National University of Computer and Emerging Sciences

Laboratory Manual

for

Operating Systems Lab

(CL-220)

Course Instructor	Mr. Mubashar Hussain
Lab Instructor (s)	Haiqa Saman, Rasaal Ahmad
Section	BCS-4G & BCS-4F
Semester	Spring 2024

Department of Computer Science

FAST-NU, Lahore, Pakistan

Objectives

In this lab, students will practice memory mapped files:

- A **memory-mapped file** is a segment of virtual memory that has been assigned a direct byte-for-byte correlation with some portion of a file or file-like resource.
- This let us perform I/O without using read() and write () system calls.

1.1 Name

mmap, munmap - map or unmap files or devices into memory

1.2 Synopsis

#include <sys/mman.h>
void *mmap(void *addr, size_t lengthint " prot ", int " flags ,
int fd, off_t offset);int munmap(void *addr, size_t length);
See NOTES for information on feature test macro requirements.

1.3 Description

mmap() creates a new mapping in the virtual address space of the calling process. The starting address for the new mapping is specified in the *addr*. The *length* argument specifies the length of the mapping.

If *addr* is NULL, then the kernel chooses the address at which to create the mapping; this is the most portable method of creating a new mapping. If *addr* is not NULL, then the kernel takes it as a hint about where to place the mapping; on Linux, the mapping will be created at a nearby page boundary. The address of the new mapping is returned as the result of the call.

The contents of a file mapping (as opposed to an anonymous mapping; see **MAP_ANONYMOUS** below), are initialized using *length* bytes starting at offset in the file (or another object) referred to by the file descriptor *fd*. *offset* must be a multiple of the page size as returned by *sysconf(_SC_PAGE_SIZE)*.

The *prot* argument describes the desired memory protection of the mapping (and must not conflict with the open mode of the file). It is either **PROT_NONE** or the bitwise OR of one or more of the following flags:

PROT_EXEC
Pages may be executed.
PROT_READ
Pages may be read.
PROT_WRITE
Pages may be written.
PROT_NONE

Pages may not be accessed.

The *flags* argument determines whether updates to the mapping are visible to other processes mapping the same region and whether updates are carried through to the underlying file. This behavior is determined by including exactly one of the following values in *flags*:

MAP_SHARED
Share this mapping. Updates to the mapping are visible to other processes that map this file and are carried through to the underlying file. The file may not be updated until ***msync(2)*** or **munmap**() is called.
MAP_PRIVATE
Create a private copy-on-write mapping.

munmap()

The **munmap**() system call deletes the mappings. The region is also automatically unmapped when the process is terminated. On the other hand, closing the file descriptor does not unmap the region.

Lab Questions

Question 1

Write C/C++ code for a program that takes as a command line argument a file name. Your program will make a memory map of the file and change the case of English alphabets. Make two threads for this purpose. The first thread will change the case from the first half of the map and the second thread will change the case from the second half of the map. Assume that there are 100 bytes in the file. (Hint: Create a single map. Pass the map pointer to the first thread as a parameter. For the second thread, add 50 to the map pointer and pass it to the second thread as parameter, i.e., map+50.

Sample Data for File:

v1gU6OTg7MifG7zmQWp04ZEGRi!q1uFs%9RzZWc^!@#QL7jBMNKUQVEAIsKZia40M3Tq^&eGE
MEkkSagfU&7mU3PbQ*zsiJm23Hq