```c
/*
 ============================================================================
 Name        : MemoryMappedCopy.c
 Author      : W. Schilling
 Version     :
 Copyright   : Your copyright notice
 Description : This program will copy a file using memory mapped IO.
 ============================================================================
 */

#include <sys/mman.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/types.h>  /* Type definitions used by many programs */
#include <stdio.h>      /* Standard I/O functions */
#include <stdlib.h>     /* Prototypes of commonly used library functions,
                           plus EXIT_SUCCESS and EXIT_FAILURE constants */
#include <unistd.h>     /* Prototypes for many system calls */
#include <errno.h>      /* Declares errno and defines error constants */
#include <string.h>     /* Commonly used string-handling functions */

/**
 * This is the main function.  It starts the program.
 * @param argc This is the number of arguments.  For the program to work it must be 3.
 * @param argv These are the arguments.  arg[0] is the name of the program.  arg[1] is the source file.  arg[2] is the destination file.
 * @return
 */
int main(int argc, char *argv[])
{
    char *src, *dst;
    int fdSrc, fdDst;
    struct stat sb;

    // Check the usage provided by the user of the program.
    if (argc != 3)
    {
        fprintf(stderr, "%s source-file dest-file\n", argv[0]);
        exit(-1);
    }

    // Open the file that is the source as a read only file.
    fdSrc = open(argv[1], O_RDONLY);
    if (fdSrc == -1)
    {
        fprintf(stderr, "open failed.  Source file does not exist.\n");
        exit(-1);
    }
    /* Use fstat() to obtain size of file: we use this to specify the
       size of the two mappings */
    if (fstat(fdSrc, &sb) == -1)
    {
        fprintf(stderr, "fstat could not read statistics about the file.");
```

```
53          exit(-1);
54      }
55
56      /* Handle zero-length file specially, since specifying a size of
57         zero to mmap() will fail with the error EINVAL */
58      if (sb.st_size == 0)
59      {
60          exit(EXIT_SUCCESS);
61      }
62
63      // Create a shared location in memory that is the source.
64      src = mmap(NULL, sb.st_size, PROT_READ, MAP_PRIVATE, fdSrc, 0);
65      if (src == MAP_FAILED)
66      {
67          fprintf(stderr, "mmap failed to create a shared partition");
68          exit(-1);
69      }
70
71      // Open the destination file for writing.
72      fdDst = open(argv[2], O_RDWR | O_CREAT | O_TRUNC, S_IRUSR | S_IWUSR);
73      if (fdDst == -1)
74      {
75          fprintf(stderr, "open of destination file failed.");
76          exit(-1);
77      }
78
79      if (ftruncate(fdDst, sb.st_size) == -1)
80      {
81          fprintf(stderr, "ftruncate");
82          exit(-1);
83      }
84
85      // Map the destination file to point to a mapped segment of memory.
86      dst = mmap(NULL, sb.st_size, PROT_READ | PROT_WRITE, MAP_SHARED, fdDst, 0);
87      if (dst == MAP_FAILED)
88      {
89          fprintf(stderr, "mmap");
90          exit(-1);
91      }
92
93      // Copy the memory from the source to the destination.
94      memcpy(dst, src, sb.st_size);       /* Copy bytes between mappings */
95
96      if (msync(dst, sb.st_size, MS_SYNC) == -1)
97      {
98          fprintf(stderr, "msync");
99          exit(-1);
100     }
101
102     // Close the files.
103     close(fdSrc);
104     close(fdDst);
105
```

```
106         printf("Copy of %d bytes from %s to %s is now completed.\n", sb.st_size,
107                 argv[1], argv[2]);
108
109         exit(EXIT_SUCCESS);
110     }
111
```