Interprocess Communications (IPC)

- Allow s a process communicate with another process
- Communications can be of two types:
  - Between related processes (parent and Child)
  - Between unrelated process(one or more different processes)

IPC can use:

- Pipes
- First In, First Out(FIFO)
- Message Queues
- Shared Memory
- Semaphores
- Signals

Pipes: Unnamed pipes

- Pipes are half-duplex(meaning they can only read or write from a process but only in one direction)
- Ex: ls -lr | more

- Named Pipes (FIFO)
- FIFO are also called First In first out and are half duplex

Example:

- Ashoks-MacBook-Pro:Documents ashokkafle$ mkfifo testpipe
- ls -ltr
- prw-r--r--   1 ashokkafle staff   0 May 1 14:40 testpipe
  Here the 'p' denotes that it is pipe, also if you try it on terminal this process remains dangling until the content are accessed using cat on next step
- Echo "Hello from Pipe World" > testpipe
- cat testpipe
  Hello From Pipe World

They are called FIFO because the first piece of data out is the first piece of data the other side reads

<u>Shared Memory:</u>

- They are full duplex, either process can read or write
- Most efficient because there is no syscall that goes to kernel unlike in other inter process connection
- Requires a program in order to use shared memory

- Any number of processes can read or write to the same shared segment
- Can query shared memory with the **ipcs** command
- Processes must manage shared memory
- Processes must protect(synchronize) shared memory benign written or race conditions will occur

```
Ashoks-MacBook-Pro:Documents ashokkafle$ ipcs
IPC status from <running system> as of Sun May  1 14:54:46 EDT 2022
T     ID     KEY        MODE       OWNER    GROUP
Message Queues:

T     ID     KEY        MODE       OWNER    GROUP
Shared Memory:

T     ID     KEY        MODE       OWNER    GROUP
Semaphores:
s  65536 0x0b046c90 --ra-ra-ra-     root    wheel
s  65537 0x0b043172 --ra-ra-ra- ashokkafle    staff
s  65538 0x25046b64 --ra-ra-ra- ashokkafle    staff
s  65539 0x0b045c6c --ra-ra-ra- ashokkafle    staff
s 262148 0x7b0401e0 --ra-ra-ra- ashokkafle    staff
s  65541 0xd1046b64 --ra-ra-ra- ashokkafle    staff
s  65542 0x5e046b64 --ra-ra-ra- ashokkafle    staff
s  65543 0x7b04043c --ra-ra-ra- ashokkafle    staff
s  65544 0x7b0404d1 --ra-ra-ra- ashokkafle    staff
s  65545 0x0b04f3b9 --ra-ra-ra- ashokkafle    staff
s  65546 0x7b040504 --ra-ra-ra- ashokkafle    staff
```

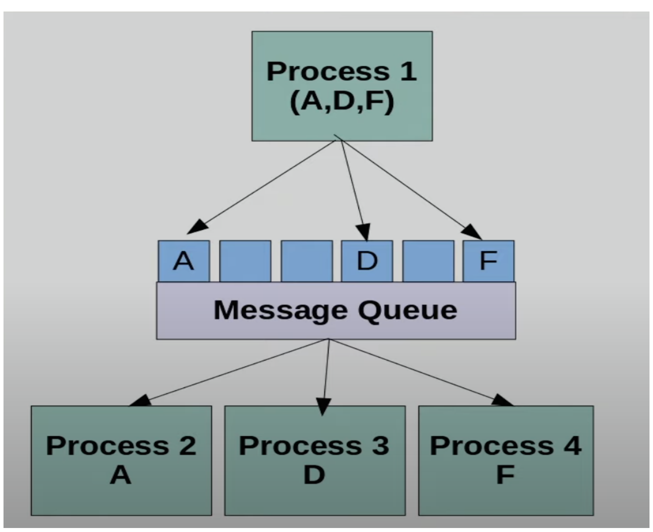
Race Condition:

- Let’s assume process 1 and process 2 are sharing a memory location
- Process1 wants to add 2 to the value that is currently in the shared memory
- Process 2 wants to add 3
- Both processes read the initial value of 0
- Process 1 adds 2 and changes the memory location to 2

- Process 2 adds 3 and changed the shared memory allocation to 3
- This is wrong, the final value should have been 5, this is a race condition
  The solution is synchronization, the mechanism, where which process grabs it first, it will lock and block the other process from accessing it. After the 1st process is done, it will release it and the other process starts accessing it.

Message Queues:
- They are created by a syscall
- They are managed by the kernel
- Once a message is read, it is deleted from the queue by the kernel
- Each read and write creates a syscall to the kernel
- The message queue helps eliminate the occurrences of race conditions but condition at the expense of permanence due to the syscall interrupt(trap)

- Message queues can have any number of read and write processes
- Usually there is just one write process but this is just convention not a rule
- Process 1 write A,D and F
- Process 2 reads and receives MEssage A, Message A is the removed
- Process 3 reads adn receives D, and then removed
- Process 4 reads and receives F, which is then removed
- Data remains in the queue until it is read and once read it’s deleted
-



Semaphores:

- They are used to protect critical/common regions of memory shared between multiple processes.
- Semaphores are the atomic structure of operating systems.
- Two types of semaphores:
    - Binary - only two states 0 and 1, unlocked/locked, or available unavailable etc
    - Counting semaphores – allow arbitrary resources counters

Semaphore Example (Linux):

- When a process allocates (accesses) a semaphore, it waits (blocks) the other processes to access the semaphore until the the first process issues a “release” indicating it has completed its operation
- The kernel then makes semaphores available for allocation

Signals:

- Notification of a n event occurrence
- Also known as trap or software interrupt
- Kill -l displays a full list of signals used by linux
- Signals can be generated by a user, a process or the kernel
- A process is supposed to be written to handle them
- Certain signals numeric 9 - 15 cannot be handled by the process, they will immediately cause the termination of the process and cannot be blocked, there are called “process-crash outs”
- Example CTRL-C at the terminal will a send SIGINT signal to the process which is currently running in the foreground

Credit: https://www.youtube.com/channel/UC05XpvbHZUQOfA6xk4dlmcw