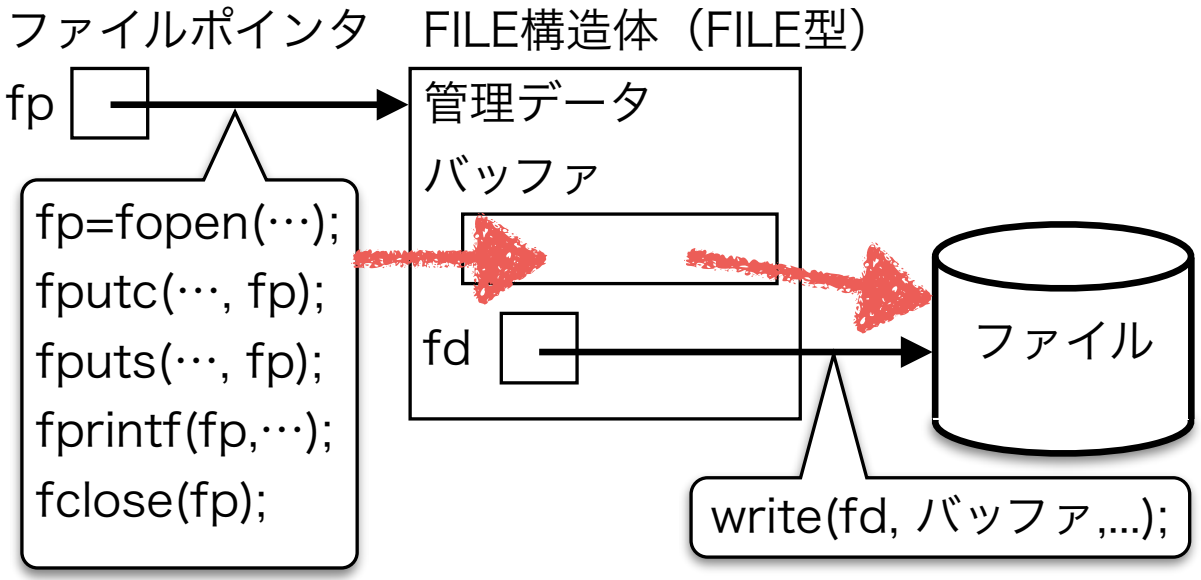
オペレーティングシステムの機能を使ってみよう
第３章 高水準入出力と低水準入出力

高水準入出力と低水準入出力

ファイルを読み書きするための機能
(API：Application Proguram Interface)

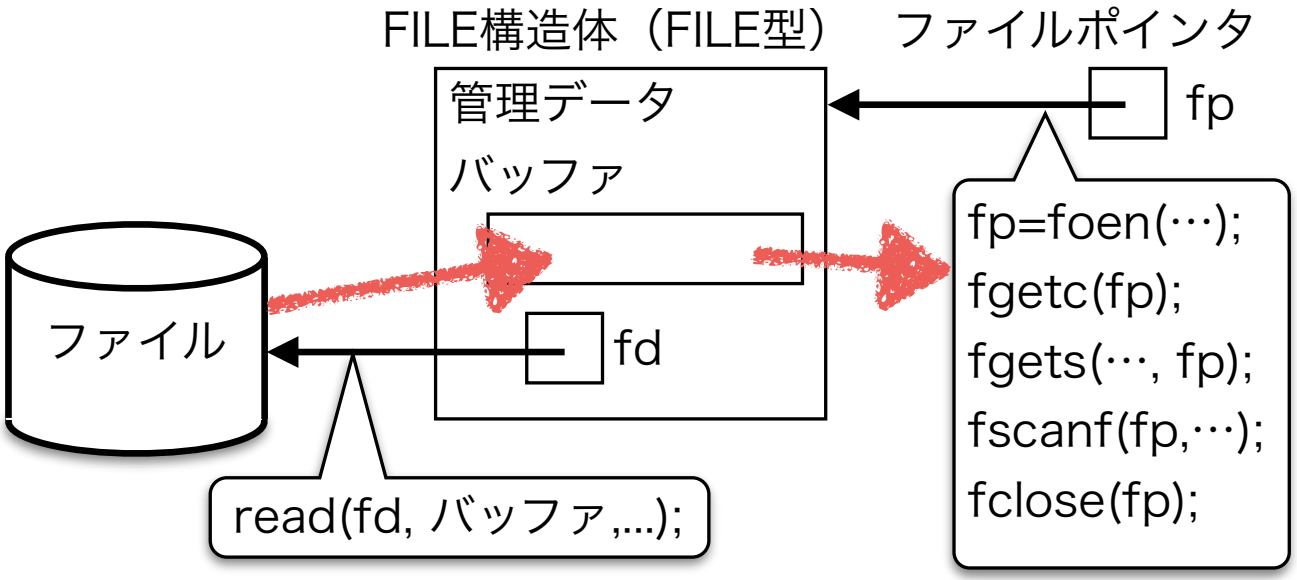
- **高水準入出力（高水準 *I/O*）**
  多くの高機能な関数群
  (`printf()`, `scanf()`, `putchar()`, `getchar()`, ...)
- **低水準出力（高水準 *I/O*）**
  システムコールのこと
  少なく，かつ，シンプルな API
  (`open()`, `read()`, `write()`, `lseek()`, `close()`)

高水準I/Oのデータ構造（書き込み）



- ファイルポインタ（fp）
- FILE構造体
- バッファリング
- writeシステムコール

高水準I/Oのデータ構造（読み出し）



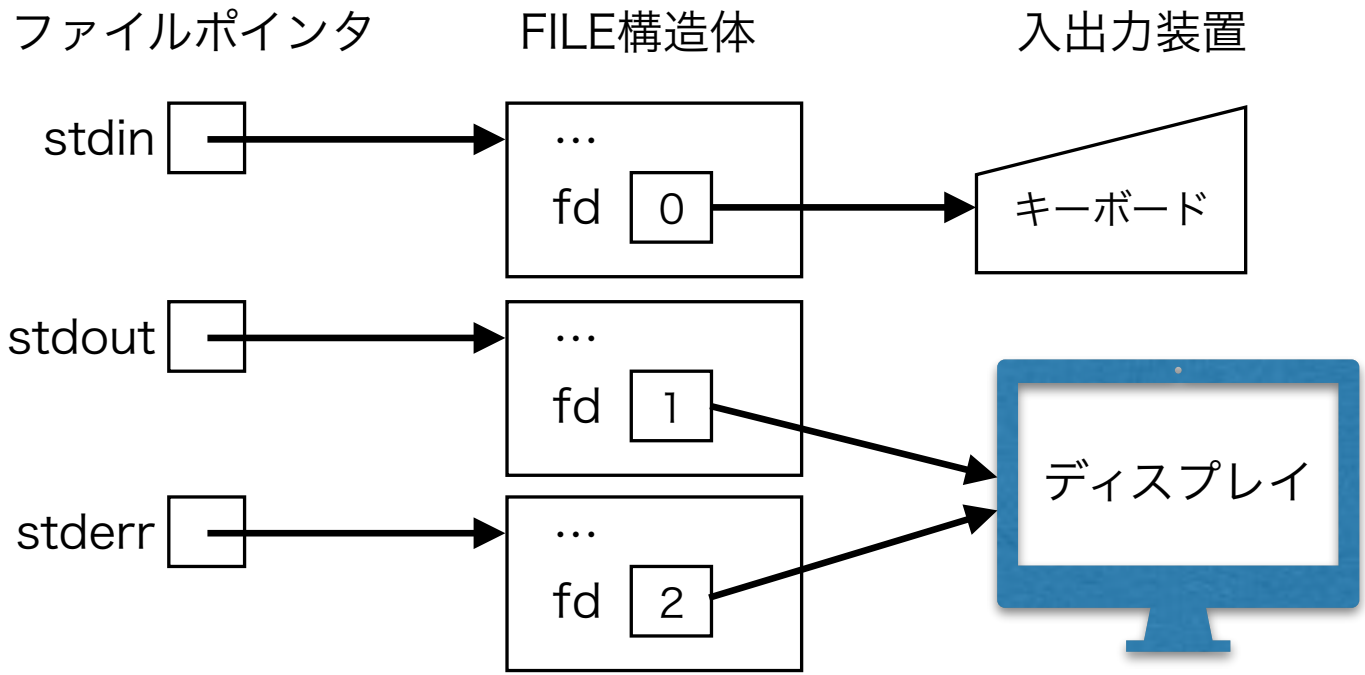
- ファイルポインタ（fp）
- FILE構造体
- readシステムコール
- バッファリング

標準入出力（標準入出力ストリーム）

名称	*fd*	*fp*	通常の接続先
標準入力ストリーム	0	stdin	キーボード
標準出力ストリーム	1	stdout	ディスプレイ
標準エラー出力ストリーム	2	stderr	ディスプレイ

fd : ファイルディスクリプタ

fp : ファイルポインタ

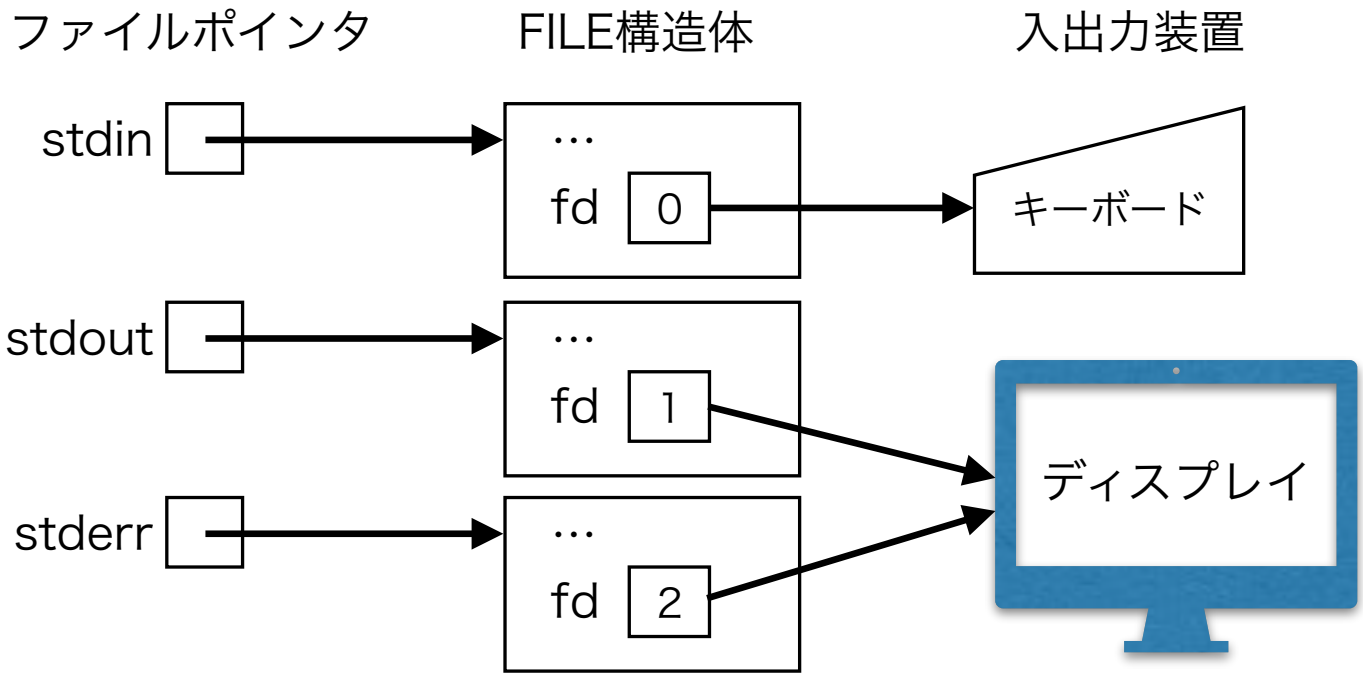


ユニファイド I/O

標準ストリーム	同じ意味の呼出し	役割り
`scanf(...)`	`fscanf(stdin, ...)`	書式付きの入力
`getchar()`	`fgetc(stdin)`	1 文字入力
-	`fgets(stdin, ...)`	1 行入力
`printf(...)`	`fprintf(fp, ...)`	書式付きの出力
`putchar(c)`	`fputc(c, stdout)`	1 文字出力
`puts(buf)`	`fputs(buf, stdout)`	1 行出力

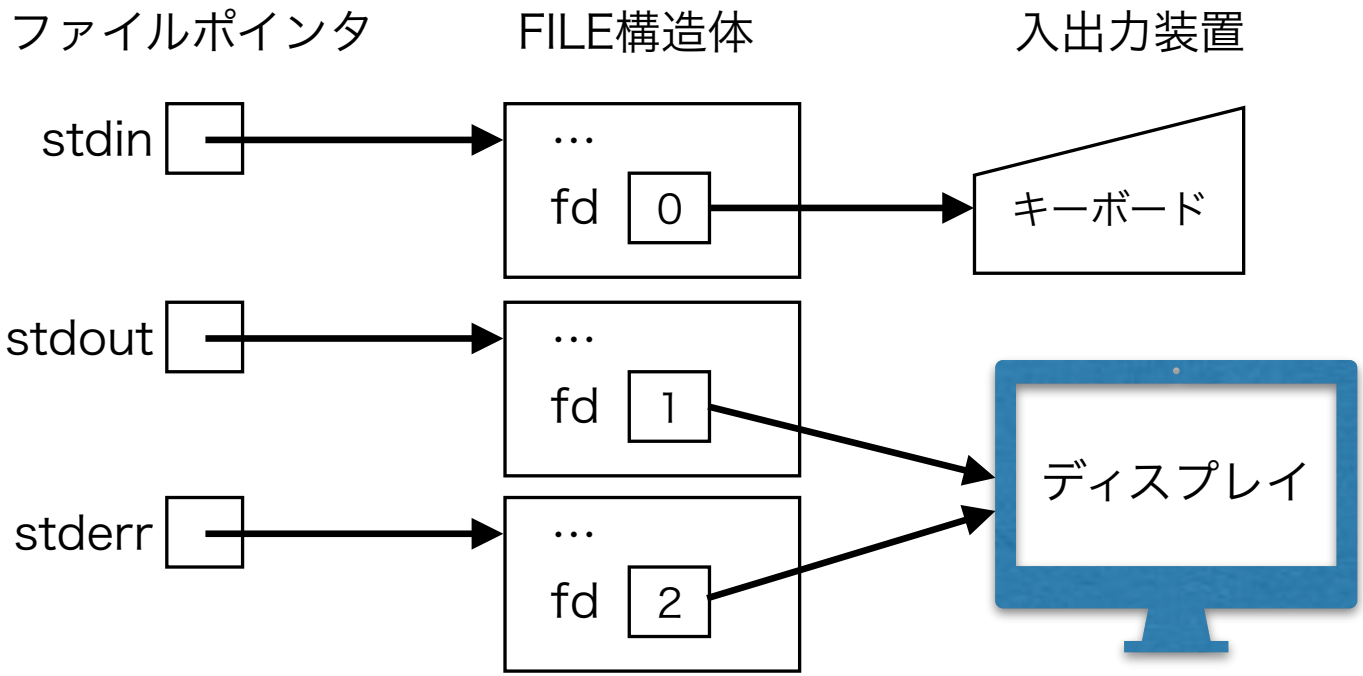
- `printf(...)` と `fprintf(stdout,...)` は同じ
- `fp` の代わりに `stdin`, `stdout` 等が使用できる.
- キーボードやディスプレイ（**入出力装置**）と**ファイル**を同じ要領で操作できる.
- 入出力装置をファイルに統合＝（**ユニファイド** *I/O*）

標準入力ストリーム



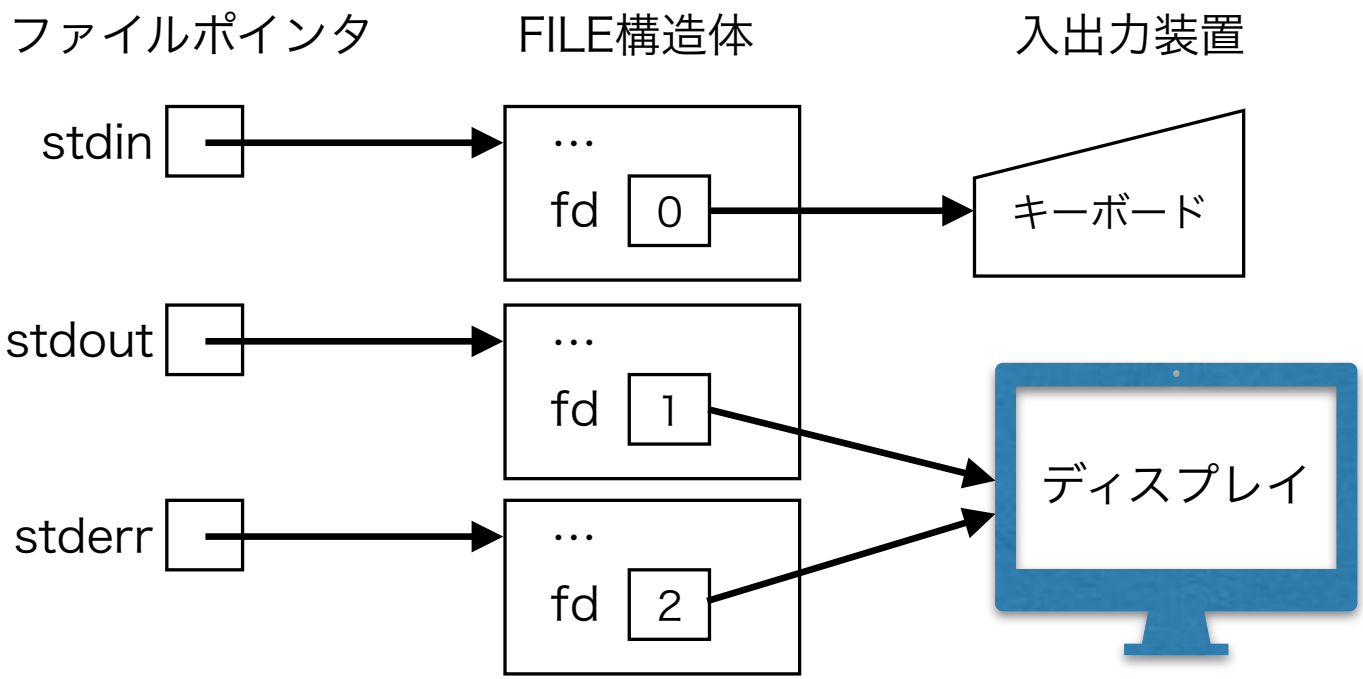
- ファイルポインタは `stdin`
- ファイルディスクリプタは 0 番
- ファイルディスクリプタ 0 は通常キーボードに接続
- ファイルポインタと FILE 構造体はプログラム起動時に初期化
- シェルはファイルディスクリプタ 0 をリダイレクト可能

標準出力ストリーム



- ファイルポインタは `stdout`
- ファイルディスクリプタは 1 番
- ファイルディスクリプタ 1 は通常ディスプレイに接続
- ファイルポインタと FILE 構造体はプログラム起動時に初期化
- シェルはファイルディスクリプタ 1 をリダイレクト可能

標準エラー出力ストリーム



- エーラメッセージ出力用のストリーム
- ファイルポインタは `stderr`
- ファイルディスクリプタは 2 番
- ファイルディスクリプタ 2 は通常ディスプレイに接続
- ファイルポインタと FILE 構造体はプログラム起動時に初期化
- シェルはファイルディスクリプタ 2 をリダイレクト可能

性能比較（1/2）

1. プログラムを準備する
   - mycp　：　高水準 I/O 版
   - mycp2_1　：　低水準 I/O 版（バッファサイズ＝ 1 バイト）
   - mycp2_1024　：　低水準 I/O 版（バッファサイズ＝ 1,024 バイト）
2. 大きめのファイルを作る

```
$ dd if=/dev/random of=aaa bs=1024 count=10240  <-- 10MiB のファイル aaa を作る
10240+0 records in
10240+0 records out
10485760 bytes transferred in 1.019062 secs (10289621 bytes/sec)
]$ ls -l aaa
-rw-r--r--  1 sigemura  staff  10485760 Apr 15 17:35 aaa  <-- できている
```

性能比較 (2/2)

3 実行時間を測定方法

```
$ rm bbb                    <--- 念のため bbb を消す
rm: bbb: No such file or directory
$ time mycp2_1 aaa bbb
real    1m31.664s
user    0m11.653s
sys     1m16.554s
$ cmp aaa bbb          <--- コピー結果が正常かチェック
$
```

4 実行時間の測定

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

課題 No.2：三つのプログラムの性能比較

上記の性能比較を実際に行う．提出物は以下の通りとする．

1. 三つのプログラムについて実行結果を整理したもの
2. 使用したプログラムのソースコード
3. 感想・考察（ソースコードの余白に記入する）

課題 No.2 の解答例（1/5）低水準I/O(1/2)

```
#include <stdio.h>                      // perror のため
#include <stdlib.h>                     // exit のため
#include <fcntl.h>                      // open のため
#include <unistd.h>                     // read,write,close のため

//#define  BSIZ 1                       // !!バッファサイズ:変化させ性能を調べる!!
#define  BSIZ 1024                      // !!バッファサイズ:変化させ性能を調べる!!

void err_exit(char *s) {                // システムコールでエラー発生時に使用
  perror( s );                          //  エラーメッセージを出力して
  exit(1);                              //  エラー終了
}

int main(int argc, char *argv[]) {
  int fd1, fd2;                         // ファイルディスクリプタ
  ssize_t len;                          // 実際に読んだバイト数
  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 : %s <srcfile> <dstfile>\n", argv[0]);
    exit(1);
  }
```

課題 No.2 の解答例（2/5）低水準 I/O(2/2)

```
  // 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);
  return 0;                        // 正常終了
}
```

課題 No.2 の解答例（3/5）高水準 I/O(1/2)

```
#include <stdio.h>                                          // 入出力のために必要
#include <stdlib.h>                                         // exit のために必要

// err_exit : ファイルのオープンに失敗したときエラーメッセージを表示し終了
void err_exit(char *prog, char *fname) {
    fprintf(stderr,                                         // 標準エラー出力に
            "%s : can't open %s\n",                         // エラーメッセージを表示し
            prog, fname);
    exit(1);                                                // エラー終了
}

int main(int argc, char *argv[]) {
  FILE *fps;                                                // コピー元ファイル用
  FILE *fpd;                                                // コピー先ファイル用
  int  ch;                                                  // コピー時使用

  if (argc != 3) {                                          // 引数の個数が予定と異なる
    fprintf(stderr,                                         // 標準エラー出力に
            "Usage: %s <srcfile> <dstfile>\n",              // 使用方法を表示して
            argv[0]);
    exit(1);                                                // エラー終了
  }
```

課題 No.2 の解答例（4/5）高水準 I/O(2/2)

```
  if ((fps = fopen(argv[1], "rb"))==NULL)      // コピー元のオープン失敗
    err_exit(argv[0], argv[1]);

  if ((fpd = fopen(argv[2], "wb"))==NULL)      // コピー元のオープン失敗
    err_exit(argv[0], argv[2]);

  while((ch=getc(fps)) != EOF) {               // EOF になるまで
    putc(ch ,fpd);                             // 1 バイト毎のコピー
  }

  fclose(fps);                                 // ファイルクローズ
  fclose(fpd);

  return 0;                                    // 正常終了
}
```

課題 No.2 の解答例（5/5）実行時間のまとめ

1 バイトの write システムコール使用の場合

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

1,024 バイトの write システムコール使用の場合

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
user	0.003	0.003	0.003	0.003	0.003	0.003
sys	0.032	0.037	0.028	0.027	0.028	0.030

高水準 I/O 使用の場合

mycp						
	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