National University of Computer and Emerging Sciences



Laboratory Manual

for

Operating Systems Lab

| Course Instructor | Mr. Raziuddin |
| --- | --- |
| Lab Instructor(s) | Usman Anwer |
| Section | BDS-A |
| Semester | Fall-2022 |

Department of Computer Science

FAST-NU, Lahore, Pakistan

**Question 1:** (Ordinary Pipes) Write a C or C++ program that accepts a filename as command line argument and send it to the child process. The child reads the data from the file which is the array of numbers and sends the array to the parent which calculates the palindromes from the arrays.

The file contents are:

12 121 131 45 65

21 111 321 34 66

31 51 323 55 77

You have to find the palindromes row by row.

The result of the above file are:

121 131

111 66

323 55 77

A number is called a palindrome if it remains the same despite flipping. like 131 after flipping is also 131.

**Question :02**

**Named Pipes**

* + **Step 1:** background

A pair of unrelated processes can use a 'named pipe' to pass information between them. Unlike 'unnamed pipes', however, 'named pipes' are accessed as a file on the file system. This allows a situation where two processes started in separate shells can communicate with each other through a 'named pipe' on the file system. A named pipe, or FIFO, can be created using the 'mkfifo()' function. It can be removed (like any other file on the file system) using the 'unlink()' function. Once a named pipe file exists, programs can open it like they would other files and then use the file descriptor obtained to perform regualar file IO operations on the (read, write, close...).

* + **Step 2:** my\_npipe

You will write two simple programs 'my\_npipe\_reader.c' and 'my\_npipe\_writer.c' that use a named pipe to communicate. The 'my\_npipe\_reader' program will set up a named pipe using 'mkfifo()', open it read only, and read strings from it until it recieves the string 'exit'. The writer will open the named pipe file, read strings from the user and write them to the named pipe. When the user enters 'exit', the program will write the string to the pipe and then exit. Execution should look something like this (note that you must start the reader first):

reader:  
gawaine:$ ./my\_npipe\_reader  
Creating named pipe: /tmp/mypipe  
Waiting for input...Got it: 'hello world'  
Waiting for input...Got it: 'foober goober'  
Waiting for input...Got it: 'exit'  
Exiting  
  
  
writer:  
gawaine:$ ./my\_npipe\_writer  
Opening named pipe: /tmp/mypipe  
Enter Input: hello world  
Writing buffer to pipe...done  
Enter Input: foober goober  
Writing buffer to pipe...done  
Enter Input: exit  
Writing buffer to pipe...done  
Exiting

Note that the 'my\_npipe\_reader' and 'my\_npipe\_writer' need to be executed in separate shells at the same time. The reader stops at 'Waiting for input...' until it recieves data from the pipe (the read completes).

**Alternatively, You can also follow the procedure discussed in class to solve Question 2.**