1. ソースコードと実行例

リスト 1: 低水準 I/O 版の mycp(mycp2.c)

```
// perror のため
#include <stdio.h>
#include <stdlib.h>
                               // exit のため
#include <fcntl.h>
                              // open のため
#include <unistd.h>
                               // read,write,close のため
                               // !!バッファサイズ:変化させ性能を調べる!!
#define BSIZ 1
                                // !!バッファサイズ:変化させ性能を調べる!!
//#define BSIZ 1024
                             // システムコールでエラー発生時に使用
void err_exit(char *s) {
perror( s );
 exit(1);
int main(int argc, char *argv[]) {
                               // ファイルディスクリプタ */
 int fd1, fd2;
                               // 実際に読んだバイト数
 int len;
 char buf[BSIZ];
                               // バッファ
 // ユーザの使い方エラーのチェック
 if (argc!=3) {
   fprintf(stderr, "Usage_{\sqcup}:_{\sqcup}\%s_{\sqcup} < srcfile>_{\sqcup} < dstfile> \\ ", argv[0]);
   exit(1);
 }
 // 読み込み用にファイルオープン
 fd1 = open(argv[1], O_RDONLY);
 if (fd1<0) err_exit( argv[1] ); // オープンエラーのチェック
 // 書き込み用にファイルオープン
 fd2 = open(argv[2], O_WRONLY|O_CREAT|O_TRUNC,0644);
 if (fd2<0) err_exit( argv[2] ); // オープンエラーのチェック
 // ファイルの書き写し
 while ((len=read(fd1, buf,BSIZ))>0) {
   write(fd2,buf,len);
 close(fd1);
 close(fd2);
 exit(0);
```

リスト 2: 実行例 (動作テスト!!)

```
<-- コマンド行引数がない場合
$ mycp2
Usage: mycp2 <srcfile> <dstfile>
$ mycp2 a.txt
                                      <-- コマンド行引数が一つしかない場合
Usage: mycp2 <srcfile> <dstfile>
                                      <-- コピー元が存在しない場合
$ mycp2 z.txt a.txt
z.txt: No such file or directory
                                      <-- コピー先が書き込み禁止の場合
$ mycp2 a.txt /a.txt
/a.txt: Permission denied
$ echo aaa bbb > a.txt
                                     <-- a.txt を作って
                                     <-- b.txt にコピーしてみる
$ mycp a.txt b.txt
                                     <-- b.txt の内容を確認
$ cat b.txt
aaa bbb
$ echo ccc ddd > c.txt
                                     <-- c.txt を作って
$ mycp c.txt b.txt
                                     <-- b.txt に上書きしてみる
                                     <-- b.txt の内容を確認
$ cat b.txt
ccc ddd
```

2. 性能測定の手順を確認する。

「1MB(メガバイト (2²⁰ バイト))のファイルをコピーする時間」を測定したい。以下のように dd コマンドでファイルを作成して time コマンドで時間を測定しようと思う。うまくできるか確認する。

リスト 3: 性能測定手順

```
$ dd if=/dev/random of=aaa bs=1024 count=1024
1024+0 records in
1024+0 records out
1048576 bytes transferred in 0.076589 secs (13690929 bytes/sec)
$ ls -l aaa
-rw-r--r-- 1 sigemura staff 1048576 Apr 23 11:32 aaa
$ time mycp2 aaa bbb
real 0m0.012s
user 0m0.001s
sys 0m0.008s
[sigemura@02_低水準 IO]$ ls -l aaa bbb
-rw-r--r-- 1 sigemura staff 1048576 Apr 23 11:32 aaa
-rw-r--r-- 1 sigemura staff 1048576 Apr 23 11:33 bbb
```

3. 測定の結果

前の方法で 1MB のファイルをコピーする時間を測定してみる。mycp (高水準)、mycp2_1024 (低水準で BSIZ=1024)、mycp2_1 (低水準で BSIZ=1) について何度か測定して平均を求めた。

туср						
	1回目	2回目	3回目	4回目	5回目	平均
real	0.104	0.090	0.086	0.088	0.093	0.092
user	0.080	0.082	0.079	0.081	0.084	0.081
sys	0.005	0.004	0.004	0.004	0.005	0.004

mycp2_1024						
	1回目	2回目	3回目	4回目	5回目	平均
real	0.011	0.008	0.008	0.010	0.009	0.009
user	0.001	0.001	0.001	0.001	0.001	0.001
sys	0.007	0.005	0.005	0.005	0.005	0.005

mycp2_1						
	1回目	2回目	3回目	4回目	5回目	平均
real	1.772	1.786	1.742	1.768	1.830	1.780
user	0.172	0.174	0.172	0.179	0.176	0.175
sys	1.593	1.606	1.564	1.583	1.646	1.598

低水準で BSIZ=1 が遅いことが分かる。しかし、実行時間が短すぎて誤差の割合が高そうだ。

4. 再測定の結果

10MB のファイルをコピーする時間を測定した。

туср						
	1回目	2回目	3回目	4回目	5回目	平均
real	0.828	0.829	0.842	0.848	0.822	0.834
user	0.793	0.787	0.810	0.814	0.789	0.799
sys	0.022	0.024	0.019	0.019	0.018	0.020

	mycp2_1						
	1回目	2回目	3回目	4回目	5回目	平均	
real	18.021	17.812	17.709	17.744	17.679	17.793	
user	1.707	1.674	1.695	1.723	1.692	1.698	
sys	16.253	16.096	15.977	15.976	15.935	16.047	

mycp2_1024 1回目 2回目 3回目 4回目 5回目 平均 real 0.051 0.052 0.042 0.042 0.043 0.046 0.003 0.003 0.003 0.003 0.003 0.003 0.037 0.028 0.027 0.028 0.030 0.032

ファイルサイズを 1 0 倍にしても傾向は同じ ようだ。

1番:低水準でBSIZ=1024

2番:高水準

3番:低水準で BSIZ=1