課題 No.4 の解答例

1. 簡易版の UNIX コマンドを作成しなさい.

リスト 1: ファイルの削除（myrm.c）

```
// myrm : 簡易 rm コマンド
#include <stdio.h>     // perror のため
#include <unistd.h>    // unlink のため

int main(int argc, char *argv[]) {
  // (1) 使用方法を間違っていないかチェックする
  if (argc<2) {
    fprintf(stderr,"使用方法:␣%s␣file␣...\n", argv[0]);
    return 1;
  }

  // (2) ファイルの削除(リンクの削除を実行する)
  int err = 0;          // エラーが発生したか
  for (int i=1; i<argc; i++) {
    if (unlink(argv[i])<0) {
      perror(argv[i]);
      err = 1;          // エラーが発生したことを記録して処理は続行する
    }
  }

  return err;
}
```

リスト 2: myrm の実行例（動作テスト！！）

```
$ echo aaa  > a.txt                                    <-- 実験用ファイルを作る
$ echo aaa  > b.txt
$ echo aaa  > c.txt
$ ls -l *.txt
-rw-r--r--  1 sigemura  staff  4 May 18 11:19 a.txt <-- できているか確認
-rw-r--r--  1 sigemura  staff  4 May 18 11:19 b.txt
-rw-r--r--  1 sigemura  staff  4 May 18 11:19 c.txt
$ ./myrm                                               <-- 引数が 0 個の場合
使用方法: ./myrm file ...
$ ./myrm /bin/rm                                       <-- システムのファイルは
/bin/rm: Permission denied                                  消せない (権限無し)
$ ./myrm xyz                                           <-- 存在しないファイル
xyz: No such file or directory
$ ./myrm a.txt                                         <-- ファイルを 1 つ消す
$ ls -l *.txt
-rw-r--r--  1 sigemura  staff  4 May 18 11:19 b.txt <-- a.txt は消えた
-rw-r--r--  1 sigemura  staff  4 May 18 11:19 c.txt
$ ./myrm b.txt c.txt                                   <-- まとめて 2 つ消す
$ ls -l *.txt
ls: *.txt: No such file or directory                   <-- 両方消えた
$
```

リスト 3: ディレクトリの作成 (mymkdir.c)

```
//
// mymkdir : 簡易 mkdir コマンド
//
#include <stdio.h>     // perror のため
#include <sys/types.h> // mkdir のため
#include <sys/stat.h>  // mkdir のため

int main(int argc, char *argv[]) {
  // (1) 使用方法を間違っていないかチェックする
  if (argc<2) {
    fprintf(stderr,"使用方法:␣%s␣directory␣...\n", argv[0]);
    return 1;
  }

  // (2) ディレクトリを作成する
  int err = 0;          // エラーが発生したか
  for (int i=1; i<argc; i++) {
    if (mkdir(argv[i],0755)<0) {
      perror(argv[i]);
      err = 1;
    }
  }

  return err;
}
```

リスト 4: mymkdir の実行例（動作テスト！！）

```
$ ./mymkdir                                          <-- 引数がない場合
使用方法: ./mymkdir directory ...
$ ./mymkdir a                                        <-- 引数が 1 つの場合
$ ./mymkdir b c d                                    <-- 引数が 3 つの場合
$ ls -l a b c d
drwxr-xr-x  2 sigemura  kan   512 Jan 20 10:04 a
drwxr-xr-x  2 sigemura  kan   512 Jan 20 10:04 b
drwxr-xr-x  2 sigemura  kan   512 Jan 20 10:04 c
drwxr-xr-x  2 sigemura  kan   512 Jan 20 10:04 d
$ ./mymkdir a                                        <-- ディレクトリ名が衝突
a: File exists
$ ./mymkdir /a                                       <-- / には作れない
/a: Permission denied
$ ./mymkdir e a f                                    <-- 複数の 1 つがエラー
a: File exists
$ ls -l a b c d e f
drwxr-xr-x  2 sigemura  kan   512 Jan 20 10:04 a
drwxr-xr-x  2 sigemura  kan   512 Jan 20 10:04 b
drwxr-xr-x  2 sigemura  kan   512 Jan 20 10:04 c
drwxr-xr-x  2 sigemura  kan   512 Jan 20 10:04 d
drwxr-xr-x  2 sigemura  kan   512 Jan 20 10:05 e    <-- エラーにならなかった
drwxr-xr-x  2 sigemura  kan   512 Jan 20 10:05 f      ディレクトリはできてる
$
```

リスト 5: ディレクトリの削除 (myrmdir.c)

```
//
// myrmdir : 簡易 rmdir コマンド
//
#include <stdio.h>      // perror のため
#include <unistd.h>     // rmdir のため

int main(int argc, char *argv[]) {
  // (1) 使用方法を間違っていないかチェックする
  if (argc<2) {
    fprintf(stderr,"使用方法:␣%s␣directory␣...\n", argv[0]);
    return 1;
  }

  // (2) ディレクトリを削除する
  int err = 0;          // エラーが発生したか
  for (int i=1; i<argc; i++) {
    if (rmdir(argv[i])<0) {
      perror(argv[i]);
      err = 1;
    }
  }

  return err;
}
```

リスト 6: myrmdir の実行例（動作テスト！！）

```
$ ./myrmdir                                          <-- 引数がない場合
使用方法: ./myrmdir directory ...
$ ./myrmdir a                                        <-- 引数が１つの場合
$ ./myrmdir b c d                                    <-- 引数が３つの場合
$ ./myrmdir a                                        <-- ディレクトリが存在しない
a: No such file or directory
$ ./myrmdir myrmdir.c                                <-- ディレクトリではない
myrmdir.c: Not a directory
$ ./myrmdir /tmp                                     <-- システムのディレクトリ
/tmp: Permission denied
$ ls -l                                              <-- 結果の確認
total 38
...
drwxr-xr-x  2 sigemura  kan   512 Jan 20 10:05 e
drwxr-xr-x  2 sigemura  kan   512 Jan 20 10:05 f
-rwxr-xr-x  1 sigemura  kan  1234 Jan 20 10:05 myrmdir.c
...
$ ./myrmdir ?                                        <-- ワイルドカード
$ ls -l                                              <-- 結果の確認
...                                                  <-- e,f が消えた
-rwxr-xr-x  1 sigemura  kan  1234 Jan 20 10:05 myrmdir.c
...
$
```

リスト 7: ハードリンクの作成 (myln.c)

```
//
// myln : 簡易 ln コマンド
//
#include <stdio.h>            // perror のため
#include <unistd.h>           // link のため

int main(int argc, char *argv[]) {
  // (1) コマンド行引数の数を確認する
  if (argc!=3) {
    fprintf(stderr, "使用方法：␣%s␣file1␣file2\n", argv[0]);
    return 1;
  }

  // (2) リンクを作成する
  if (link(argv[1], argv[2])<0) {
    perror("link");           // どっちの引数が原因か不明なので。。。
    return 1;
  }

  return 0;
}
```

リスト 8: myln の実行例（動作テスト！！）

```
$ ./myln                                              <- 引数の数が 0 個
使用方法： ./myln file1 file2
$ ./myln a                                            <- 引数の数が 1 個
使用方法： ./myln file1 file2
$ ./myln a b c                                        <- 引数の数が 3 個
使用方法： ./myln file1 file2
$ echo aaaa > a                                       <- ファイル a を作る
$ ls -l                                               <- a ができたか確認
total 36
-rw-r--r--  1 sigemura  kan     5 May 10 14:12 a
$ ./myln a b                                          <- a を b へハードリンク
$ ls -l                                               <- b ができたか確認
total 40
-rw-r--r--  2 sigemura  kan     5 May 10 14:12 a
-rw-r--r--  2 sigemura  kan     5 May 10 14:12 b
$ ./myln a b
link: File exists                                     <- 既にある名前を使うと
$ ./myln x y
link: No such file or directory                       <- 存在しない名前を使うと
$
```

リスト 9: シンボリックリンクの作成 (mylns.c)

```
//
// mylns : 簡易 ln -s コマンド
//
#include <stdio.h>            // perror のため
#include <unistd.h>           // symlink のため
```

```
int main(int argc, char *argv[]) {
  // (1) コマンド行引数の数を確認する
  if (argc!=3) {
    fprintf(stderr, "使用方法：␣%s␣file1␣file2\n", argv[0]);
    return 1;
  }

  // (2) シンボリックリンクを作る
  if (symlink(argv[1], argv[2])<0) {
    perror(argv[2]);          // エラー原因は argv[2] だろう
    return 1;
  }

  return 0;
}
```

リスト 10: mylns の実行例（動作テスト！！）

```
$ ./mylns
使用方法： ./mylns file1 file2
$ ./mylns b.txt
使用方法： ./mylns file1 file2
$ ./mylns b.txt a.txt
$ echo abc > b.txt
$ ls -l
total 10
lrwxr-xr-x  1 sigemura  staff   5 Apr 28 17:03 a.txt -> b.txt
-rw-r--r--  1 sigemura  staff   4 Apr 28 17:05 b.txt
$ cat a.txt
abc
$ ./mylns b.txt a.txt                          <- a.txt はすでに存在
a.txt: File exists
$ ./mylns zzz.txt c.txt                        <- リンク先は存在しなくてもよい
$
```

リスト 11: ファイルの移動（名前変更）(mymv.c)

```
//
// mymv : 簡易 mv コマンド (ファイルを移動するプログラム)
//
#include <stdio.h>  // perror と rename のため

int main(int argc, char *argv[]) {
  // (1) 使い方が正しいか確認する
  if (argc!=3) {
    fprintf(stderr,"使用方法:␣%s␣<もとの名前>␣<新しい名前>\n", argv[0]);
    return 1;
  }

  // (2) ファイルの移動を行う
  if (rename(argv[1], argv[2])<0) {
    perror(argv[0]);                  // エラーメッセージはプログラム名で妥協
```

```
    return 1;
  }

  return 0;
}
```

リスト 12: mymv の実行例（動作テスト！！）

```
$ ./mymv
使用方法: ./mymv <もとの名前> <新しい名前>
$ ls -l *.txt
-rw-r--r--  1 sigemura  staff  1536 Feb  9 11:06 s.txt
$ ./mymv s.txt x.txt                <-- x.txt に変更した
$ ls -l *.txt
-rw-r--r--  1 sigemura  staff  1536 Feb  9 11:06 x.txt
$ ./mymv s.txt x.txt                <-- s.txt が存在しない場合
./mymv: No such file or directory
$ ./mymv x.txt /x.txt               <-- / に移動する権限がない
./mymv: Permission denied
$ ./mymv x.txt /tmp/x.txt           <-- ハードディスクをまたいだ移動はできない
./mymv: Cross-device link
$
```

リスト 13: ファイルのモード変更 (mychmod.c)

```
//
// ファイルのモードを変更するプログラム
//
#include <stdio.h>       // perror のため
#include <stdlib.h>      // strtol 等のため
#include <sys/stat.h>    // chmod のため

int main(int argc, char *argv[]) {
  // (1) 使い方が正しいか確認する
  if (argc<3) {
    fprintf(stderr, "使用方法␣:␣%s␣<8 進数>␣<file>␣[<file>...]\n", argv[0]);
    return 1;
  }

  // (2) argv[1] を 8 進数と見做し int に変換する
  char *ptr;
  int mode = strtol(argv[1], &ptr, 8);  // 8 進数を表す文字列の値を整数で求める
  if (*argv[1]=='\0' || *ptr!='\0') {
    fprintf(stderr, "\'%s\'␣:␣8 進数の形式が不正\n", argv[1]);
    return 1;
  }
  if ((mode & ~0777)!=0) {
    fprintf(stderr, "\'%s\'␣:␣8 進数の値が不正\n", argv[1]);
    return 1;
  }

  // (3) ファイルのモードを変更する
  for (int i = 2; i<argc; i++) {
```

```
    if (chmod(argv[i], mode)<0) {
      perror(argv[i]);
      return 1;
    }
  }

  return 0;
}
```

リスト 14: mychmod の実行例（動作テスト！！）

```
$ ls -l a b
-rw-r--r--  1 sigemura  kan  0 May  6 21:58 a
-rw-r--r--  1 sigemura  kan  0 May  6 21:58 b
$ ./mychmod
使用方法 : ./mychmod <8 進数> <file> [<file>...]
$ ./mychmod 755 a b
$ ls -l a b
-rwxr-xr-x  1 sigemura  kan  0 May  6 21:58 a
-rwxr-xr-x  1 sigemura  kan  0 May  6 21:58 b
$ ./mychmod 644 b
$ ls -l a b
-rwxr-xr-x  1 sigemura  kan  0 May  6 21:58 a
-rw-r--r--  1 sigemura  kan  0 May  6 21:58 b
$ ./mychmod 644 c
c: No such file or directory
$ ./mychmod 789 a
'789' : 8 進数の形式が不正
$ ./mychmod 12345 a
'12345' : 8 進数の値が不正
$
```

2. 本物の UNIX コマンドとの相違

(a) 複数のファイルを一度に操作できる。

上の解答では複数のファイルを一度に操作できるプログラムになっている。皆さんが作ったプログラムは、ほとんどが一度に一つのファイルしか操作できない。

(b) 色々なオプションが用意されている。

リスト 15: rm コマンドのマニュアル（一部）

```
NAME
     rm, unlink -- remove directory entries

SYNOPSIS
     rm [-dfiPRrvW] file ...
     unlink file

DESCRIPTION
...
```

(c) 以下は間違え

i. 拡張子の操作関係
拡張子は単にファイル名の一部である。特別扱いは必要ない。

ii. ワイルドカード関連
ワイルドカードはシェルの機能である。ワイルドカードをシェルが展開した後、展開結果を用いてコマンドが起動される。(myrmdir の実行例参照)

3. rename システムコールの必要性

(a) ディレクトリの移動が link-unlink ではできない。
ディレクトリの link は禁止されている。(ディレクトリの link を認めるとファイル木が複雑になりすぎる。) ディレクトリの移動が可能なシステムコールが必要である。

(b) ファイルの移動はアトミック操作であるべき。
link-unlink 手法は複数のステップで目的を完了するので、何らかの理由（例えばユーザの Ctrl-C）でプログラムが途中で終了したら、link は完了したが unlink をする前の中途半端な状態になる可能性がある。そのような状態になるのを防ぐために、アトミック操作（不可分操作）としてファイル移動操作を行う rename システムコールが提供される。
通常、システムコールの途中でプログラムは終了できないので、一つのシステムコールで全ての操作を行えば途中の状態で終わることがない。

(c) ハードリンクが使用できないファイルシステムでもファイルの移動は必要。
例えば、IE 電算の Mac のホームディレクトリ（ネットワークドライブ）ではハードリンクが使用できない。(使用できないから`/tmp`でハードリンクの演習をした。) その他に、FAT ファイルシステム（USB メモリ等）でもハードリンクが使用できない。
ハードリンクが使用できないファイルシステムでは、link-unlink の手法は使用できない。ハードリンクが使用できるかできないかに関係なく、ファイルの移動を同じ手法（システムコール）で行えるべきである。

4. stat システムコール，readdir 関数について調べる.

(a) stat システムコール
ファイルの属性情報（メタデータ）を読み出すシステムコールである.

(b) readdir 関数
opendir 関数で開いてあるディレクトリファイルから，ディレクトリエントリを読み出す関数である.

ls コマンドでは，これらを組み合わせて使っている筈だ.（ファイルの一覧を知るために readdir 関数，`ls -l` で属性情報を表示するために stat システムコールが必要だ.）