、fopen 関数の返値を代入することで、開いたファイルの情報にアクセスするためのポインタとして使用することが出来る。ファイルの情報を格納するための領域は OS が用意するため、プログラム中で意識する必要は無い。

fopen 関数はファイルを開けずに失敗することがある。たとえば存在しないファイルを読みとり専用モードで開こうとしたり、書き込みが出来ないディレクトリで書き込み専用モードでファイルを開こうとしたり、ファイル名が無効であった場合などである。このようなときは返値としてヌルポインタが返される。NULL はヌルポインタを表す。プログラムでは if 文の中で fopen 関数を実行することで、fopen 関数の返値を fpin 代入しつつ、代入した値がヌルポインタかどうかを判断している。if 文の中はファイルが開けなかった場合の処理が記述されており、エラーメッセージを表示して、エラーコード 2 でプログラムを終了している。

fprintf 関数は、printf 関数とほとんど同じ機能をもった関数であるが、printf 関数の出力先が常に端末画面であるのに対し、ファイルポインタを与えて出力先を選ぶことができる。個々ではファイルポインタ stderr を出力先として選んでいる。stderr は標準エラー出力と呼ばれる。通常の端末画面は stdout、標準出力と呼ばれる。これらのファイルポインタはプログラムの実行時に最初から準備されているので、ポインタを用意したり fopen 関数によってオープンする必要はない。標準出力と標準エラー出力は通常どちらも端末画面に表示されるが、リダイレクトを使ってそれぞれ別のファイルに保存することも出来る。

```
while ((c = fgetc(fpin)) != EOF){
   if (c >= 'a' && c <= 'z')
      c += 'A' - 'a';
   fputc(c, fpout);
}
```

この部分は、upper1.c とほとんど同じであるが getchar 関数の代わりに fgetc 関数が、putchar 関数の代わりに fputc 関数が使われている。これらの関数も fprintf 関数と同様で、getchar や putchar 関数とほとんど同じ機能を持っているが、その入力先や出力先をファイルポインタにより指定することが出来る。このプログラムでは先ほどオープンした２つのファイルを、ファイルポインタ fpin と fpout によって指定している。

```
fclose(fpout);
fclose(fpin);
```

オープンしたファイルはプログラムの終了前にクローズする必要がある。ファイル入出力を完結させ、OS がファイル情報を格納するために準備したメモリ領域を開放するためであるこのように fclose 関数を使う。

リダイレクトでは入力、出力とも１つしか使用できないが、このようにして、プログラム中でファイルを開くことによって複数のファイルを同時に扱うことが出来る。

課題：指定されたファイルを読み込み、大文字を小文字に変換して出力するプログラムを作れ。
課題：指定されたファイルを読み込み、各アルファベットが出現する回数をカウントするプログラムを作れ。ただし、大文字と小文字は同じ文字として扱え。
課題：２つのファイル名を実行時引数として得て、１つめに指定されたファイルを読み込み、各行に行番号をつけて、２つめに指定されたファイルに書き込むするプログラムを作れ。
課題：２つのファイル名を実行時引数として得て、それぞれのファイルを開き、各行を読み込んで違いがある行だけを表示するプログラムを作れ。１行の読み込みには fgets 関数を、文字列の比較には strncmp 関数を使用せよ。