FAST NUCES

Laboratory Manual

For

Operating Systems

Instructor(s)	Muhammad Saifullah Usman
Section	BCS-4A

Department of Computer Science

National University of Computer & Emerging Sciences, Lahore

Spring 2022

System calls (Low-Level Functions)

1. **Open**

   Opens a file

   *int open(const char * pathname, int flags);*

   open function returns a file descriptor as below:

   0 – standard input

   1 – standard output

   2 – standard error

   -1 – operation is failed

2. **Read**

   Reads from an open file

   *ssize_t read(int fd, void *buf, size_t count);*

3. **Write**

   Write to a file

   *ssize_t write(int fd, const void *buf, size_t count);*

4. **Close**

   Close a file

   int close(int fd);

Pipes

On UNIX and Linux systems, ordinary pipes are constructed using the function

• int pipe (int fd[2]) -- creates a pipe

• returns two file descriptors, fd[0], fd[1].

• fd[0] is the read-end of the pipe

• fd[1] is the write-end.

Question # 1 (Unamed Pipes)

Design a program using ordinary pipes in which the parent process sends a message from a file named file.txt to a child process, and the child process removes the occurrences of all the special characters including &,@,#,%,*,? &,$,", and ~. And send the modified version back to the parent process and the parent process writes the modified data to the file updated.txt. This will require using two pipes, one for sending the original message from the first to the second process, and the other for sending the modified message from the second back to the first process.

Hint: create the file manually, by yourself.

Question # 2 (Execlp)

Write a program that takes the program name from the user and executes it as a child process. The parent waits, and once the child returns, the control goes to the next iteration of the user input. The user should terminate the program with Ctrl+D like keystroke (depends on the operating system).