```c
/*
 * This is a free program sample that may be reproduced in any form.
 * The author's information should be retained to preserve its identity.
 *
 * Date written: December 18, 2003
 * Written by:   Peraphon Sophatsathit
 * Department of Mathematics, Faculty of Science, Chulalongkorn University.
 * email: Peraphon.S@chula.ac.th
 * http://pioneer.netserv.chula.ac.th/~sperapho
 *
 * Computer Systems (2301274) class supplement.
 * Description: This program demonstrates how to read input parameters
 *          from the command line and issue usage information if
 *          need be.  The code may not compile/run on DOS platform.
 *
 * syntax of invocation:
 *          program_name  infile_name  outfile_name
 */

#include    <stdio.h>
#include    <stdlib.h>
#include    <string.h>
#include    <unistd.h>

#define     rmode       "r"
#define     wmode       "w"
#define     Normal      0
#define     Fail_file   1
#define     Fail_usage  2
#define     Bsize       BUFSIZ

/*
 * function prototype
 */
int   process_io(char **);
int   help_msg(char *);

/*
 * limit input parameters to two, i.e. input and output files.
 * Other options can be extracted by means of 'getopt'.
 */
int
main(int argc, char *argv[])
{
      int   rt_code = Normal;

      switch (argc)
      {
            case 3:
                  rt_code = process_io(argv);
                  break;
            case 1:
                  perror("missing input parameters");
            default:
                  rt_code = help_msg(argv[0]);
                  break;
      }
      return rt_code;
}
```

```
/*
 * If the input file is missing, 'fopen' will fail.
 */
int
process_io(char *av[])
{
      FILE  *fi, *fo;
      int   len;
      char  tmp[Bsize-1];

      if ((fi = fopen(av[1], rmode)) == NULL ||
          (fo = fopen(av[2], wmode)) == NULL)
      {
            printf("Unable to open files %s and/or %s\n", av[1], av[2]);
            return Fail_file;
      }
      while (fgets(tmp, Bsize, fi) != NULL)
      {
            /*
             * just to demonstrate how to get rid of the newline
             * character from the read buffer, but is put back in
             * subsequent transfer (fprintf) to the output file
             * which could be done in much simpler and faster
             * approaches.
             */
            len = strlen(tmp);
            tmp[len-1] = '\0';
            fprintf(fo, "%s\n", tmp);
      }
      fclose(fi);
      fclose(fo);

      /*
       * This is a quick and dirty way to execute a system command
       * and delete a file within the program via 'system' and 'unlink'.
       * There is a better approach to do this on UNIX.
       */
      sprintf(tmp, "cat %s", av[2]);
      (void)system(tmp);
      (void)unlink(av[2]);

      return Normal;
}

/*
 * inform the user of the program usage syntax.
 */
int
help_msg(char *pname)
{
      printf("Usage: %s  input_filename  output_filename\n\n", pname);
      return Fail_usage;
}
```