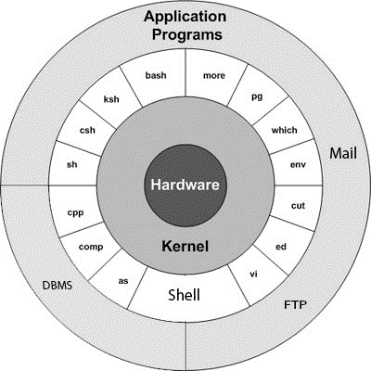
**Operating System**

***What is OS:*** An operating system is an interface between hardware and software which helps us to interact with hardware efficiently. Like Windows, MAC, Linux, Unix, Android, Unix etc.

***What is Kernel:*** Kernel is the center/core component of an operating system that manages operations of computer hardware. The kernel acts as a bridge between application and data processing at hardware level using Inter Process Communication and System-call. For example, Unix is the kernel of MAC, Linux and Android.

***Different functionalities of OS:***Process-management, Memory-management, File-system, Device-management, Network-management, Security etc.

***John-Von-Neumann architecture:*** John von Neuman architecture is also known as the Stored-program concept. Where instruction data and program data are stored in the same memory. This design is still used in most computers produced today.

***History-Of-OS:***There are 4 stages of the revolution, **the first generation (1940’s to early 1950’s)** when electronic computers were first introduced & they were without any OS. All programming was done in absolute machine language during this generation computers were used to solve simple mathematical calculations.

**The second generation (1955-1965)** was the time when the first operating system was introduced. These OS were called batch processing systems because the data was submitted in the form of a group & These new machines were called mainframe computers. They were used by professionals in a large computer room.

**The third generation (1965-1980)** is the time when the multiprogramming concept was introduced. It was a major part in the development of the OS.

**The fourth generation (1980-present-day)** introduced the personal computer gradually to MS-DOS, Windows-95, Windows-98, Windows-XP, Windows-7, Macintosh. Then came quite advanced OS as well like Network-OS, RTOS (IOT), Parallel-processing etc.

***Uniprogramming vs Multiprogramming:* Uni-programming** is at a times in RAM only one program can be present where **Multi-programming** stands for more than one programs simultaneously can be present on RAM.

***Multiprogramming vs Multitasking:* Multi-programming** stands for more than one programs can be present in RAM & **Multi-tasking** is more than one process can execute simultaneously however at any point of time a single CPU can execute only one process because of massive processing speed it looks like that all the process are running at a glance.

***Preemptive vs not-preemptive:*** **Preemptive** is, let say a process is running in CPU and another high-level process or Interrupt want to execute stopping the running process and placing the ready process in order to execute is called **Preemption**, Which OS are provided this feature are called Preemptive Operating system.

In which OS this feature is not available those operating systems are called Non-Preemptive Operating systems.

***User-mode vs Kernel-mode or Supervisor-mode:*** When we start a process in an operating system, the operating system creates a **PSW (Process status word. It is a special DS made up of a bunch of bits)** associated with the newly started process in order to store metadata of this process. Within this PSW a special bit known as **Mode-bit** if it is **0** represents this process in user mode right now if it is **1** then processes in kernel mode right now. If any process needs to use **System call** then the process must be in kernel mode and in kernel mode no process can’t be preempted.

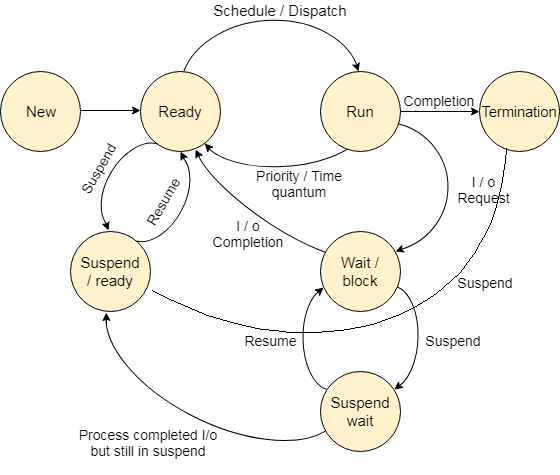
***Fork system call***:System call **fork()** is used to create child processes. It takes no arguments and returns a **process ID** ofthe child if no has been created then return **0**. The purpose of fork() is to create a new process, which becomes the child process of the caller. After a new child process is created, both processes will execute the next instruction following the fork() system call. Variables of each generated process are unique or independent. Total no of child processes including parent for n **fork()** call is 2^n. **fork()** may return -ve values if for any reason the child process is not created.

***Program vs process:*** A program is a compiled quantity stored on secondary storage. Where a process is when a program loaded into the **RAM** gets allocated resources like CPU, Network, Registers etc. this state is called process.

***Swap space:*** Virtual memory is a combination of **RAM** & **Secondary-storage** that a running process can use. Swap space is a portion of virtual memory that is on the hard disk used when RAM is full in order to store suspended processes.

***Process control block (PCB):*** **PCB** is a data structure, while creating a process the operating system performs several pieces of operation. As the OS supports multiprogramming it needs to keep track of all the processes. For this task PCB is used to track the process's execution status like Program-counter, Stack-pointer, Status of opened files, Scheduling algorithm etc.

***Life cycle of a process:***



***LTS:*** LTS stands for **Long term scheduler** it is also known as **Job scheduler** it is used to select which job should be sent to the **Ready state** from the **New state**.

***STS:*** STS stands for **Short term scheduler** it is also known as **CPU scheduler** it is used to select jobs from **Ready state** which to be transferred to the **Running state**, then by **Dispatcher** the selected jobs get transferred to the **running state**.

***MTS:* Midterm scheduler** is used to transfer jobs from the **Ready state** to **Suspend state**.

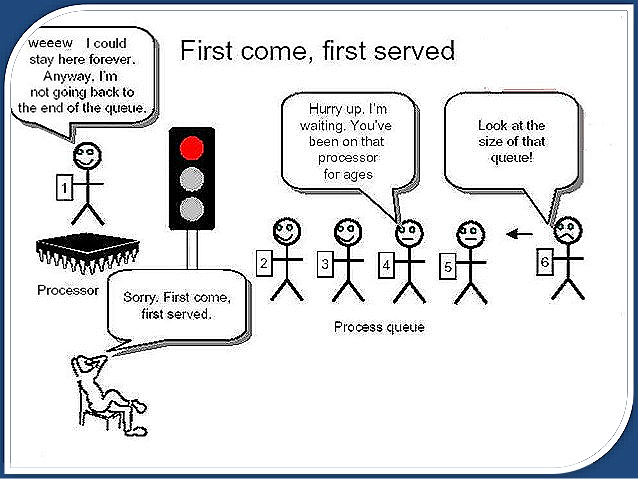
***I/O bound vs CPU bound processes:*** Which process spends tons of time in wait/block state in order to do I/O these processes are called I/O bound processes, however it uses CPU but comparatively less than CPU bound process. Inversely which process spends lots of time in the running state these processes are called CPU bound processes.

***Context switching:*** In the Operating System, there are cases when we have to bring back the process that is in the running state to some other state like ready state or wait/block state etc.

A context switching is a process that involves switching of the CPU from one process or task to another. In this phenomenon, the execution of the process that is present in the running state is suspended by the kernel and another process that is present in the ready state wants to execute itself due high-priority, time-quantum or for any scheduling algorithm. It is one of the essential features of the multitasking operating system. But the context switching process involved a number of steps that needed to be followed. We can't directly switch a process from the running state to any other state. We have to save the context of that process. If we are not saving the context of any process P then after some time, when the process P comes into the CPU for execution again, then the process will start executing from starting. But in reality, it should continue from that point where it left the CPU in its previous execution. So, the context of the process should be saved before putting any other process in the running state.

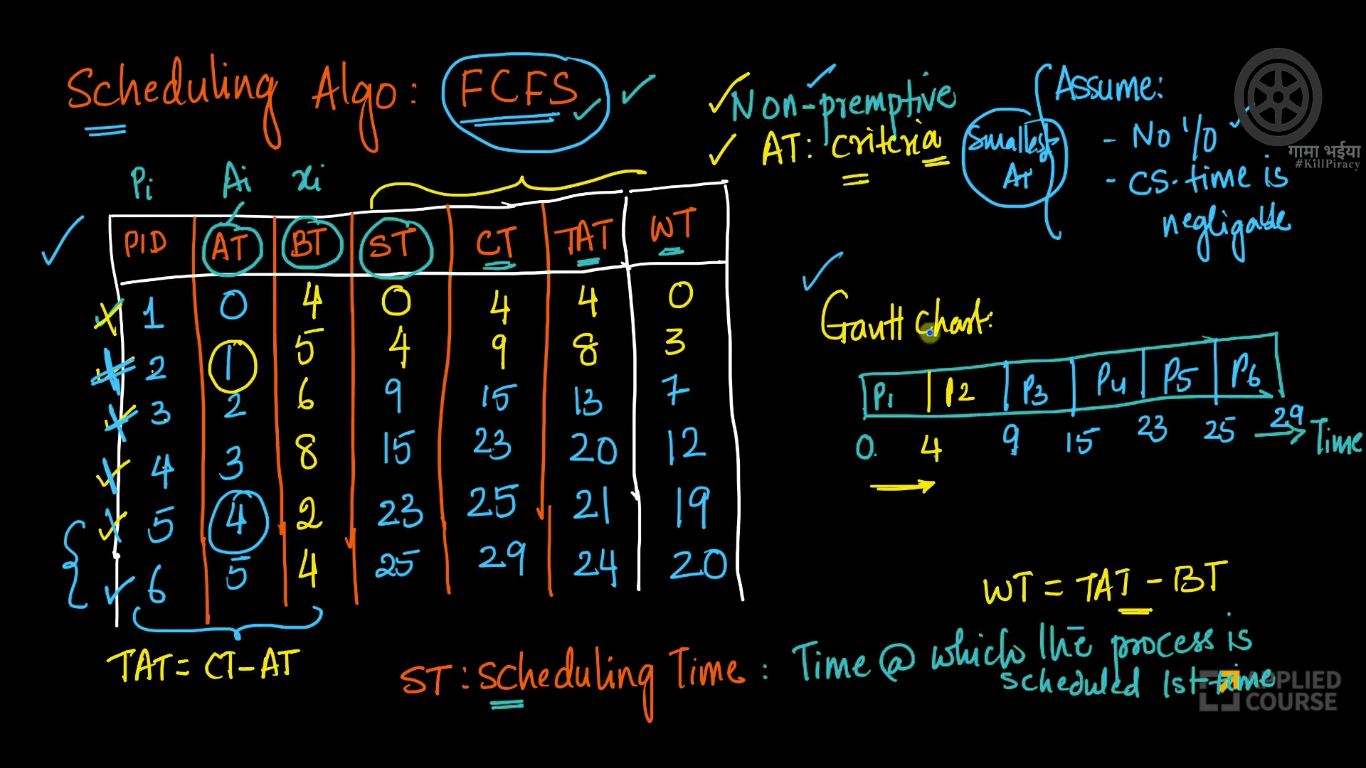
***Different types of time:***

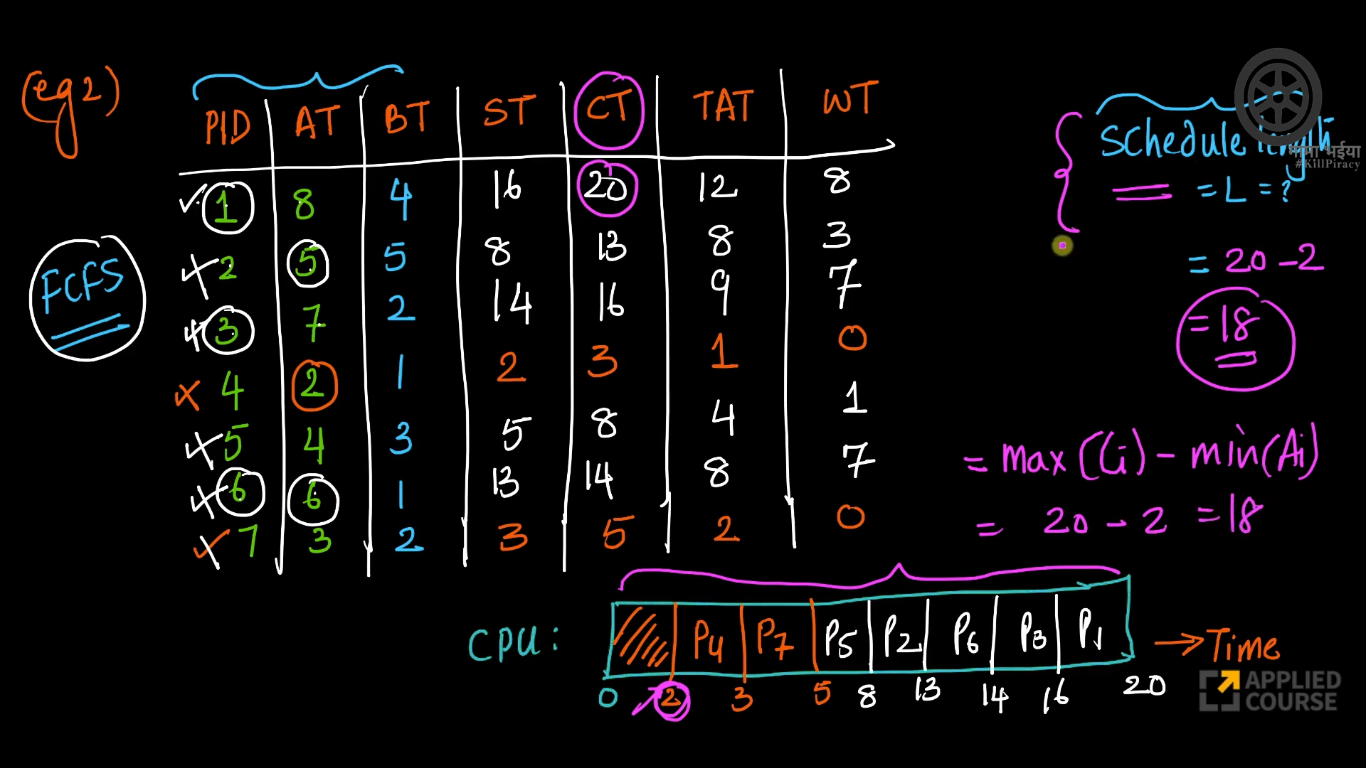
1. Throughput **→** Number of processes completed per amount of time. Unit can be a Second, Min, Hour etc.
2. Arrival/Submission/Admission time → The time at which a process enters in Ready state.
3. Waiting time → The total time spent by the process in the ready state waiting for the CPU. How much time is waiting in the ready state.
4. Burst/ Service time → The total time spent by a process for it’s execution on the CPU. Can be two type CPU, I/O burst time.
5. Completion time → Time at which a process completed its execution.
6. Turnaround time → The total amount of time spent by a process in the ready state for the first time to its competition time.
7. Response time → The amount of time spent between ready state and getting the CPU for the first time.
8. Avg TAT → Sum of turnaround time of all the processes divided by number of processes.
9. Weighted turnaround time → How much of the total time of a process is spent on the CPU. (Burst time/TAT).
10. Weight time → Total amount of time of all waiting times for a process. (TAT-Burst time).
11. Schedule length → Total time to complete all processes in a given schedule or set of processes.
12. Deadline/Time quantum → The target completion is called Deadline. **Deadline-over-run** stands for If CT>DL it over sorts the deadline & **Deadline-under-run** CT<=DL.

***FCFS algorithm:***

According to the increasing arrival time process will be executed which process will enter once in CPU that will execute completely first then whose arrival time is strictly greater that process will get placed in CPU. If the CPU is allocated to a process then it can't be taken back until it finishes the execution of that process. This is implemented by using FIFO.

Numerical examples,





Therefore, without calculating these ST, CT, TAT, WT, we can answer L/scheduling length. The main thing wanted in order to draw the **Gantt chart** is PID, AT, BT.

