UNIX 演習 12 回目　　2024 年 7 月 8 日

先週のサンプルプログラム mycat1.c では、標準入出力を使っていたため、フィルタコマンドのように使うことができるが、テキストファイルの内容を入力としたい場合にはリダイレクトなどが必要になる問題があった。mycat1 < input.txt をアーギュメントで指定して mycat1 input.txt の様に実行したい場合についてまず説明する。

mycat1.c　　標準入力の文字列をそのまま標準出力に書き出す　　前回の課題

```
#include <stdio.h>
int main(void)
 {
 char c;                          // 1 文字ずつ読み込む時のキャラクタ文字変数
 while((c=getc(stdin))!=EOF)      // while ループで読む文字がファイルの末尾になるまで
   {                              // 標準入力から一字ずつ読む
   putc(c,stdout);                // 読んだ文字を標準出力に書き出す
   }
 return 0;
 }
```

---

mycat1.c では、標準入出力を使うため、getc(stdin)と書く代わりに getchar()としたり putc(c,stdout)と書く代わりに putchar(c)とした方が簡潔にはなるが、stdin, stdout を明示することで入出力を明示した。今週は引き続き getc, putc を使い、stdin や stdout の代わりに定義するファイルハンドルというもので入出力を標準有出力からファイルに置き換えることを学ぶ。
コンパイルしたプログラムを実行する際に付した、アーギュメントをプログラム内で利用する方法から説明する。
gcc -o mc2 mc2.c としてコンパイルした実行形式ファイルである mc2 を実行する際に、mc2 foo.txt のように mc2 の読み込むファイル名(foo.txt)を実行時に指定できるようなプログラムを記述する。そのために、このプログラムのメイン関数を定義する際に、

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

と記述する。整数値 argc は、コマンド名自身を含む実行時のアーギュメントの数で上記の例では 2 となる（mc2 と foo.txt）それぞれのアーギュメントの値（文字列）は argv（ポインタのポインタで定義されており、文字列の配列になっている）に順に入り、上記の例では argv[0]に mc2 が、argv[1]に foo.txt が入る。3 つ以上のアーギュメントがある場合には以下同様に argv[2],argv[3]と入っていく。この辺りはシェルスクリプトの位置変数$1, $2 に類似するものとなっている。

mc2.c　　一つ目のアーギュメントで指定したファイル名のファイルの中身をそのまま出力する

```
#include <stdio.h>
int main(int argc, char **argv)
 {
 if(argc != 2)   // コマンド名以外のアーギュメントの数が 1 でなければ
  {
  printf("Error: Wrong number of argument¥n");
  return 1;       // 異常終了
  }
 printf("Will open %s¥n",argv[1]);
 ////////////////////////////////////////以下 argv[1]のファイル名のファイルを開いて
 ////////////////////////////////////////内容を読み込み、標準出力へそのまま出力するコードを挿入
 return 0;
 }
```

```
gcc -o mc2 mc2.c
```

としてコンパイルし

```
./mc2  foo.txt
```

とすると

```
Will open foo.txt
```

と、mc2 の実行時に第二アーギュメントとして指定した文字列が、プログラム内で argv[1]という文字列変数として扱えることが確認できた。次に、このアーギュメントで指定するファイル名のファイルを読み取りファイルとして開き、ファイルの内容を getc で 1 文字ずつ読み取ることができる様にする。そのために FILE *infile;としてファイルハンドルという特殊な型の変数をまず宣言し、読み取ったファイル名でファイルを開く。

```
FILE *infile;
infile = fopen(argv[1],"r");
```

これで、アーギュメントで指定したファイルから情報を読み取る準備ができた。あとは mycat1.c と同様に、stdin という標準入力のハンドルを infile というファイルハンドルで置き換えればファイルから 1 文字ずつ読み取る操作が行える。

存在しないファイル名を指定した場合など、fopen ができなかった場合には NULL を返す。その場合にはそこでエラーメッセージを出力してプログラムを停止する。

```
if(infile == NULL)
 {
 printf("Failed to open input file %s¥n",argv[1]);
 return 1;
 }
```

mycat1.c での getc(stdin) を getc(infile) とすれば、入力ファイルから 1 文字ずつ読み込むことができる

課題提出用のファイル名の付け方について

012Name07Job.c

012 は自分の学生番号の下３桁、Name は自分の名前、YamadaTaro などで置き換える
Job はそれぞれの課題の名前として

012Yamada12mc2.c

の様につける。

メールの宛先は knakamura.maebit@gmail.com

メールのタイトルは日本語で自分の名前と Unix の U１２基本（または応用）の数字を入れる　山田 U12 基本

基本課題１．ジョブ名：mc3（必須：７月９日）

mc2.c と mycat1.c を参考にアーギュメントとしてファイル名をとって、mc3　foo.txt と書けば foo.txt の中身が標準出力に出るプログラム mc3 を記述して 012name07mc3.c （012 は学生番号下３桁、name は自分の名前で置き換え）というファイル名で提出しなさい。

応用課題２．ジョブ名：mc4（任意：７月 12 日）

アーギュメントがあればそれを入力ファイルとし、なければ標準入力から読み込んだ内容を標準出力にそのまま出すプログラム mc4 を記述して、012name07mc4.c というファイル名で提出しなさい。ヒント：FILE　*infile;を定義した状態で、infile=stdin; とすれば、infile からの getc で標準入力からの読み込みができる。