リスト 1: mycp（解答例１：真っ先に思いつく方法）

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

void fileCopy(FILE *ifp, FILE *ofp) {               // mycat から流用
  int c;                                            // int 型にする必要がある
  while ((c=getc(ifp))!=EOF)
    putc(c, ofp);
}

int main(int argc, char *argv[]) {
  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（return 1; でも同じ）
  }

  FILE *fps = fopen(argv[1], "rb");                 // バイナリモードでオープン
  if (fps == NULL) {                                // コピー元のオープン失敗
    perror(argv[1]);                                // オープン失敗の原因を表示
    exit(1);                                        // エラー終了（return 1; でも同じ）
  }

  FILE *fpd = fopen(argv[2], "wb");                 // バイナリモードでオープン
  if (fpd == NULL) {                                // コピー先のオープン失敗
    perror(argv[2]);                                // オープン失敗の原因を表示
    exit(1);                                        // エラー終了（return 1; でも同じ）
  }

  fileCopy(fps, fpd);                               // ファイルのコピー

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

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

リスト 2: 実行例

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

リスト 3: mycp（解答例２：エラー処理を関数化）

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

// 使用方法を表示して終了する関数
void usage(char *cmd) {
  fprintf(stderr, "Usage:␣%s␣<srcfile>␣<dstfile>\n", cmd);
  exit(1);
}

// エラーチェック付きの拡張 fopen 関数
FILE *eOpen(char *fname, char *mode) {
  FILE *fp= fopen(fname, mode);
  if (fp==NULL) {
    perror(fname);
    exit(1);
  }
  return fp;
}

// mycat から流用したファイルコピー関数
void fileCopy(FILE *ifp, FILE *ofp) {
  int c;
  while ((c=getc(ifp))!=EOF)
    putc(c, ofp);
}

int main(int argc, char *argv[]) {
  if (argc != 3) usage(argv[0]);                    // 引数の個数が予定と異なる

  FILE *fps = eOpen(argv[1], "rb");                 // ファイルのオープン
  FILE *fpd = eOpen(argv[2], "wb");

  fileCopy(fps, fpd);                               // ファイルのコピー

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

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

リスト 4: 解答例２の改悪バージョン１）

```
// インデントが余計に深くなる
int main(int argc, char *argv[]) {
  if (argc != 3) usage();
  else {
    FILE *fps = eOpen(argv[1], "rb");
    FILE *fpd = eOpen(argv[2], "wb");

    fileCopy(fps, fpd);

    fclose(fps);
    fclose(fpd);
  }

  return 0;
}
```

リスト 5: 解答例２の改悪バージョン２

```
// その上、エラー処理の存在に気付き難い
int main(int argc, char *argv[]) {
  if (argc == 3) {
    FILE *fps = eOpen(argv[1], "rb");
    FILE *fpd = eOpen(argv[2], "wb");

    fileCopy(fps, fpd);

    fclose(fps);
    fclose(fpd);
  }
  else usage();     // この行が目立たない

  return 0;
}
```

リスト 6: mydiff の解答例（その１）

```
// mydiff.c : 異なる最初の行を表示する
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAXLINE 100

// 使用方法を表示して終了する関数
void usage(char *cmd) { ... }

// エラーチェック付きの拡張 fopen
FILE *eOpen(char *fname, char *mode) { ... }

// 異なる行を表示する
void printDiffLine(int l, char *line1, char *line2) {
  printf("%d 行\n", l);
  printf("> %s", line1);
  printf("---\n");
  printf("> %s", line2);
  exit(1);
}

int main(int argc, char *argv[]) {
  char buf1[MAXLINE];
  char buf2[MAXLINE];

  if (argc!=3) usage(argv[0]);          // エラーになる場合を早めに判断して排除

  FILE *fp1 = eOpen(argv[1], "r");
  FILE *fp2 = eOpen(argv[2], "r");

  for (int ln=1; fgets(buf1, MAXLINE, fp1)!=NULL &&
                 fgets(buf2, MAXLINE, fp2)!=NULL;   ln++) {

    if (strcmp(buf1, buf2)!=0)
      printDiffLine(ln, buf1, buf2);
  }

  // ファイルは自動的にクローズされる
  return 0;
}
```

リスト 7: 実行例

```
$ cat a.txt
hello
world
!!
$ cat b.txt
hello
World
!!
$ cat c.txt
hello
world
!
$ cat d.txt
hello
world
$ cat e.txt
hello
world
!!
$ mydiff a.txt
Usage: mydiff <file1> <file2>
$ mydiff a.txt b.txt
2 行
> world
---
> World
$ mydiff a.txt c.txt
3 行
> !!
---
> !
$ mydiff2 a.txt d.txt
$
```

リスト 8: mydiff の解答例（その２）

```
// mydiff.c : 異なる最初の行を表示する
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAXLINE 100

// 使用方法を表示して終了する関数
void usage(char *cmd) { ... }

// エラーチェック付きの拡張 fopen
FILE *eOpen(char *fname, char *mode) { ... }

// 異なる行を表示する
void printDiffLine(int l, char *line1, char *line2) { ... }

// 短いファイルを表示する
void shortFile(char *fname) {
  printf("%s␣が短い\n", fname);
  exit(1);
}

int main(int argc, char *argv[]) {
  if (argc!=3) usage(argv[0]);

  FILE *fp1 = eOpen(argv[1], "r");
  FILE *fp2 = eOpen(argv[2], "r");

  for (int ln=1; ; ln++) {
    char buf1[MAXLINE];                    // 変数の有効範囲は
    char buf2[MAXLINE];                    //    なるべく狭くする

    char *r1 = fgets(buf1, MAXLINE, fp1);
    char *r2 = fgets(buf2, MAXLINE, fp2);

    if (r1==NULL && r2==NULL) break;
    if (r1==NULL) shortFile(argv[1]);
    if (r2==NULL) shortFile(argv[2]);

    if (strcmp(buf1, buf2)!=0)
      printDiffLine(ln, buf1, buf2);
  }

  return 0;
}

/* 実行例
$ mydiff a.txt d.txt
d.txt が短い
$ mydiff a.txt e.txt
$
*/
```