

課題 No.1 の解答例

低水準入出力を直接使用するファイルコピープログラムを作成した。

実行例

実行例の内容は、リストの右側に書き込んだコメントを参考に確認すること。

リスト 1: 実行例 (動作テスト)

```
1 $ ./mycp2 <-- コマンド行引数がない場合
2 Usage: ./mycp2 <srcfile> <dstfile>
3 $ ./mycp2 a.txt <-- コマンド行引数が不足の場合
4 Usage: ./mycp2 <srcfile> <dstfile>
5 $ ./mycp2 a.txt b.txt c.txt <-- コマンド行引数が過剰な場合
6 Usage: ./mycp2 <srcfile> <dstfile>
7 $ ./mycp2 z.txt a.txt <-- コピー元が存在しない場合
8 z.txt: No such file or directory
9 $ ./mycp2 a.txt /a.txt <-- コピー先が書き込み禁止の場合
10 /a.txt: Permission denied
11 $ echo aaa bbb > a.txt <-- a.txt を作って
12 $ ./mycp a.txt b.txt <-- b.txt にコピーしてみる
13 $ cat b.txt <-- b.txt の内容を確認
14 aaa bbb
15 $ echo ccc ddd > c.txt <-- c.txt を作って
16 $ ./mycp c.txt b.txt <-- b.txt に上書きしてみる
17 $ cmp c.txt b.txt <-- b.txt の内容を確認
18 $ dd if=/dev/random of=srcfile bs=1024 count=10 <-- 10KiB の長いファイルを作る
19 10+0 records in
20 10+0 records out
21 10240 bytes transferred in 0.001695 secs (6041591 bytes/sec)
22 $ rm destfile
23 rm: destfile: No such file or directory <-- destfile が存在しない場合
24 $ ./mycp2 srcfile destfile
25 $ cmp srcfile destfile <-- 正しくコピーできている
26 $ dd if=/dev/random of=srcfile bs=1023 count=10 <-- 10KiB より少し短いファイル
27 10+0 records in
28 10+0 records out
29 10230 bytes transferred in 0.003218 secs (3179057 bytes/sec)
30 $ ./mycp2 srcfile destfile <-- destfile が短くなる場合
31 $ cmp srcfile destfile <-- 正しくコピーできている
```

ソースプログラム

7 行 バッファサイズを#define で定義している。

9 行 メッセージを表示し終了する処理が複数必要なので、err_exit() 関数にまとめた。

20 行 コマンド行引数の数を確認している。確認しないと後の処理で誤動作を引き起こす。

26 行 読み込み用のオープンには O_RDONLY フラグを用いる。

30行 書き込み用のオープンには `O_WRONLY`, `O_CREATE`, `O_TRUNC` フラグを用いるのが適切である。ファイルが短くなる場合に `O_TRUNC` が必要である。ファイルを新規作成する場合は保護モードも元ファイルと同じにするべきだが、これまでの学習内容だけではできないので `rw-r--r--` に固定とした。

33行 読み込んだデータのバイト数だけ書き込むように `len` を用いる。

リスト 2: 低水準 I/O 版の `mycp(mycp2.c)`

```
1 #include <stdio.h>           // perror のため
2 #include <stdlib.h>          // exit のため
3 #include <fcntl.h>           // open のため
4 #include <unistd.h>          // read, write, close のため
5
6 // #define BSIZ 1             // !!バッファサイズ:変化させ性能を調べる!!
7 #define BSIZ 1024           // !!バッファサイズ:変化させ性能を調べる!!
8
9 void err_exit(char *s) {      // システムコールでエラー発生時に使用
10     perror( s );              // エラーメッセージを出力して
11     exit(1);                  // エラー終了
12 }
13
14 int main(int argc, char *argv[]) {
15     int fd1, fd2;              // ファイルディスクリプタ
16     ssize_t len;               // 実際に読んだバイト数
17     char buf[BSIZ];           // バッファ
18
19     // ユーザの使い方エラーのチェック
20     if (argc!=3) {
21         fprintf(stderr, "Usage: %s <srcfile> <dstfile>\n", argv[0]);
22         exit(1);
23     }
24
25     // 読み込み用にファイルオープン
26     fd1 = open(argv[1], O_RDONLY);
27     if (fd1<0) err_exit( argv[1] ); // オープンエラーのチェック
28
29     // 書き込み用にファイルオープン
30     fd2 = open(argv[2], O_WRONLY|O_CREAT|O_TRUNC, 0644);
31     if (fd2<0) err_exit( argv[2] ); // オープンエラーのチェック
32
33     // ファイルの書き写し
34     while ((len=read(fd1, buf, BSIZ))>0) {
35         write(fd2, buf, len);
36     }
37
38     close(fd1);
39     close(fd2);
40     return 0;                  // 正常終了
41 }
```