Exercise 1: Develop a program to execute ls | wc –l command by using pipe system call

Exercise 2: Develop a sample application where FIFO is used for inter process communication

Exercise 3: Write a program that creates a child process. Parent process writes data to pipe and child process reads the data from pipe and prints it on the screen.

Exercise 4: Client server application using pipes: Client sends file name to server and server sends file content or error as reply to client's request.

Exercise 5: Create two processes P1 & P2.
P1: Creates a shared memory and writes a message on the shared memory.
P2: Attaches the shared memory and reads the message from the shared memory.

Exercise 6: Simulate the producer consumer problem (bounded buffer) on a shared memory.

Exercise 7: Client-Server Communication using Message Queues
Problem Statement:
Two processes, client and server, communicate via two message queues "Up" and "Down".
The client reads a message from the standard input and sends it to the server via the Up queue, then waits for the server's answer on the Down queue. The server is specialized in converting characters from lower case to upper case and vice versa. Therefore, if the client sends the message "lower case" via the Up message queue, the server will read the message, convert it, and send "LOWER CASE" via the Down queue. When the client receives the message from the server, it prints it out. You may assume the maximum size of any message is 256 bytes.
Multiple clients must be able to connect to the up and down queues. However, what happens if one client puts a letter to convert on the system and another client is waiting for it's response from the queue? There are different ways to handle this, but you should handle this using typed messages.
Each client has a particular client number based on the last 4 digits of its process ID. Use this 4-digit number as the client identifier, and use typed messages on the queue so that each client can always be sure to receive the letter that it is waiting on to be converted.

Exercise 8: Implement a C program that uses lseek system call to copy the contents of one file into another file at position 100.

Exercise 9: Develop a program that does the following. Creates 2 processes, a parent and a child using fork(). The parent prints the value of I from 0 to 1000 and then exits. The child process sleeps for a second after it is created, sends a SIGUSR1 signal to the parent and then exits. The parent could catch that signal, print the value of I at that instant, sleep for a second and then continue

Exercise 10: write a program to create a file test.txt using system call, and then do fork to create two processes. In parent process write 2 lines about hyderabad and in child program read whatever you have written to test.txt. Are you able to read test.txt in child process? If yes, what inference can you take out about? What are the different parameters are shared between parent and child when fork is done?

Exercise 11: Do the above question using exec to generate child process. Now, are you able read the lines written to test.txt? If no why?

Exercise 12: Write a programs that will both send and receive messages and construct the following dialog between them
(Process 1) Sends the message "Are you hearing me?"
(Process 2) Receives the message and replies "Loud and Clear".
(Process 1) Receives the reply and then says "I can hear you too".