## **Objectives**

- To perform reading data from standard input and writing data to standard output with their respective file descriptors using read() and write() system calls.
- To perform open and close operations on a file using open() and close() system calls.
- To demonstrate an understanding of open flags
- To perform reading from a file and writing data to the file with their respective file descriptors using read() and write() system calls.

### Pre-Lab Theory

#### 1. read() synopsis:

int read(int fd, char\*buffer, int size)

#### Description:

read from file descriptor fd

## file descriptor:

Represents a handle for the opened file. It is the index of the process file table entry.

Default Descriptors in the process file table with symbolic names:

Standard output: STDOUT\_FILENO

Standard input: STDIN\_FILENO

Standard Error: STDERR\_FILENO

**buffer:** The data read from the file descriptor representing a file/console is stored in the buffer

size: The size/chunk/block to be read in the buffer

return: The number of bytes successfully read

#### 2. write() synopsis:

int write(int fd, char\*buffer, int size)

#### file descriptor:

Represents a handle for the opened file. It is the index of the process file table entry.

Default Descriptors in the process file table with symbolic names:

Standard output: STDOUT FILENO

Standard input: STDIN FILENO

Standard Error: STDERR\_FILENO

**buffer:** The data stored in the buffer is written to the file represented by the file descriptor

size: The size/chunk/block to be written to the file

return: The number of bytes successfully written

# 3. open() synopsis:

int open(char\* path, int mode\_flags)

path: The location of the file

mode\_flags: The access flags, O RDONLY, O WRONLY, O RDWR, O CREAT

return: The file descriptor of the file if successfully opened.

## 4. Close(int fd):

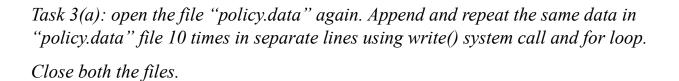
int close(int fd)

#### In-Lab Tasks

```
header file: unistd.h, stdio.h
System call: read(), write(), open(), close()
Taskl(a)
Include the header files and the following line of code in your program. Compile
and run the code and describe its behavior.
char buffer[] = "Welcome";
write(STDOUT_FILENO, buffer, sizeof(buffer);
Brief the working/output:
Task 1(b)
Include the header files, declare the required variables, and the following line of
code in your program. Compile and run the code and brief the working of the code.
bytes_read = read(STDIN_FILENO, buffer, sizeof(buffer));
if(bytes_read < 0) printf("read() ERROR");exit();</pre>
bytes_write = write(STDOUT_FILENO, buffer, bytes_read);
if(bytes write < 0) printf("write() ERROR");exit();</pre>
Brief the Working/output:
```

Task 2(a) first, create the file "policy.data" in the present working directory using the open() system call in read/write mode.
Error-free Code:
Task 2(b)
Write the statement "Honesty is the Best Policy" in the above-created file using a write system call
Error-free Code:
Task 2(c) Create another file, "backup.data" in the directory one level above the present working directory with write mode flag.
Error-free Code:
Task 2(d) Read data from "policy.data" file and write to the "backup.data" file .  Error-free Code:

#### LAB # 3 UNIX Input/Output



Error Free Code:

Task 3(b):open the "policy.data" file and "backup.data" file in the appropriate mode and copy the whole file "policy.data" to the "backup.data" file. Detect End-of-File during reading data from source file. Print the number of bytes written.

Error-free Code: