Ch#3 "Process"

Q. What is Process?

=> A process is a program in execution including the current values of variables, registers, of program counter.

=> Process is the "activity"

=> A process is an active entity with a program counter specifying the next instruction to execute and a set of associated resources.

=> Two processes may be associated with the same program.

=> A process is controlled and scheduled by the OS and execute in sequence.

Memory layout of a process / Attributes of process / Components of Proc

=> The memory layout of a process is typically divided into multiple sections:

=> These sections or attributes include:

Text section:

- Contains the executable code
- The text and data sections are fixed, as their sizes do not change during program run time.

Data section:
- Contains the global variables
- Can grow and shrink during program execution

Heap Section:
- memory that is dynamically executed during program run time.

Stack section:
- Temporary data storage when invoking functions such as functioning address, local variables and function parameters.

Process States or Life cycle:

State:
- Process state describes the nature of the current activity of that process.
- As Process executes, it changes state.

=> when a process runs, it goes through many states.
=> Different OS have different states.

1) New State:
=> This is the first state of the process life cycle.
=> when process creation is taking place, the process is in new state.
=> Stored process on secondary storage.

2) Ready State:
=> After being loaded into the main memory and ready for execution.
=> In this state the process is placed in the queue of processes which are waiting for the processor OS CPU allocation
=> Many processes may be in ready state in multiprogramming environment
=> when the process is in the creation is in a new state and when the process gets created is in the ready state.

3) Running State:
=> After being alloted the CPU to the process for instruction executions, the process state changes from ready to running state.

4) Block or Wait State:

⇒ when the process is executing the instructions the process might require carrying out a few tasks which might not require CPU.

⇒ if the process requires performing I/O task or the process needs some resources which are already acquired by other processes, during such conditions process is brought back into the main memory and the state is changed to blocking or wait for the state.

⇒ Process is placed in the queue of processes that are in waiting or block state in the main memory.

⇒ The process advances to the ready state after the I/O operation is completed or the resource becomes available

5) Terminated or Completed:-

⇒ When the entire set of instructions is executed and the process is completed.

The process is changed to terminated completed state.

⇒ During this state the PCBs of the process is also deleted.

⇒ It is possible that there are multiple processes present in the main memory at the same time

6) Suspend ready:

⇒ if a process with a higher priority needs to be executed while the main memory is full, the process goes from ready to suspend ready state.

⇒ Moving a lower-priority process from the ready state to the suspended ready state frees up space in the ready state for a higher-priority process.

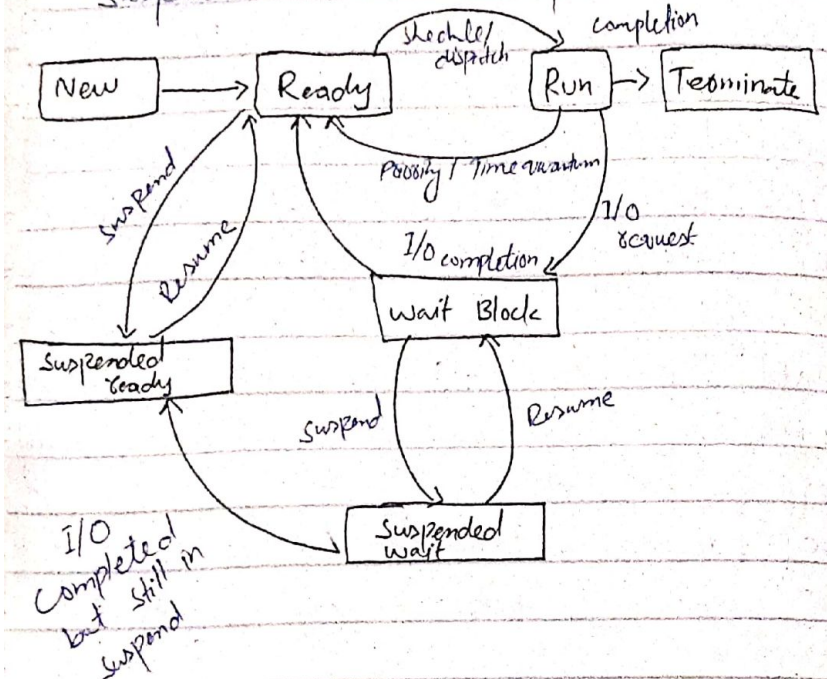
⇒ until the main memory becomes available, the process stays in the suspended-ready state.

⇒ The process is brought to its ready state when the main memory becomes accessible.

⇒ So whenever the main memory is full the process which is in the ready state is swapped out from main memory to secondary memory.

7) Suspended wait or Suspended Blocked

=> Whenever the process that is in waiting for state or block state in main memory gets to swap out to secondary memory due to main memory being completely full, the process state is changed to suspended wait or Suspended blocked state.



State transition in Process:-

New → Ready:-

- Admitted to ready queue and can be considered by CPU scheduler.

Ready → Running:

CPU scheduler chooses that process to execute next according to some scheduling algorithm.

Running → Ready:

Process has used its current time slice.

Running → Blocked:

Process is waiting for I/O operations.

Blocked → Ready:-

The event has occured for which the process was waiting.

Process Block Control:-

=> Each process is represented in the OS by a process control block.

=> Also called a task control block.

=> It is a data structure that contains many pieces of information associated with a specific process.

=> Every process has its own PCB.

=> The information that is in PCB is:

Process state:
- Contains new, ready, running, halted state so on.

Program Counter:
- Contains the address of the next instruction to be executed.
- The value of the PC is incremented automatically refers to the next instruction when each instruction is executed.

Process number:
- The process id is an identifier for each system process.
- Each process is given a unique number.

Priority:
- Contain, the priority of all the processes that has the greatest priority.

Registers:
- General purpose register
- When interrupt occur process state is stored.

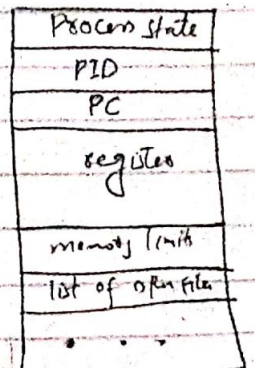
CPU Scheduling info:
- Contains pointers to scheduling queues
- Priority
- other scheduling parameters

Memory management:
=> Contain information
- Base register
- Limit register
- Page table
- Segment table

Accounting:
=> Amount of CPU and real-time used
=> Time limits
=>
=> account numbers
=> Job or process number

I/O Status information:
- List of I/O devices allocated for process
- A list of open files

Note:
- Each process PCB is stored in the main memory.
- List in the linked list



Interprocess Communication

⇒ Process executing in the OS may be either independent or dependent process.

Independent Process	Dependent process
○ A process that does not share data with other processes executing in the system & do not affect by any other process is called independent process.	○ A Process is said to be cooperating if it can affect or be affected by any other processes or share data with other processes.

Advantages of process cooperation:

⇒ information sharing
⇒ Computation Speedup
⇒ modularity
⇒ Convenience.

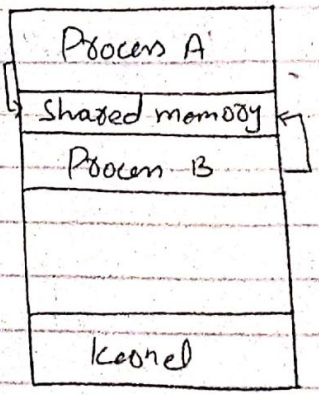
IPC allows two processes to share, send and receive data with each other.

IPC models

⇒ There are two models of IPC
- ○ Shared memory model
- ○ Message passing model

Shared - memory model:

⇒ A region of memory is shared by the cooperating processes is established.
⇒ Typically, a shared memory region is resides in the address space
⇒ Processes can exchange information by reading and writing data to the shared region.



- The Process may cooperate by sharing data including variables, memory, and files.
- The OS provide data integrity

Example:

⇒ Producer - Consumer Problem
⇒ Compiler, Assembler & Loader
⇒ Client & Server model

Buffers used in Shared-memory model.

=> Two types of buffers can be used:

Unbounded buffer	Bounded buffer
=> No limit on the size of the buffer	=> Limit on the size of the buffer
=> Consumer need to be wait for new items	=> Consumer must wait if the buffer is empty, producer must wait if the buffer is full.

Advantages	Dis-advantages
=> Memory Communication is safer.	=> Synchronization
=> Implicit Communication (no shared data)	=> memory protection that need to be addressed
=> Low overhead & easier	=> Complex to scale well

IPC in Message-Passing Systems

=> Message passing model allows multiple processes to read and write data to the message queue without being connected to each other.

=> Messages are stored on the queue until their recipient retrieves them.

=> Message queues are quite useful for interprocess communication and are used by most Operating systems.

=> Allow processes to communicate and to synchronize their actions without sharing the same address space.

=> Useful in distributed environment where the communicating processes may reside on different computers connected by a network.

Example:

- Internet chat program could be designed so that chat participants communicate with one another exchanging messages.

Operations Provided by MPS:-

⇒ A message-passing facility provides at least two operations:

- send(message)
- receive(message)

Message Size:-

⇒ Message sent by a process can be either:-

- Fixed size
- Variable size

Fixed-size:

- if only fixed-sized messages can be sent, the system-level implementation is straightforward.
- makes the task of programming more difficult.

Variable size:

- Requires a more complex-level implementation
- Programming task becomes simpler.

Communication link:-

Communication link must exist b/w them:

- Physical link
- Logical link

Issues:

- Naming
  - direct communication
  - indirect communication
- Synchronization
  - Blocking/non-blocking sender
  - Blocking/non-blocking receiver
- Buffering
  - Zero capacity
  - Automatic buffering
    - bounded capacity
    - unbounded capacity

(i) Naming:

- it defines how a one process can send message to another process and how a process can receive a message from another.

Direct communication	Indirect communication
○ Each process that wants to communicate must explicitly name recipient or sender of the communication.	○ message sends and received from mail box or ports.

* Direct communication:

=> In this scheme, the send() and receive() primitives are defined as:-

=> Send (P, message)
- Send a message to process P.

=> receive (Q, message)
- Receive a message from process Q.

Communication link properties:-

=> A communication link in this scheme has the following properties:-

- Automatically established
- Associated with exactly two processes
- Bw each pair of processes, there exists exactly one link.

Symmetry	Asymmetry
o Both sender and the receiver process must name the other to communicate.	o only the sender names the recipient; the recipient is not required to name the sender.
o Send (P, message)	o receive (id, message)

Cons:

- limited modularity

* Indirect Communication:

- the messages are sent to and received from mailboxes, or ports.
- mailbox act as mediator
- Each mailbox has unique identification number.
- Send (A, message) ← mailbox
- receive (A, message)

Communication link properties:

- Link is established if both members of the pair have a shared mailbox.
- may be associated with more than two processes.
- Different links may exist between each pair of communicating processes, with each link corresponding to one mailbox.

Problem:

- if User are User processes P1, P2, P3.....

How to owner come?

.....

=> A mailbox may be owned by:
- A process.
- OS

Synchronization:

Blocking Sender:

- Process is blocked until the message is received by the receiver.

Non-blocking Sender:

- Sender don't care whether the message is received or not.

- **Blocking receiver:**
  Receiver blocks until a message is available

- **Nonblocking receiver:**
  Receiver retrieves either a valid message or a null.

Buffering:-

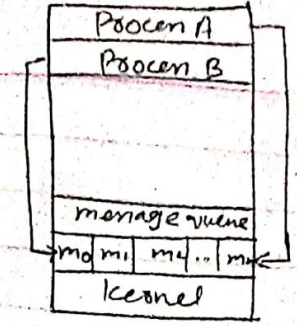
⇒ Determines what is the size of buffer.

Zero capacity: No buffering

Bounded Capacity: finite size of the buffer

Unbounded capacity: Infinite size of the buffer

Advantages	Disadvantages
o Explicit communication	o High-message passing overs head.
o No automatic caching	o Complex to program



Why a process is called active and a program is call passive entity?

⇒ A program will be inactive unless it is being used. hence it is considered as passive entity and resides in the secondary memory.

⇒ A process gets activated when a program is executed. therefore it is called active entity and it is loaded in the main memory and it also require some resources.

(Ch #3) PP-SQ (2014–2020) (Total 19+Repeat) (II)

- Why do we call a program passive entity and a process active entity? (Repeat)
- What does it mean to preempt a process? (Repeat)
- What is the ready queue? (Repeat)
- What are the states of process? (Repeat)
- What is process synchronization?
- What is the PCB? (Repeat)
- What are the capacities of queues in message passing system? (Repeat)
- What is the purpose of the PCB?
- what do you know about process & programs? What is Zombie process? What is Orphan process?
- How parent and child share the address space after fork()?
- What is the degree of multi-programming?
- What is cascading termination?

Long

- What is interprocess communication? Explain its different communication strategies?
- Explain the process control block?
- Explain process creation & termination?