SOFTENGER
Strive to Excel

Inter Process Communication

Unix

Process and Program in OS

SOFTENGER
Strive to Excel

- **Program** is a set of instructions or code in the form of computer programming language to perform specified task.
- **process** is a **program** in execution. It is the instance of a computer program that is being executed by one or many threads.
- Single program if executed by multiple users will be creating multiple processes.
- What is a process? A process is a program in execution.
- What is a program? A program is a file containing the information of a process and how to build it during run time. When you start execution of the program, it is loaded into RAM and starts executing.
- Process properties - PID, PPID, UID, EUID etc. which will be stored in kernel as structure called as process table.
- The kernel usually limits the process ID to 32767, which is configurable and can varies as per the OS.
- System calls related to process: fork(), Exec(), getpid(), getppid()

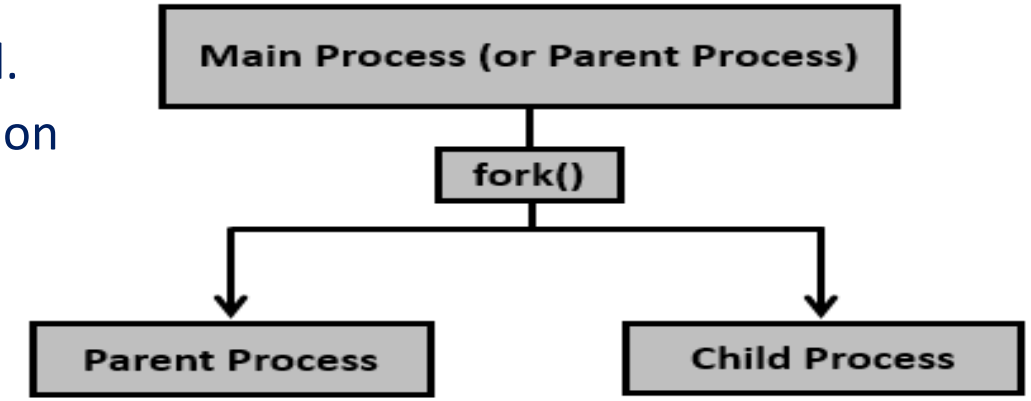
Program Vs Process

SOFTENGER
Strive to Excel

- A program is an executable file which contains a particular set of instructions written to complete the specific job on your computer
- A process is an execution of any specific program
- A process has a very limited lifespan
- A program is resided on disk.
- Various processes may be related to the same program

Process generation

- If you want to execute command **ps** to check process details , you will be entering the text **ps** on your Linux command prompt. what happened at back end?
  - The “C” library function system() executes a shell command.
  - the arguments passed to system() are commands executed on shell i.e. in our example it is ‘**ps**’
  - Unix programs are executed through a combination of two system calls called ***fork*** and ***exec***
  - The ***exec*** system call tells the kernel to execute another program. However, the kernel replaces the calling program with the new one being called.
  - Process creation is achieved through the ***fork()*** system call.
  - The newly created process is called the child process and the process that initiated it (or the process when execution is started) is called the parent process.



Fork() and exit() system call

SOFTENGER
Strive to Excel

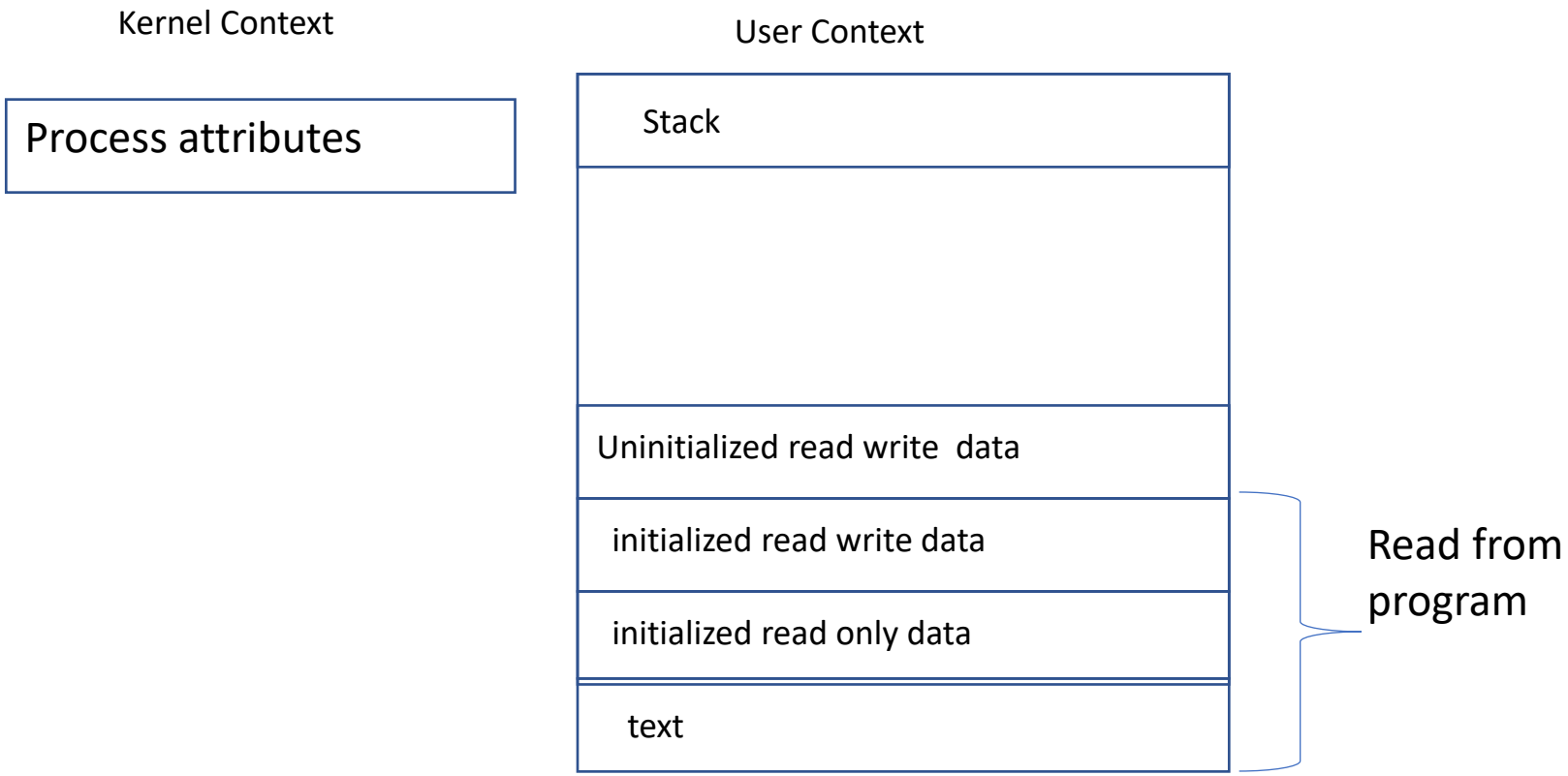
Syntax:

```
#include <sys/types.h>
#include <unistd.h>
pid_t fork(void);
```

- After this call, there are two processes, the existing one is called the parent process and the newly created one is called the child process.
- The fork() system call returns either of the three values −
  - Negative value to indicate an error, i.e., unsuccessful in creating the child process.
  - Returns a zero for child process.
  - Returns a positive value for the parent process. This value is the process ID of the newly created child process.
- A process can terminate in either of the two ways −
  - Abnormally, occurs on delivery of certain signals, say terminate signal.
  - Normally, using _exit() system call (or _Exit() system call) or exit() library function.

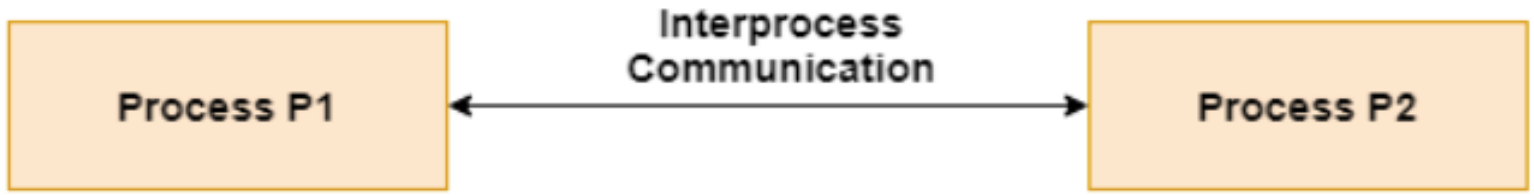
Process structure

- Process creation is achieved through the **fork() system call**.
- The newly created process is called the child process and the process that initiated it (or the process when execution is started) is called the parent process.



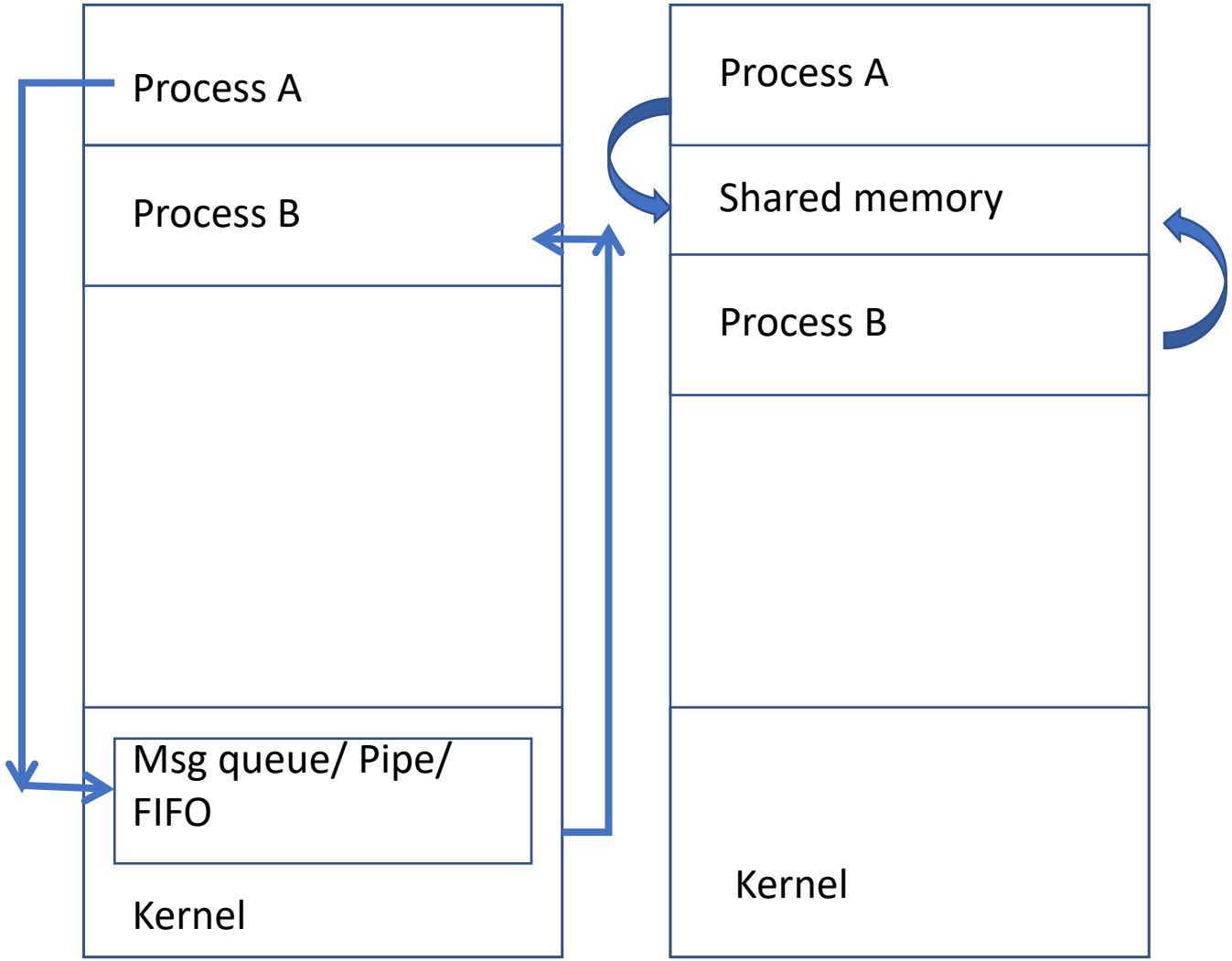
What is IPC

- Inter-process communication is the mechanism provided by the operating system that allows processes to communicate with each other. This communication could involve a process letting another process know that some event has occurred or the transferring of data from one process to another.
- Communication can be of two types –
- Between related processes initiating from only one process, such as parent and child processes.
- Between unrelated processes, or two or more different processes.



IPC

- Working together with multiple processes, require an interprocess communication (IPC) method which will allow them to exchange data along with various information. There are two primary models of interprocess communication:
  - shared memory and
  - message passing.



Important terms in IPC

SOFTENGER
Strive to Excel

- **Race Condition** – When two or more processes are accessing the shared resource and results depends who runs precisely when.
- **Critical section** – Part of program where shared resource is accessed.
- **Mutual exclusion** – When two or more processes are sharing resource and one process is in critical section other processes will be blocked till the shared resource become free.
- **Pipes** – Communication between two related processes. The mechanism is half duplex meaning the first process communicates with the second process. To achieve a full duplex i.e., for the second process to communicate with the first process another pipe is required.
- **FIFO** – Communication between two unrelated processes. It is also called as Named Pipes.
- **Message Queues** – Communication between two or more processes with full duplex capacity. The processes will communicate with each other by posting a message and retrieving it out of the message queue. Once retrieved, the message is no longer available in the queue.

Important terms in IPC contd.

SOFTENGER
Strive to Excel

- **Shared Memory** – Communication between two or more processes is achieved through a shared piece of memory among all processes. The shared memory needs to be protected from each other by synchronizing access to all the processes.
- **Semaphores** – Semaphores are meant for synchronizing access to multiple processes. When one process wants to access the memory (for reading or writing), it needs to be locked (or protected) and released when the access is removed. This needs to be repeated by all the processes to ensure consistency of the data.
- **Signals** – Signal is a mechanism to communication between multiple processes by way of signaling. This means a source process will send a signal (recognized by number) and the destination process will handle it accordingly.

Race Condition

SOFTENGER
Strive to Excel

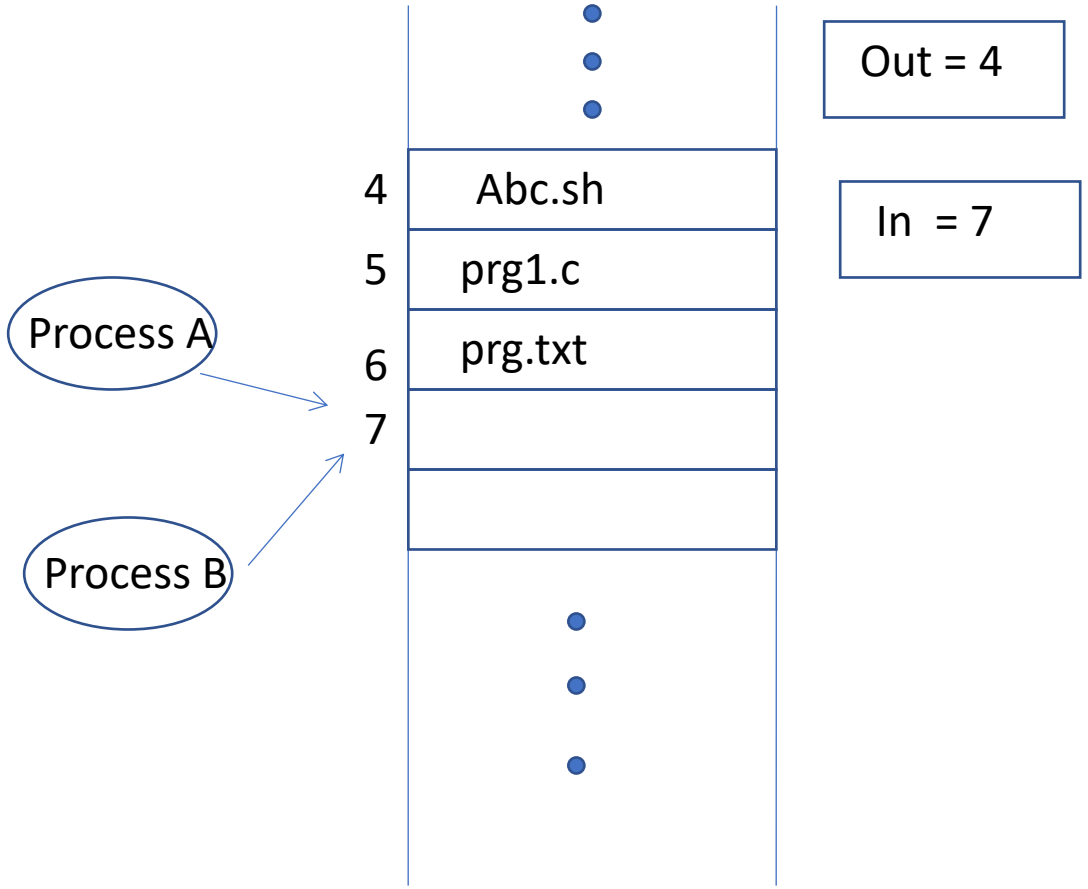
- In OS processes that are working together may share some common storage that each can read and write.
- Shared storage possibly is in main memory i.e. in kernel data structure or may be a shared file.
- When resource is shared synchronization of access to shared resource is important to ensure it should not lead to race condition.

Race condition Example

SOFTENGER
Strive to Excel

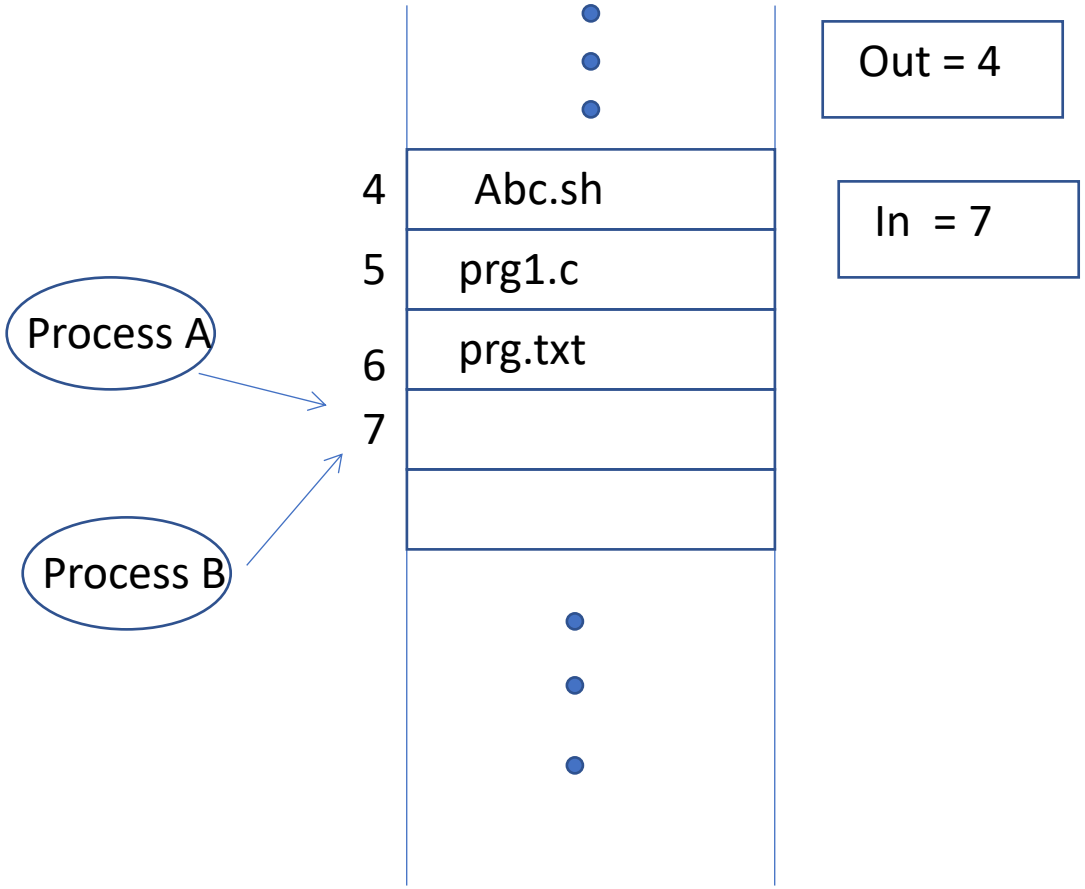
Example:

- There is a shared resource printer spooler directory.
- When process wants to print a file, it enters the file name in a special file **printer spooler directory.**
- **Printer daemon periodically** to check if so are any files to be printed and if so removes their names from the directory.
- Imagine that spooler directory are having 0 to n slots each one to hold name of file to print.
- At certain instant, slots 1 to 3 are empty. Files are already printed and 4 to 6 are full
- There are two shared variables out = next file to be printed and in = next free slot in spooler directory



Race condition Example contd.

- **Process A** and **process B** want to queue a file for printing.
- **Process A** read in and store value 7 in a local variable called next-free-slot. Just then the CPU interrupt occurs and CPU decides that **process A** had run long enough so it switches to **Process B**. **Process B** also reads next-free-slot 7 and stores name of its file in slot 7 and updates it 8 then goes off to work other tasks.
- Eventually **process A** runs again starting from the place where it left last time. Find next-free-slot 7 and stores name of its file in slot 7 and updates it 8. printer spooler daemons not noticed anything wrong but **process B** will hang around the printers room with no output called as Race condition.



Race condition can be avoided?

SOFTENGER
Strive to Excel

- When two or more processes are accessing the shared resource and results depends who runs precisely when.
- To avoid race condition:
  - No 2 or more processes should be in CS
  - No assumptions about speed and number of CPUs
  - No Process outside CS should block access to CS
  - To ensure no process to wait forever to access CS.

Synchronization in IPC

SOFTENGER
Strive to Excel

- synchronization is a necessary part of interprocess communication. It is either provided by the interprocess control mechanism or handled by the communicating processes.
- **Semaphore** A semaphore is a variable that controls the access to a common resource by multiple processes. The two types of semaphores are binary semaphores and counting semaphores.
- **Mutual Exclusion** Mutual exclusion requires that only one process thread can enter the critical section at a time. This is useful for synchronization and also prevents race conditions.
- **Spinlock** This is a type of lock. The processes trying to acquire this lock wait in a loop while checking if the lock is available or not. This is known as busy waiting because the process is not doing any useful operation even though it is active.

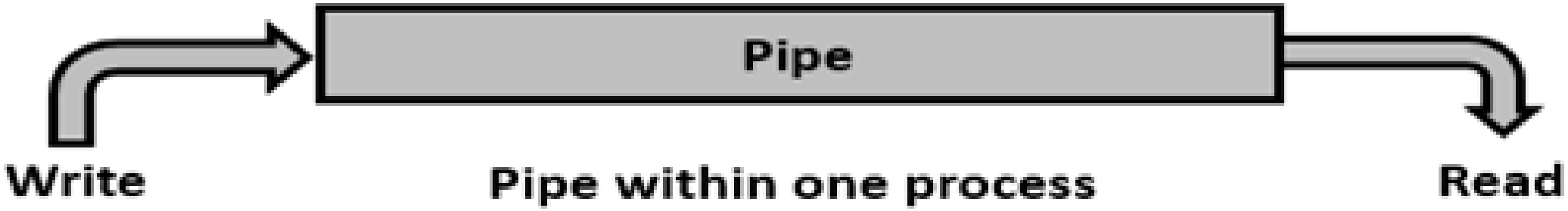
IPC channels

SOFTENGER
Strive to Excel

- **Pipe / FIFOs** A pipe is a data channel that is unidirectional. Two pipes can be used to create a two-way data channel between two processes. This uses standard input and output methods. Pipes are used in all POSIX systems as well as Windows operating systems.
- **Socket** The socket is the endpoint for sending or receiving data in a network. This is true for data sent between processes on the same computer or data sent between different computers on the same network. Most of the operating systems use sockets for interprocess communication.
- **Message Queue** Multiple processes can read and write data to the message queue. Messages are stored in the queue until their recipient retrieves them. Message queues are quite useful for interprocess communication and are used by most operating systems.
- **Shared Memory** Shared memory is the memory that can be simultaneously accessed by multiple processes. This is done so that the processes can communicate with each other. All POSIX systems, as well as Windows operating systems use shared memory.

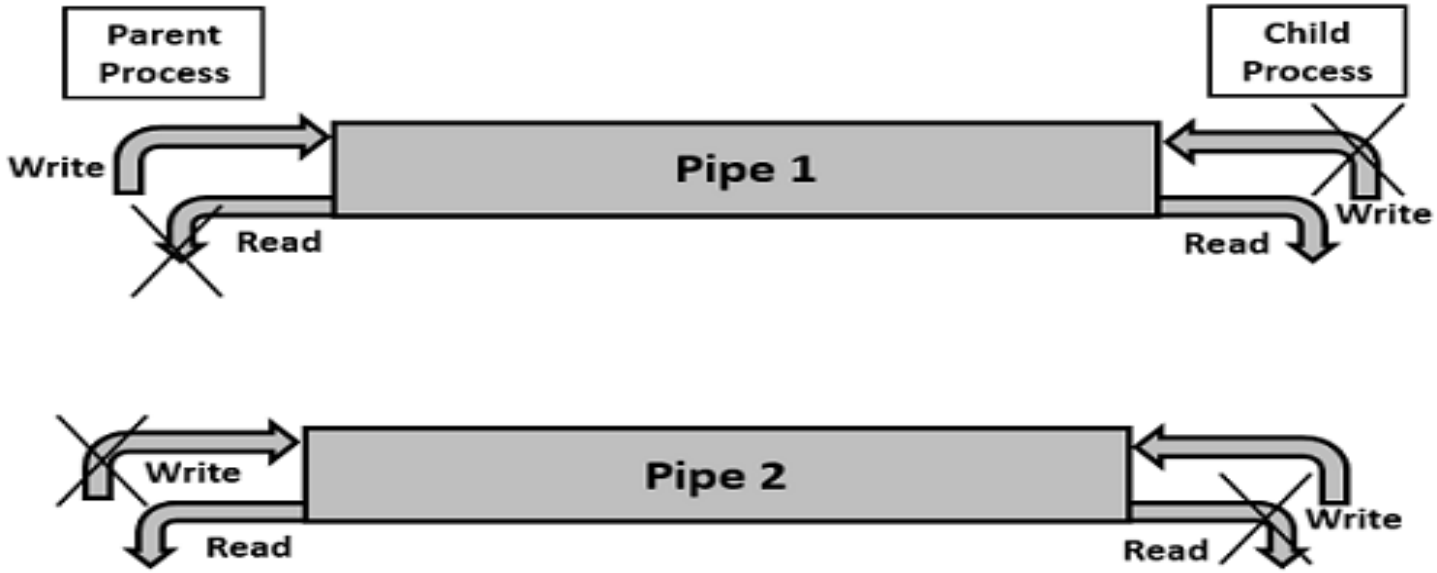
PIPES

- communication medium between two or more related or interrelated processes (parent and child process).



Two way IPC using pipes

- **Step 1** – Create two pipes. First one is for the parent to write and child to read, say as pipe1. Second one is for the child to write and parent to read, say as pipe2.
- **Step 2** – Create a child process.
- **Step 3** – Close unwanted ends as only one end is needed for each communication.
- **Step 4** – Close unwanted ends in the parent process, read end of pipe1 and write end of pipe2.
- **Step 5** – Close the unwanted ends in the child process, write end of pipe1 and read end of pipe2.
- **Step 6** – Perform the communication as required.



IPC Named PIPES (FIFO)

SOFTENGER
Strive to Excel

- use pipes for unrelated process communication, called as named pipes or FIFO
- Named Pipes are using file (ordinary file, device file or FIFO) which will be used as media for IPC.
- Int mknod() will create such special file which is used as media for IPC.

Limitations for pipes

SOFTENGER
Strive to Excel

- Simple pipes can be used for IPC between only 2 related processes (processes for which parent process ID is same.
- Can be used only for IPC in between 2 processes.
- Vanished once the process execution is completed. Data is not available for future requirement.
- It is used for unidirectional IPC. For bio-directional communication, it required #2 pipes to be created.
- It carries the message in the form of stream of bytes. Message interpretation is not done.

IPC using Message Queues

SOFTENGER
Strive to Excel

- A **message queue** is a linked list of messages stored within the kernel and identified by a message queue identifier. A new queue is created or an existing queue opened by **msgget()**
- New messages are added to the end of a queue by **msgsnd()**. Every message has a positive long integer type field, a non-negative length, and the actual data bytes (corresponding to the length), all of which are specified to msgsnd() when the message is added to a queue.
- Messages are fetched from a queue by **msgrcv()**. We don't have to fetch the messages in a first-in, first-out order. Instead, we can fetch messages based on their type field.
- The sending process places a message (via some (OS) message-passing module) onto a queue which can be read by another process.
- Each message is given an identification or type so that processes can select the appropriate message. Process must share a common key (**ftok key)**in order to gain access to the queue in the first place.

System calls used for message queues:

SOFTENGER
Strive to Excel

- **ftok()**: is use to generate a unique key.
- **msgget()**: either returns the message queue identifier for a newly created message queue or returns the identifiers for a queue which exists with the same key value.
- **msgsnd()**: Data is placed on to a message queue by calling msgsnd().
- **msgrcv()**: messages are retrieved from a queue. **msgctl()**: It performs various operations on a queue. Generally it is use to destroy message queue.

Message Queue

SOFTENGER
Strive to Excel

- Kernel maintains structure for each IPC :
- Struct ipc_perm {
  Ushort uid;
  Ushort gid;
  Ushort cuid
  Ushort cgid
  Ushort mod
  Key key_t
  }

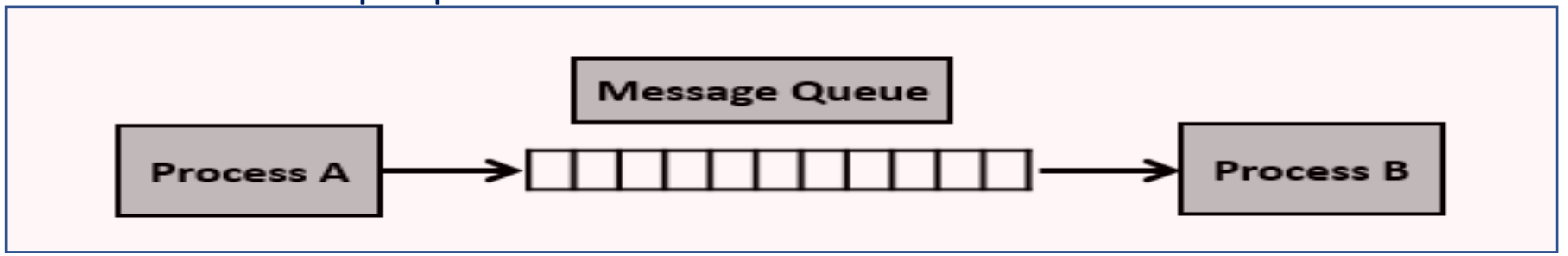
Message queue structure

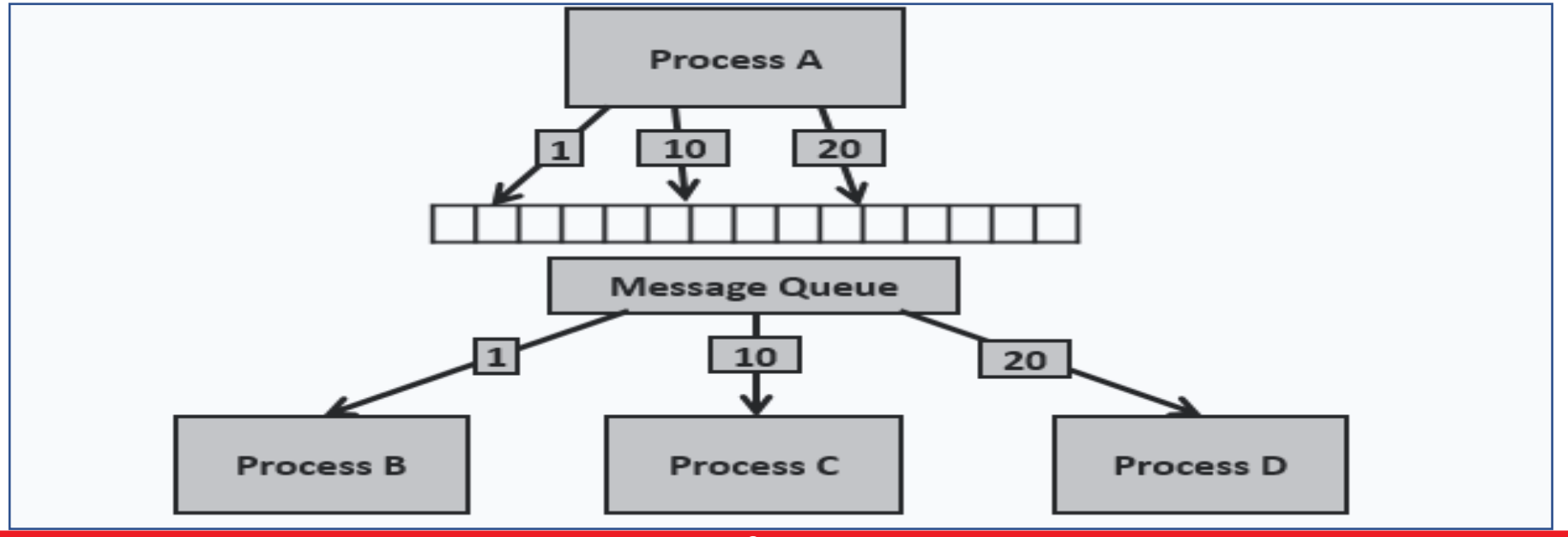
- In Pipes, once the message is received by a process it would be no longer available for any other process.
- For every message queue kernel maintains structure as below:

```
Struct msgqid_ds
{
    struct msgqid_perm;
    struct msg *msg_first;
    struct msg *msg_last;
    ushort msg_cbytes;
    ushort msg_qbytes;
    ushort msg_qnum;
    ushort msg_lspid;
    ushort msg_lrpid;
    time_t msg_stime;
    time_t msg_rtime;
    time_t msg_ctime;
}
```

IPC using Message queues

- Communication using message queues can happen in the following ways –
  - Writing into the pipe by one process and reading from the pipe by another process. Reading can be done with multiple processes as well





IPC Semaphores

SOFTENGER
Strive to Excel

- Race condition – multiple processes are accessing the same set of resources (Critical section). Leads to race condition.
- **Semaphores** are synchronization primitive to access shared resource.
- Basically semaphores are classified into two types –
  - **Binary Semaphores** – Only two states 0 & 1, i.e., locked/unlocked or available/unavailable, Mutex implementation.
  - **Counting Semaphores** – Semaphores which allow arbitrary resource count are called counting semaphores.

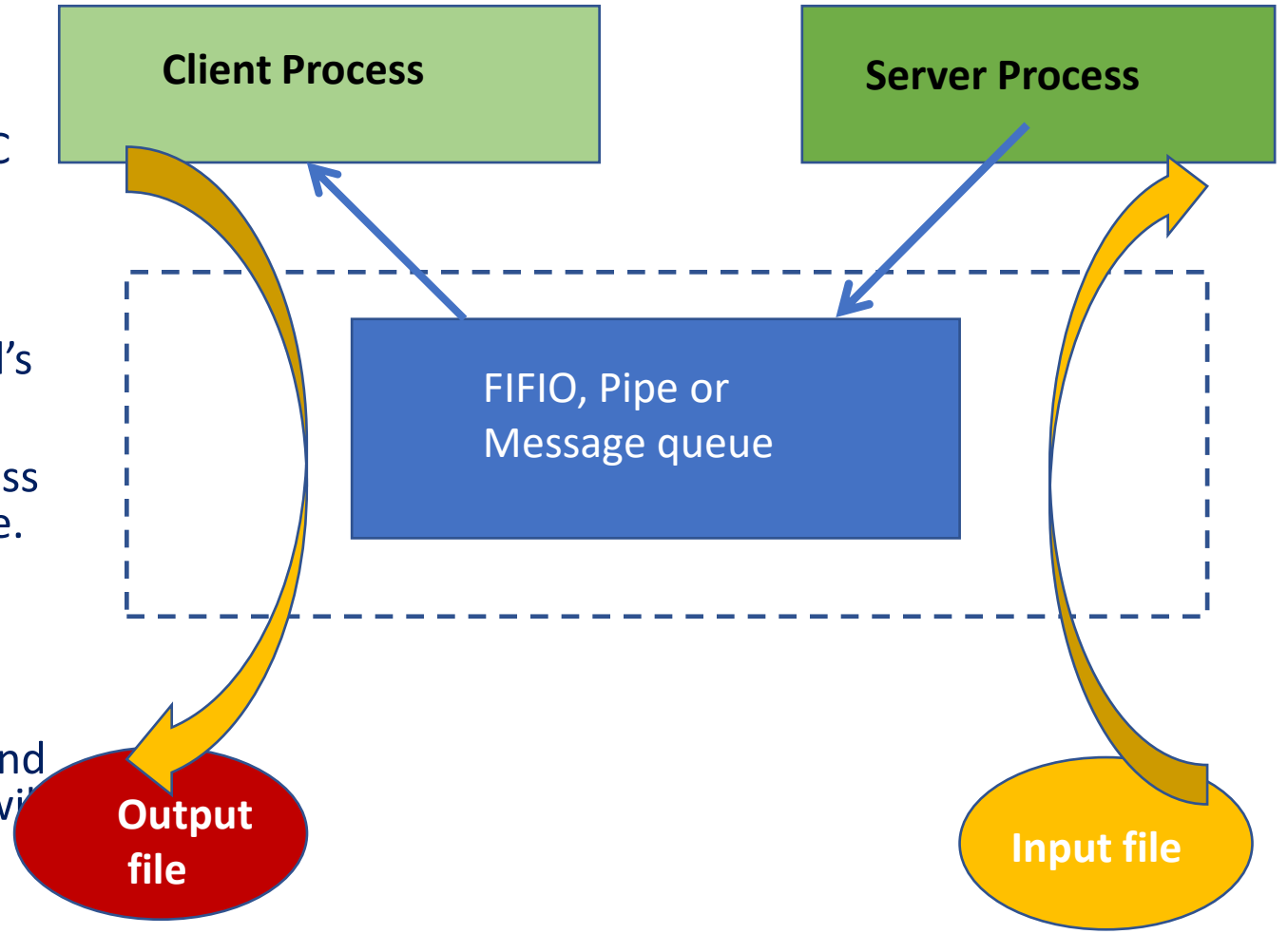
IPC - Semaphores

SOFTENGER
Strive to Excel

- System call sem_op(), indicates the operation that needs to be performed for shared resource−
  - If sem_op is –ve, allocate or obtain resources. Blocks the calling process until enough resources have been freed by other processes, so that this process can allocate.
  - If sem_op is zero, the calling process waits or sleeps until semaphore value reaches 0.
  - If sem_op is +ve, release resources.

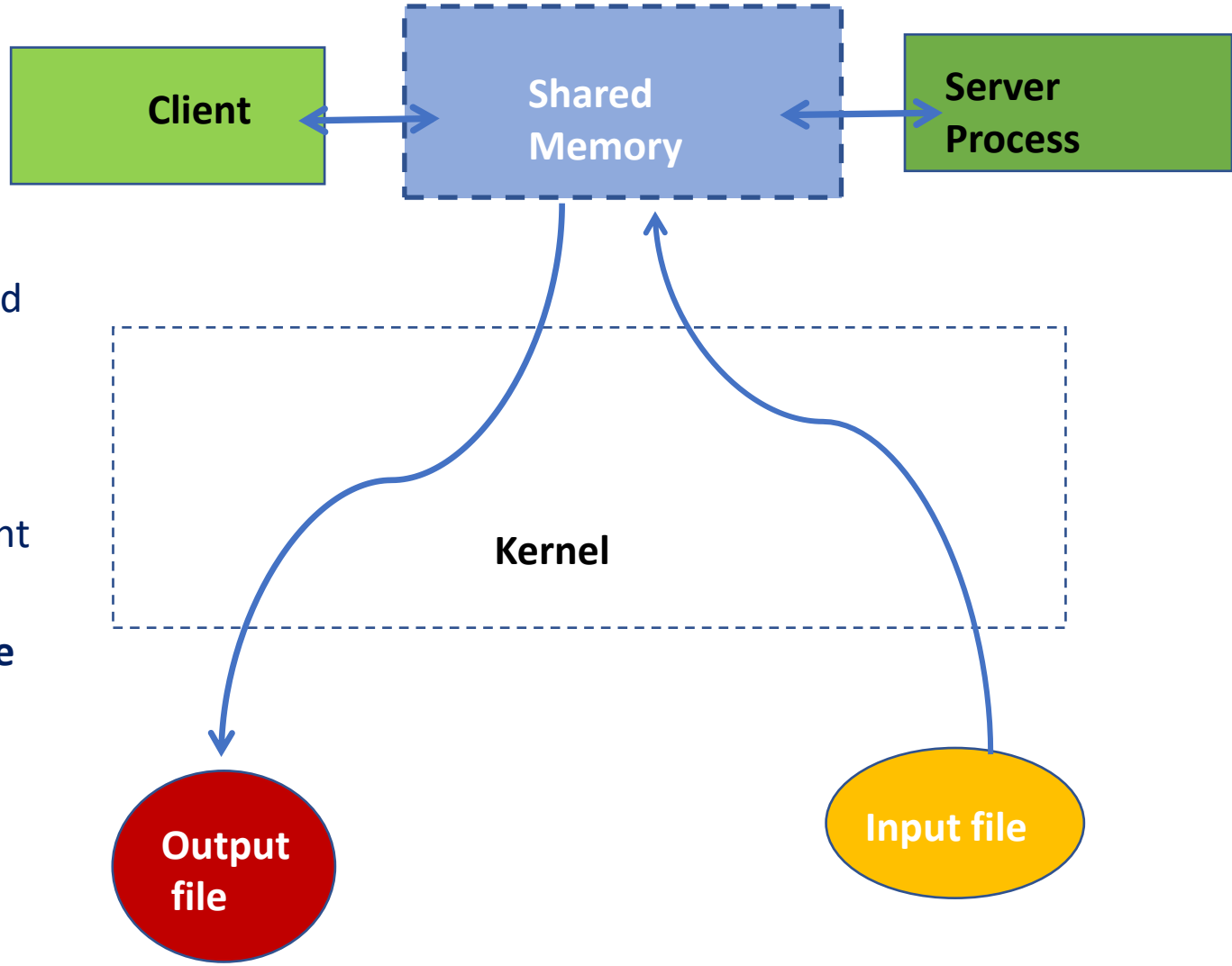
IPC WITHOUT Shared Memory

- Data is read by server process using read system call from input file to one of internal block buffer.
- Then server process writes data using any IPC channel i.e. pipes, named pipes or Message queue in kernel IPC buffer.
- Client process reads data from IPC channel, again requiring data to be copied from kernel's IPC buffer into client process buffer.
- Finally data will be copied from client's process buffer through write system call to output file.
- **A total of four copies of data are required (2 read and 2 write)**
- With Shared Memory the data is only copied twice – from input file into shared memory and from shared memory to the output file. We wi see this in next slide.



IPC using shared memory

- IPC using shared memory includes following steps:
  - Process get access to shared memory using a semaphore
  - Server reads from input file into shared memory segment. The address to read into the second argument to the read system call, points into shared memory.
  - When the read is complete server notify the client, again using a semaphore.
  - The client writes data from shared memory segment to the output file.
  - **With Shared Memory the data is only copied twice – from input file into shared memory and from shared memory to the output file.**



How IPC is done with Shared memory?

SOFTENGER
Strive to Excel

- Process create the shared memory segment or use an already created shared memory segment (shmget())
- Other processes get attached the process to the already created shared memory segment (shmat())
- Data required to communicate for multiple processes will be stored in shared memory segment created in the RAM can be accessed by all processes which are attached to this segment.
- If process need not do IPC will get detached the process from the already attached shared memory segment using system call (shmdt())
- Control operations (remove shared memory segment) on the shared memory segment using system call (shmctl())

System calls in Shared Memory

SOFTENGER
Strive to Excel

- ***ftok()****: is use to generate a unique key can be used in all system calls of shared memory IPC.*
- ***shmget()****: int shmget(key_t,size_tsize,intshmflg); create shared memory ssegment. upon successful completion, shmget() returns an identifier for the shared memory segment.*
- ***shmat()****: Before you can use a shared memory segment, you have to attach process to it using shmat(). void *shmat(int shmid ,void *shmaddr ,int shmflg);*
  shmid is shared memory id. shmaddr specifies specific address to use but we should set it to zero and OS will automatically choose the address.
- ***shmdt()****: When you're done with the shared memory segment, your program should detach itself from it using shmdt(). int shmdt(void *shmaddr);*
- ***shmctl()****: when you detach from shared memory,it is not destroyed. So, to destroy shmctl() is used. shmctl(int shmid,IPC_RMID,NULL);*

Sockets

SOFTENGER
Strive to Excel

- Sockets provide point-to-point, two-way communication between two processes on single system and between different systems.
- This method is mostly used to communicate over a network between a client and a server. It allows for a standard connection which is computer and OS independent.
- IPC sockets enable channel-based communication for processes on the same physical device (*host*), whereas network sockets enable this kind of IPC for processes that can run on different hosts, thereby bringing networking into play.
- Network sockets need support from an underlying protocol such as TCP (Transmission Control Protocol) or the lower-level UDP (User Datagram Protocol).

Review Questions

SOFTENGER
Strive to Excel

1. What is IPC? What are the IPC channels ?
2. What is the difference between process and Program?
3. How the process is generated?
4. What is critical section?
5. What is Race Condition and how it can be avoided?
6. What is the difference between PIPES and FIFOs?
7. What is multiplexing of messages using message queue?
8. Where the shared memory segment is created?
9. What is the use of semaphore?
10. Which IPC channel is used to perform IPC in between processes from different systems.