Processes

On Linux

Prayas Mohanty (Red Hat Certified Instructor)

Red Hat Certification ID: 100-005-594

Objective

- What is a Process
- What is (PCB) Process Control Block
- Explore Various Attributes kept in PCB
- What is a Process State
- What is Process Scheduling
- What is Context switching

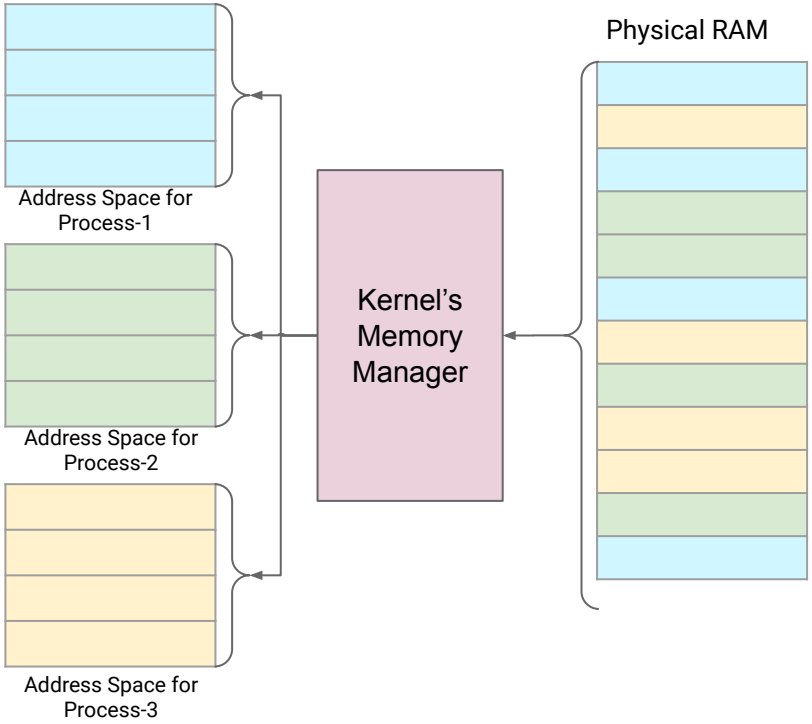
Prerequisite of Participants

- Having Knowledge on Linux Platform
- Having familiarity with Linux Command line
- Basic Knowledge on vi Editor
- Proper Knowledge on C programming
- Basic Knowledge on File Handling in C

What is a Process

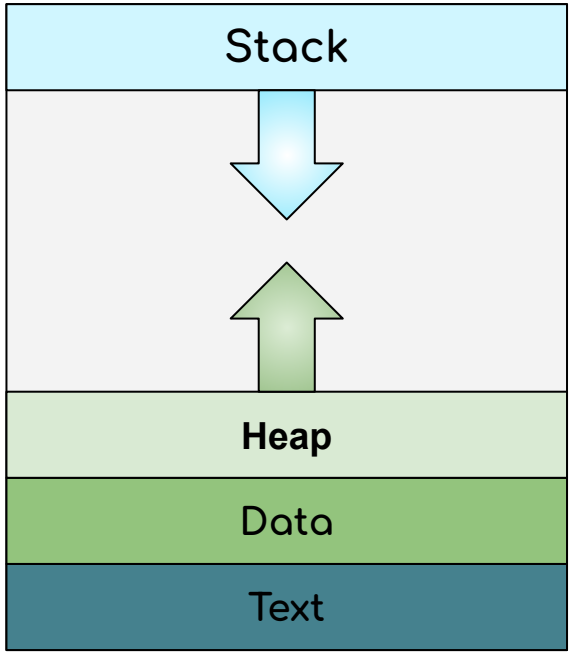
- A Program does nothing unless its instructions are executed by a CPU.
- In simple, a process is a program under execution.
    - A computer program itself is just a passive collection of instructions stored in disks.
    - While a process is the actual execution of those instructions in real time.
- But as per UNIX standards, specifically IEEE Std 1003.1, 2004 Edition: A process is defined as “an address space with one or more threads executing within that address space, and the required system resources for those threads”.

Address Space



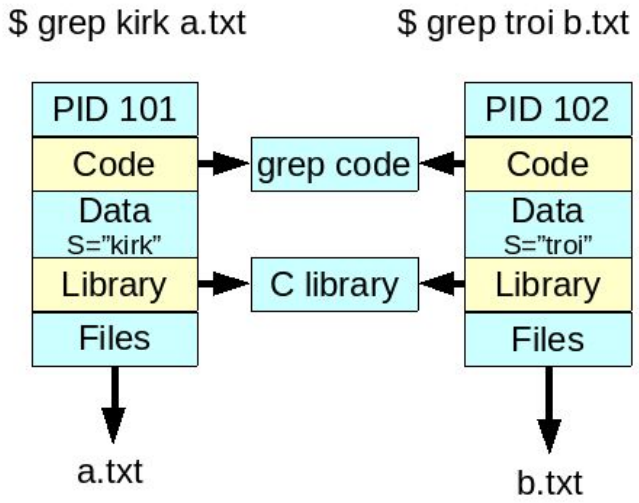
- In computing, an process address space defines a range of discrete addresses in memory, each of which is associated with the specific process.
- Address spaces are created by combining enough uniquely identified qualifiers to make an address unambiguous within the address space.
- An address space usually provides (or allows) a partitioning to several regions according to the mathematical structure it has.

Required System Resource in the address space



- Stack:
  - Temporary data such as function parameters, local variables and return addresses.
  - The stack grows from high addresses towards lower address.
- Heap:
  - Dynamically allocated (malloc) by the program during runtime.
  - The heap grows from low addresses towards higher addresses.
- Data:
  - Statically (known at compile time) global variables and data structures.
- Text:
  - The program executable machine instructions.

How to Identify the Address space (PID)



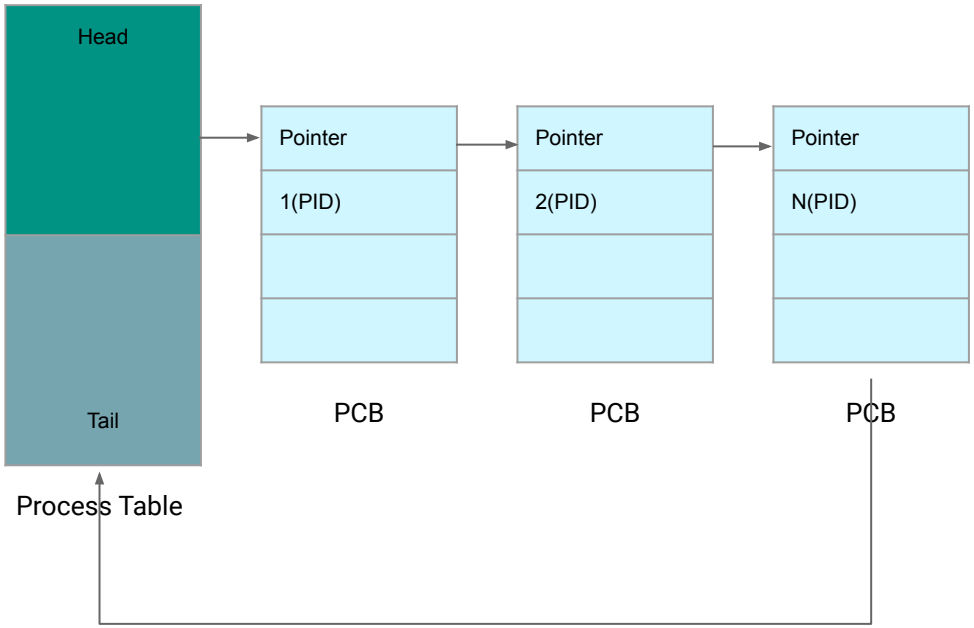
- Each process / Address Space for the process is allocated a unique number, called a process identifier or PID.
- This is usually a positive integer between 2 and 32,768.
- PID is an index to process table who holds the information about all running Process.

The Process Table (Currently allocated address spaces)

- The process table is a data structure maintained by the operating system to facilitate context switching, scheduling, and some other activities.
- Each entry in the table, often called a **Process Control Block**, contains information about a process such as process name, state, PID and many more information discussed later.
- Linux provide a “ps” command to retrieve required information from the process table.
  - ps
  - ps aux
  - ps -ef

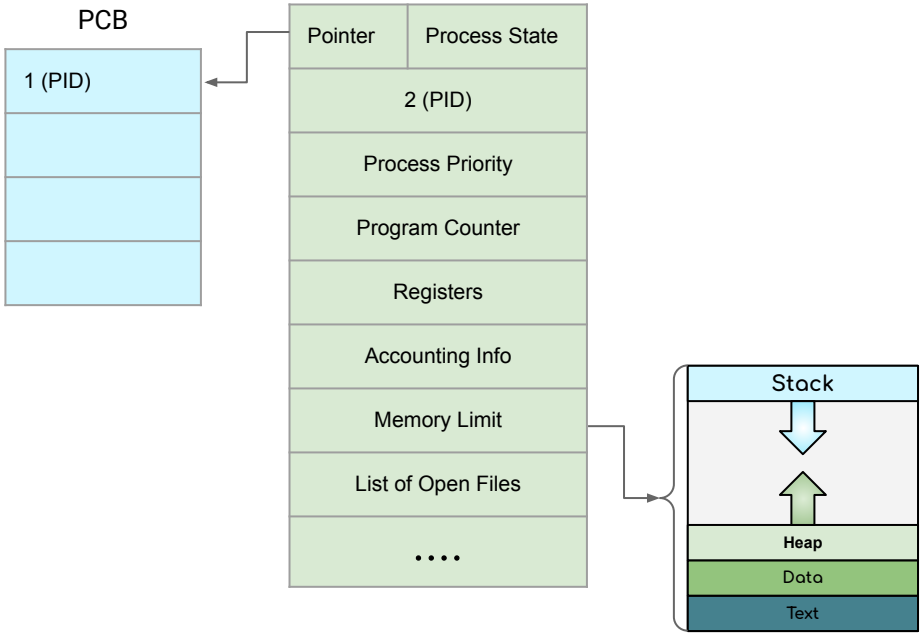
```
[prayas@prayashost-isdac-net ~]$ ps au
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
gdm         1323  0.0  0.0 371256  6408 tty1     Ssl+ 10:51   0:00 /usr/libexec
gdm         1326  0.0  0.3 310200 59876 tty1     Sl+  10:51   0:01 /usr/libexec
gdm         1534  0.0  0.0 467052 13992 tty1     Sl+  10:51   0:00 /usr/libexec
prayas      1920  0.0  0.0 371256  6520 tty2     Ssl+ 10:51   0:00 /usr/libexec
prayas      1922  1.7  0.5 401696 95508 tty2     Sl+  10:51   1:34 /usr/libexec
prayas      1939  0.0  0.0 467140 14316 tty2     Sl+  10:51   0:00 /usr/libexec
prayas      8573  0.0  0.0 235204 14200 pts/0    Ss   12:14   0:00 bash
prayas      8905  0.0  0.0 227032  3572 pts/0    R+   12:19   0:00 ps au
```

Process Control Block (PCB)



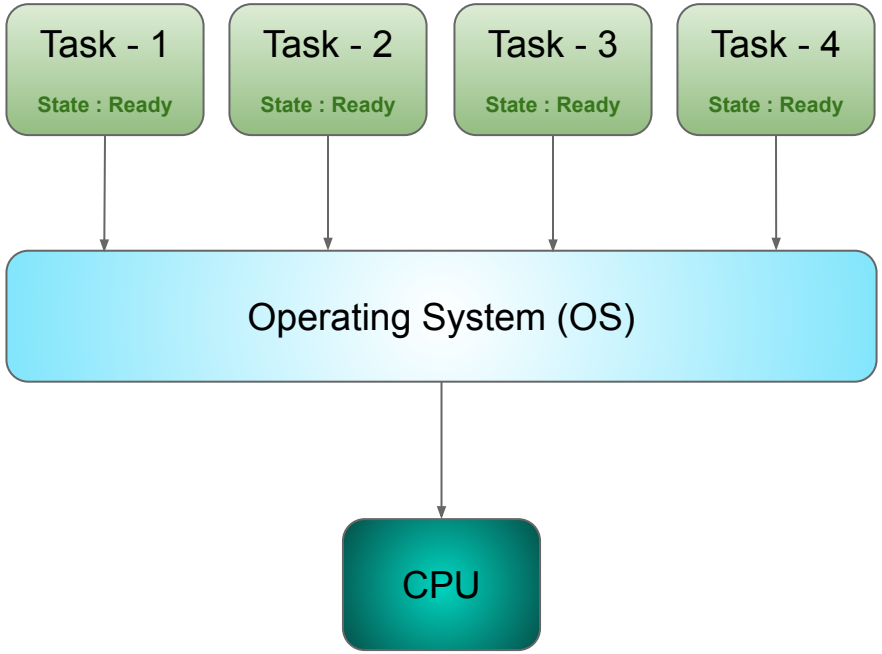
- Every process is represented in the operating system by a process control block also called a task control block.
- A process control block (PCB) is a data structure used in the OS code to represent one process.
- It contains many pieces of information and it contains all the attributes of a process.
- Being many processes get executed in a system, the combination of all PCBs is called the process table.
- PCBs are maintained in a linked list manner.

Attributes of a process inside PCB



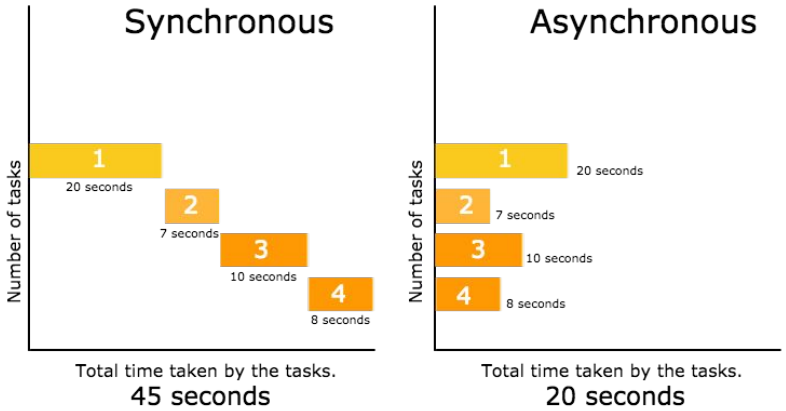
- PCBs are maintained in a linked list manner. By that we know how many processes are executing in a system and we can know the current status of the processes.
- PCB Ensure Protection by which any other processes should not get into the workspace of the concern process and similarly the same process should not get into other processes' workspace.
- PCB gets updated while the process is executing (While the state of the process changes).

Multitasking (many process in memory)



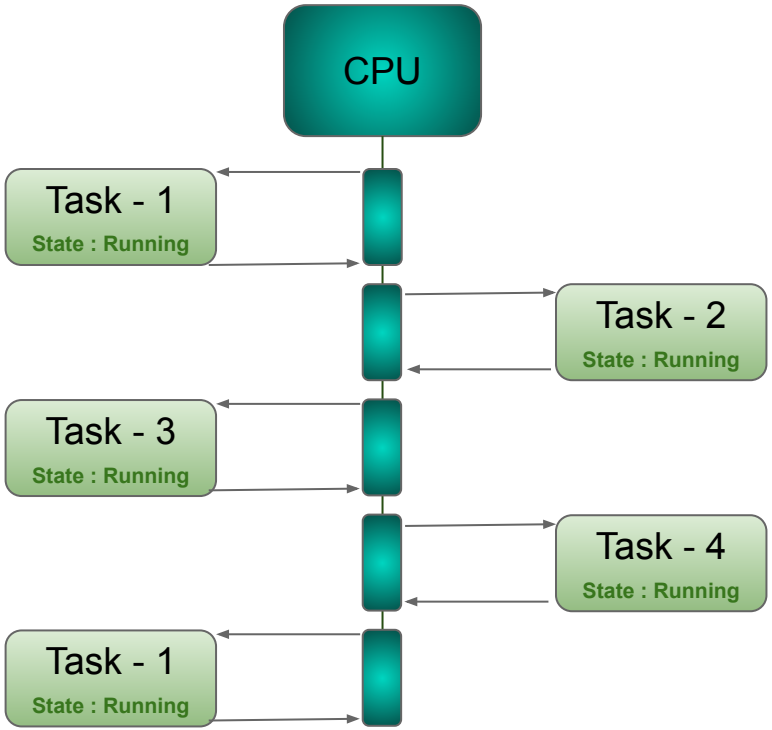
- Multitasking (aka time sharing) is a logical extension of multiprogramming where a timer is set to cause an interrupt at a regular time interval called time slice.
- At beginning, several jobs are kept in memory at the same time. Initially, all jobs are in the ready state.
- One of the ready jobs is selected to execute on the CPU and changes state from ready to running.
- The running job is replaced if the job requests I/O or if the job is interrupted by the timer. This way, the running job is given a time slice of execution than cannot be exceeded.

Synchronous and asynchronous events



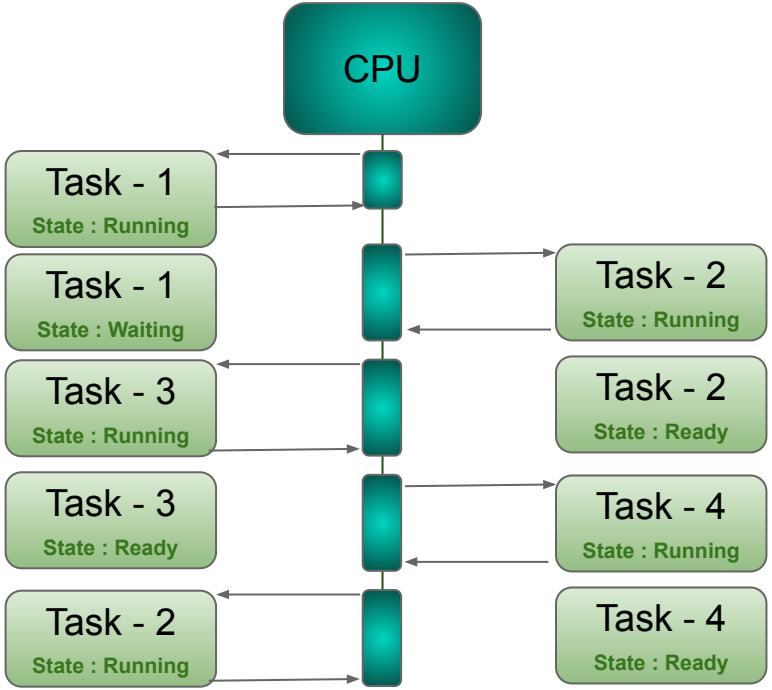
- Synchronous simply means that all events are occurring in a certain time order that can be predicted.
  - A certain event would always follow another and they can't be interchanged.
- Asynchronous is the opposite of synchronous.
  - In asynchronous processes, there is no time order.

Context Switching



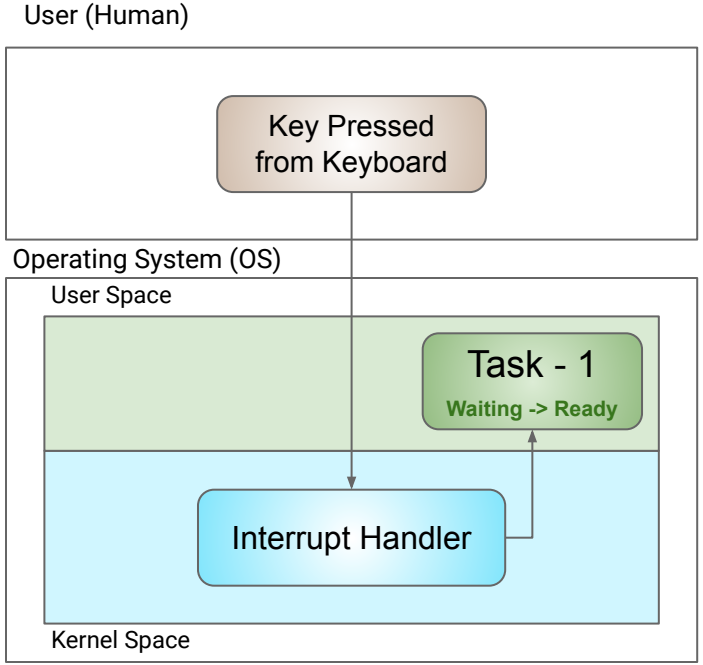
- Context switching switches the current process with the new process requiring the CPU to finish its tasks.
- At any point in time, the values of all the registers in the CPU defines the CPU context.
- When a new process takeover the processor time, the values in all registers must be saved to memory before the new process can use the registers.
- Similarly before resuming execution of the new process, the values of all registers must be restored using the values saved to memory earlier.

What about a Task waiting for keyboard Input



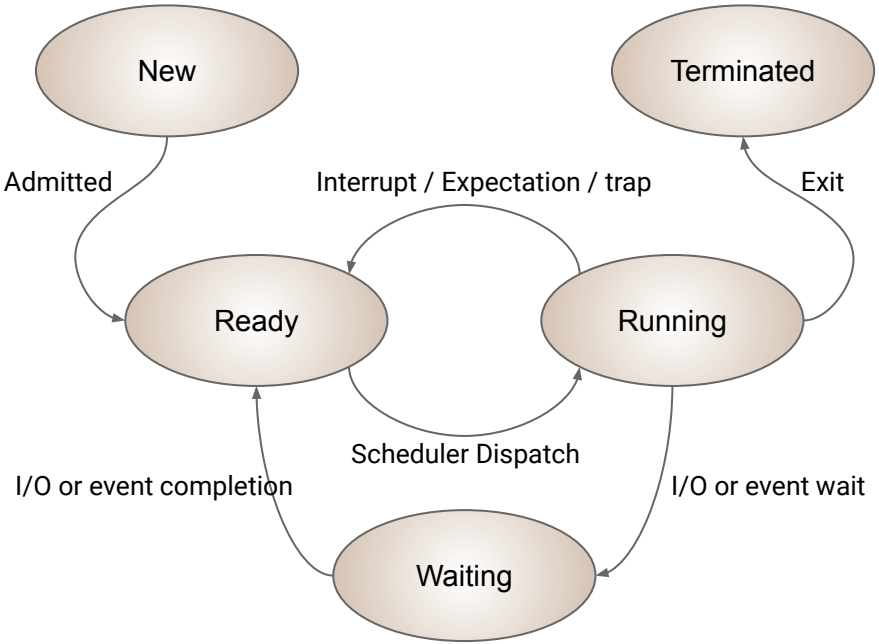
- Humans are very slow compared to the CPU.
  - No matter how fast you are, the CPU will be able to execute a huge number of instructions between every key-press you make.
- Is it possible to make the CPU do something useful while waiting for user input?
  - CPU switch context to next task and
  - The current task change it state to waiting instated of Ready.
  - When the next time slot come for the current task, it got ignored as the state is waiting but the slots are divided among ready task only.
- To bring back the Waiting task to ready state after the input got collected interrupt is used.

Interrupt driven input



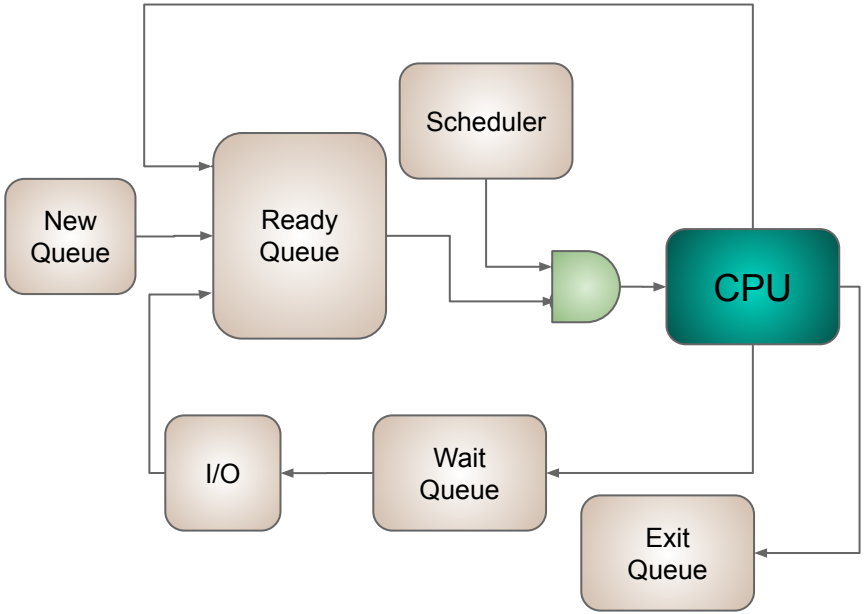
- Interrupts are used to notify the system when an I/O request is completed.
- Interrupts are generated by hardware devices outside the CPU at arbitrary times with respect to the CPU clock signals and are therefore considered to be asynchronous.
- Key-presses on a keyboard might happen at any time.
  - Even if a program is run multiple times with the same input data, the timing of the key presses will most likely vary.
- Read and write requests to disk is similar to key presses.
  - The disk controller is external to the executing process and the timing of a disk operation might vary even if the same program is executed several times.

Process State



- **NEW**- The process is being created.
- **READY**- The process is waiting to be assigned to a processor.
- **RUNNING**- Instructions are being executed.
- **WAITING**- The process is waiting for some event to occur(such as an I/O completion or reception of a signal).
- **TERMINATED**- The process has finished execution.

Process Scheduling



- On a single-processor computer, only one process can run at a time, while others wait their turn.
    - These turns, known as time slices, are quite short and give the impression that programs are running at the same time.
- When there are two or more runnable processes then it is decided by the Operating system which one to use the current time slice and which one will use the next and so on.
- The module of the OS responsible for this allocation is referred to as Process Scheduler.

Process Scheduling in Linux

Scheduling Objectives

- Be Fair
- Maximize throughput
- Maximize number of users receiving acceptable response times.
- Be predictable
- Balance resource use
- Avoid indefinite postponement
- Enforce Priorities
- Give preference to processes holding key resources
- Give better service to processes that have desirable behaviour patterns
- Degrade gracefully under heavy loads

- The Linux kernel uses a process scheduler to decide which process will receive the next time slice emphasizing on process priority.
- programs that perform short bursts of work and pause for input are considered better behaved than those that hog the processor by continually calculating some value.
- Well-behaved programs are termed nice programs, and this “niceness” get measured by a value from -20 to +19 but default to 0.
- To set or reset the nice value of any process, command “nice” and “renice” get used.
- To view the nice value (NI) use “ps -l”.

Summary

- What is a Process
- What is (PCB) Process Control Block
- Explore Various Attributes kept in PCB
- What is a Process State
- What is Process Scheduling
- What is Context switching

Any Questions ?

Thank you!