National University of Computer and Emerging Sciences

Laboratory Manual

for

Operating Systems Lab

Department of Computer Science

FAST-NU, Lahore, Pakistan

MEMORY MAPPING

Memory mapping is a technique in operating systems that maps a file on disk into the address space of a process. This enables the process to treat the file as if it were part of its own memory, allowing for efficient access to the file's contents.

In C/C++, memory mapping is typically achieved through the use of the mmap function from the sys/mman.h header. The function maps a file into memory, returning a pointer to the memory region.

Here is an example code in C++ that demonstrates the use of memory mapping to read the contents of a file:

```
#include <sys/mman.h>

#include <fcntl.h>

#include <unistd.h>

#include <iostream>


int main(int argc, char *argv[]) {

  int fd = open(argv[1], O_RDONLY);

  if (fd == -1) {

    perror("open");

    return 1;

  }


  char *map = (char*)mmap(NULL, 100, PROT_READ, MAP_PRIVATE, fd, 0);

  if (map == MAP_FAILED) {

    perror("mmap");

    return 1;

  }
```

```
  std::cout << "Contents of file: " << map << std::endl;


  if (munmap(map, 100) == -1) {
    perror("munmap");
    return 1;
  }


  close(fd);
  return 0;
}
```

In this example, the file is opened using the open function and its file descriptor is passed to the mmap function to map the contents of the file into memory. The mmap function returns a pointer to the memory region, which can then be used to access the contents of the file. Finally, the munmap function is called to unmap the file from memory.

Question 1: Write a C/C++ program that accepts a file name as a command line argument and maps the contents of the file into memory. The program should then search for a specified word in the file and replace it with another word. The program should use memory mapping for efficient access to the file's contents.

The program should be able to handle large files, and should avoid reading the entire file into memory at once.

Function signature:

```
void replaceWordInFile(const char *fileName, const char *wordToReplace, const char *replacementWord);
```

Example:

```
replaceWordInFile("file.txt", "hello", "goodbye");
```

This call should replace all occurrences of the word "hello" with the word "goodbye" in the file "file.txt".

Question 2: Write a C/C++ program that accepts a file name as a command line argument and creates a memory map of the file. The program should then use two threads to replace all integers in the file with spaces. The first thread will be responsible for replacing integers in the first half of the map, while the second thread will replace integers in the second half of the map. The file is assumed to have 100 bytes. To accomplish this, pass the map pointer to the first thread as a parameter and add 50 to the map pointer and pass it as a parameter to the second thread.

Sample Data for File:

> v1gU6OTgN7DMifG7zmQWp04ZEyGmRifq1uFsS9RzZWcCQL7jBMNKUQVEAIsKZia4
> 0M3TqJeGEMEkkSagfUc7mU3PbQ1zsiJm23Hq