Exercise

Write a program, similar to the *cp* command, which you use for implementation the *mmap ()* system call and a C function that allows us to copy memory, such as *memcpy ()*. To know the size of the source file you can use stat () and to set the size of the target file *ftruncate ()* can be used.

Solution

```
#include<sys/types.h>
#include<sys/stat.h>
#include<fcntl.h>
#include<stdlib.h>
#include<stdio.h>
#include<errno.h>
#include <unistd.h>
#include <sys/mman.h>

int main(int argc, char *argv[]) {
   struct stat attr;
   int fd1
   int fd2;
   int tamano;
   char *mem1,*mem2;

   if ((fd1=open(argv[1],O_RDONLY, S_IRWXU))<0) {
      perror("\n open error");
      exit(-1);
   }

   if ((fd2=open(argv[2],O_CREAT|O_RDWR,S_IRWXU))<0) {
      perror("\n open error");
      exit(-1);
   }
   if (stat(argv[1],&atributos) < 0) {
      perror("\n lstat error");
   }
   tamano=attr.st_size;


   printf ("Adjusting the size of the file %s a %d \n",argv[2],tama);
   ftruncate(fd2, tama);
```

```
    mem1 = (char *)mmap(0, tamano, PROT_READ, MAP_SHARED, fd1, 0);
    if (memoria1 == NULL) {
       perror(" Mapping failed ");
       exit(-1);
    }


    mem2 = (char *)mmap(0, tamano,PROT_WRITE, MAP_SHARED, fd2, 0);
    if (mem2 == NULL) {
       perror("Mapping failed ");
       exit(-1);
    }

    memcpy(mem2,mem1,tamano);

    if (munmap (mem1, tamano) == -1) {
       perror(" Error closing map \n");
       exit(-1);
    }
    if (munmap (mem2, tamano) == -1) {
       perror("Error closing map \n");
       exit(-1);
    }
     return 0;

}
```