National University of Computer and Emerging Sciences

Operating Systems Lab

Lab Manual 12

Muhammad Hassan Raza

Fall 2024

Department of Software Engineering
FAST-NU, Lahore, Pakistan

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 using 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 fd = open("example.txt", 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: " << map << std::endl;

    munmap(map, 100);
    close(fd);
    return 0;
}
```

Q1: Write a program in C/C++ that accepts a file name as a command-line argument, maps its contents into memory, and prints the first 100 bytes of the file.
Hint: Use mmap and handle errors properly.

Q2: Modify the program from Q1 to print the size of the mapped file and the memory address returned by mmap.

Q3: Write a C/C++ program that searches for a word in a memory-mapped file and counts its occurrences. The program should accept the file name and the word as command-line arguments.

Example Usage:
./searchWord file.txt "hello"

Output:
If the word "hello" appears 5 times, print:
The word "hello" was found 5 times.

Q4: Enhance the program from Q3 by replacing all occurrences of the word with another word provided as a command-line argument. Ensure the program handles:
- Words of different lengths (e.g., replacing "hi" with "goodbye").
- Large files efficiently, avoiding reading them entirely into memory.
- Any potential alignment issues in memory mapping.

Function signature:
*void replaceWordInFile(const char *fileName, const char *wordToReplace, const char *replacementWord);*

Q5: Write a multithreaded program that memory maps a file and uses two threads to convert all uppercase letters to lowercase:
- Thread 1 processes the first half of the file.
- Thread 2 processes the second half.

The program should:
- Ensure the file is at least 100 bytes.
- Use appropriate synchronization mechanisms if necessary.

Hint: Use pthread_create for threading and pass portions of the memory map to each thread.