3 Experiment 3: File Manipulation Using System Calls[1][3][2]

3.1 Objective

The goal of this lab is to introduce and apply fundamental system calls like open, read, write, lseek, and close through programming exercises, addressing practical challenges encountered in real-world scenarios.

3.2 Reading Material

3.2.1 System Calls

System calls are functions provided by the operating system kernel that enable user-level processes to request services from the operating system. Examples: Common system calls include open, read, write (for file operations), fork, exec (for process control), malloc, free (for memory management), socket, bind, connect (for networking), and many others.

System Call	Description	Programming Syntax
open	Used to open a file and obtain a file descriptor. Allows specifying flags for read, write, create, and permissions.	`int fd = open("file.txt", O_RDONLY);`
close	Closes a file descriptor, releasing resources associated with the file.	`close(fd);`
read	Reads data from an open file into a buffer in memory.	`read(fd, buffer, nbytes);`
write	Writes data from a buffer in memory to an open file.	`write(fd, buffer, nbytes);`
lseek	Moves the file pointer to a specified position in the file.	`lseek(fd, offset, SEEK_SET);`
unlink	Deletes a file by name.	`unlink("file.txt");`
mkdir	Creates a new directory.	`mkdir("new_dir", 0755);`
rmdir	Removes a directory if it's empty.	`rmdir("directory");`
rename	Renames a file or directory.	`rename("old_name", "new_name");`
stat	Retrieves file status information like permissions, size, and timestamps.	`struct stat fileStat; stat("file.txt", &fileStat);`
chmod	Changes file permissions.	`chmod("file.txt", 0644);`
chown	Changes file ownership.	`chown("file.txt", uid, gid);`
link/symlink	Creates hard or symbolic links to files.	`link("source", "target");`

Table 2: File management related system calls

3.2.2 Sample Programs

1. Program to Create and Open a File for Reading

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>

int main() {
    int fileDescriptor;

    // Create a new file named "file.txt" and open it for reading
    fileDescriptor = open("file.txt", O_CREAT | O_RDONLY, 0644);

    close(fileDescriptor); // Close the file

    return 0;
}
```

2. Program to Read from Console and Write to Console

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#define BUFFER_SIZE 1024

int main() {
    char buffer[BUFFER_SIZE];
    //0 is the file descriptor of standard input.
    ssize_t bytesRead = read(0, buffer, BUFFER_SIZE);
    write(1, buffer, bytesRead); //1 is the file descriptor of standard output.

    return 0;
}
```

3. Program to Append Data into a File

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>

int main() {
    char data[] = "This data will be appended to the file.\n";
```

```
    int fileDescriptor;

    fileDescriptor = open("file.txt", O_WRONLY | O_CREAT | O_APPEND, 0644);
    write(fileDescriptor, data, strlen(data));
    close(fileDescriptor);

    return 0;
}
```

4. Program to Read from and Write to Files

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>

#define BUFFER_SIZE 1024

int main() {
    char buffer[BUFFER_SIZE];
    int readFd, writeFd;

    readFd = open("source.txt", O_RDONLY);
    writeFd = open("destination.txt", O_WRONLY | O_CREAT | O_TRUNC, 0644);

    ssize_t bytesRead;
    while ((bytesRead = read(readFd, buffer, BUFFER_SIZE)) > 0) {
        write(writeFd, buffer, bytesRead);
    }

    close(readFd);
    close(writeFd);

    return 0;
}
```

5. a C program that reads characters from the 11th to the 20th position from a file named "input.txt" using the lseek system call.

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>

#define BUFFER_SIZE 11

int main() {
```

```
    int fileDescriptor;
    char buffer[BUFFER_SIZE];

    fileDescriptor = open("input.txt", O_RDONLY);
    lseek(fileDescriptor, 10, SEEK_SET);
    read(fileDescriptor, buffer, BUFFER_SIZE - 1);
    buffer[BUFFER_SIZE - 1] = '\0';
    printf("Characters from 11th to 20th position: %s\n", buffer);
    close(fileDescriptor);

    return 0;
}
```

Video Reference:

https://youtube.com/playlist?list=PLWjmN065fOfGdAZrlP6316HVHh8jlve

References

[1] Greg Gagne Abraham Silberschatz, Peter B. Galvin. *Operating System Concepts*. Wiley, 10 edition, 2018.

[2] Sumitabha Das. *Unix Concepts And Applications*. Wiley, 4 edition, 2006.

[3] Richard Stones Neil Matthew. *Beginning Linux Programming*. Wiley, 4 edition, 2007.

3.3 Lab Exercises

Exercise 1: Write a program in C using system calls that lets users choose to copy either the first half or the second half of a file by entering 1 or 2.

Exercise 2: Create a C program using system calls that keeps reading from the console until the user types '$'. Save the input data to a file called 'input.txt'."

Exercise 3: Write a C program that encrypts a text file using a simple encryption technique and saves the encrypted content to a new file.
Requirements:
Input: Provide a text file named "input.txt" with plain text content.
Encryption Technique: Shift each character in the file content by a fixed number