INTERPROCESS COMMUNICATION

- **Inter-Process communication (IPC), is the mechanism whereby one process can communicate with another process, i.e exchange data.**
- **IPC in linux can be implemented using pipe, shared memory, message queue, semaphore, signal or sockets.**

Pipe

- **Pipes are unidirectional byte streams which connect the standard output from one process into the standard input of another process.**
- **A pipe is created using the system call *pipe* that returns a pair of file descriptors.**
- **The descriptor pfd[0] is used for reading and pfd[1] is used for writing.**
- **Can be used only between parent and child processes.**

Shared memory

- **Two or more processes share a single chunk of memory to communicate randomly.**
- **Semaphores are generally used to avoid race condition amongst processes.**
- **Fastest amongst all IPCs as it does not require any system call.**
- **It avoids copying data unnecessarily.**

Message Queue

- **A message queue is a linked list of messages stored within the kernel**
- **A message queue is identified by a unique identifier**
- **Every message has a positive long integer type field, a non-negative length, and the actual data bytes.**
- **The messages need not be fetched on FCFS basis. It could be based on type field.**

Semaphores

- **A semaphore is a counter used to synchronize access to a shared data amongst multiple processes.**
- **To obtain a shared resource, the process should:**
  - **Test the semaphore that controls the resource.**
  - **If value is positive, it gains access and decrements value of semaphore.**
  - **If value is zero, the process goes to sleep and awakes when value is > 0.**
- **When a process relinquishes resource, it increments the value of semaphore by 1.**

Producer-Consumer problem

- **A producer process produces information to be consumed by a consumer process**
- **A producer can produce one item while the consumer is consuming another one.**
- **With bounded-buffer size, consumer must wait if buffer is empty, whereas producer must wait if buffer is full.**
- **The buffer can be implemented using any IPC facility.**

Exp# 5a **Fibonacci & Prime Number**

Aim

To generate 25 fibonacci numbers and determine prime amongst them using pipe.

Algorithm

1. Declare a array to store fibonacci numbers
2. Decalre a array *pfd* with two elements for pipe descriptors.
3. Create pipe on *pfd* using pipe function call.
   a. If return value is -1 then stop
4. Using fork system call, create a child process.
5. Let the child process generate 25 fibonacci numbers and store them in a array.
6. Write the array onto pipe using write system call.
7. Block the parent till child completes using wait system call.
8. Store fibonacci nos. written by child from the pipe in an array using read system call
9. Inspect each element of the fibonacci array and check whether they are prime
   a. If prime then print the fibonacci term.
10. Stop

Result

Thus fibonacci numbers that are prime is determined using IPC pipe.