### **Task A**

Q) Learn what is a named pipe. Learn how to create and use a named pipe using command line in Linux (or any UNIX system).

In computing, a named pipe (also known as a FIFO) is one of the methods for intern-process communication.

* A named pipe, can last as long as the system is up, beyond the life of the process. It can be deleted if no longer used where as a traditional pipe is “unnamed” and lasts only as long as the process.
* Usually a named pipe appears as a file and generally processes attach to it for inter-process communication. A FIFO file is a special kind of file on the local storage which allows two or more processes to communicate with each other by reading/writing to/from this file.
* A FIFO special file is entered into the filesystem by calling mkfifo() in C. Once we have created a FIFO special file in this way, any process can open it for reading or writing, in the same way as an ordinary file. However, it has to be open at both ends simultaneously before you can proceed to do any input or output operations on it.

Creation of named pipe using command line in linux.

* mkfifo <pipe-name>

## **Task B**

Q) Run one consumer and one producer concurrently.

* When one consumer and producer are are run concurrently the producer generates a process is using getpid()
* The producer and consumer gets executed arbitrarly until:
* Case 1: Kill the Consumer with Ctrl-C and Leave producer running.

If the consumer is killed using ctrl+c command the producer writes in to terminal with error no: 32 which denotes the broken pipe due to SIGPIPE (SIGPIPE is the "broken pipe" signal, which is sent to a process when it attempts to write to a pipe whose read end has closed (or when it attempts to write to a socket that is no longer open for reading), but not vice versa. The default action is to terminate the process.)

* Case 2: Kill the producer with Ctrl-C and Leave consumer running.

If the producer is killed using ctrl+c command the consumer writes onto terminal with error no: 0 with perror:Success which denotes (The value of errno is 0 at program startup, but it is never set to 0 by any library function. The value of errno may be set to nonzero by a library function call whether or not there is an error, provided the use of errno is not documented in the description of the function in the C Standard. It is meaningful for a program to inspect the contents of errno only after an error has been reported. More precisely, errno is meaningful only after a library function that sets errno on error has returned an error code.)

Q) Run one consumer and multiple producers concurrently.

When there is one consumer and multiple producers the writing and reading at producer and consumer happens in an asynchronous manner. This asynchronous nature increase with the increase in number of producers. This will go into its critical section whenever a consumer has calculated their part and exited their critical section.

Q) Run multiple consumers and one producer concurrently.

When there are multiple consumer and one producers the writing and reading at producer and consumer happens in an asynchronous manner. The distribution of process from one producer to multiple consumer. This asynchronous nature increases with the increase in number of producers.

Q) Run multiple consumers and multiple producers concurrently.

There is a fixed size buffer and the producer produces items and enters them into the buffer. The consumer removes the items from the buffer and consumes them. When the buffer size exceeds the there issue arises of synchronization.

The producer consumer problem can be resolved using semaphores.

**TASK C**

The above code lacks Mutex semaphore where mutex uses locks around the critical section where Locking a mutex means calling Down and Unlocking a semaphore means calling UP.

Two types of synchronization needed

• Locking the buffer to prevent concurrent modification

• Informing the other side that data/buffer is available