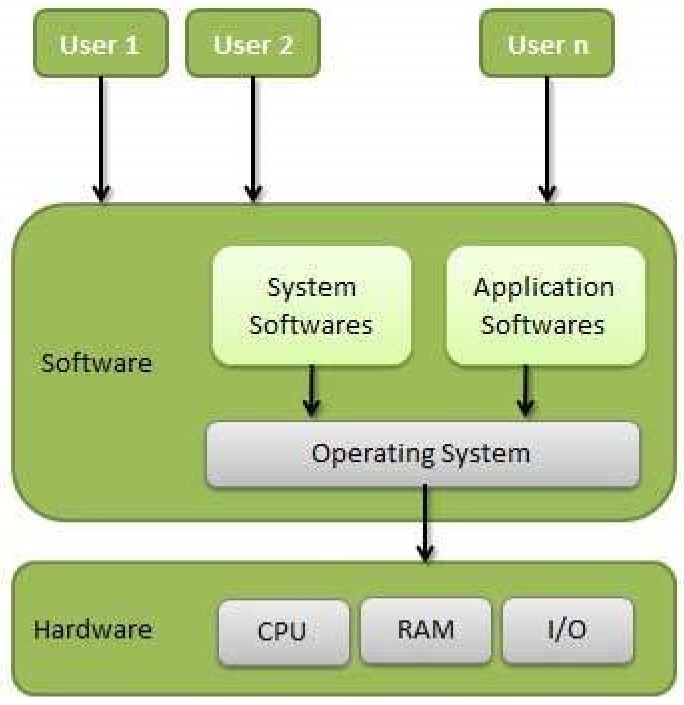
🦁

17. Operating System Basics

> An Operating System (OS) is an interface between a computer user and computer hardware. An operating system is a software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers.
>
> Some popular Operating Systems include Linux Operating System, Windows Operating System, Android.



> Following are some of important functions of an operating System.

- Memory Management

- Processor Management
- Device Management
- File Management
- Security
- Control over system performance
- Job accounting
- Error detecting aids
- Coordination between other software and users

Memory Management

> Memory management refers to management of Primary Memory or Main Memory. Main memory is a large array of words or bytes where each word or byte has its own address.
>
> Main memory provides a fast storage that can be accessed directly by the CPU. For a program to be executed, it must in the main memory. An Operating System does the following activities for memory management −

- Keeps tracks of primary memory, i.e., what part of it are in use by whom, what part are not in use.
- In multiprogramming, the OS decides which process will get memory when and how much.
- Allocates the memory when a process requests it to do so.
- De-allocates the memory when a process no longer needs it or has been terminated.

Processor Management

> In multiprogramming environment, the OS decides which process gets the processor when and for how much time. This function is called process scheduling. An Operating System does the following activities for processor management −

- Keeps tracks of processor and status of process. The program responsible for this task is known as traffic controller.
- Allocates the processor (CPU) to a process.
- De-allocates processor when a process is no longer required.

Device Management

> An Operating System manages device communication via their respective drivers. It does the following activities for device management −

- Keeps tracks of all devices. Program responsible for this task is known as the I/O controller.
- Decides which process gets the device when and for how much time.
- Allocates the device in the efficient way.
- De-allocates devices.

File Management

> A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions.
>
> An Operating System does the following activities for file management −

- Keeps track of information, location, uses, status etc. The collective facilities are often known as file system.
- Decides who gets the resources.
- Allocates the resources.
- De-allocates the resources.

Other Important Activities

> Following are some of the important activities that an Operating System performs −

- Security − By means of password and similar other techniques, it prevents unauthorized access to programs and data.
- Control over system performance − Recording delays between request for a service and response from the system.
- Job accounting − Keeping track of time and resources used by various jobs and users.
- Error detecting aids − Production of dumps, traces, error messages, and other debugging and error detecting aids.
- Coordination between other softwares and users − Coordination and assignment of compilers, interpreters, assemblers and other software to the various users of the computer systems.

Process

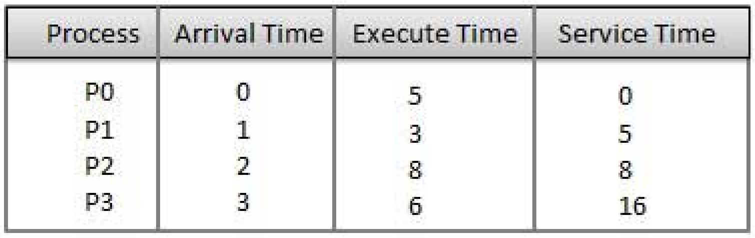
> A process is basically a program in execution. The execution of a process must progress in a sequential fashion.
>
> A Process Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms. There are

> some popular process scheduling algorithms which we are going to discuss−

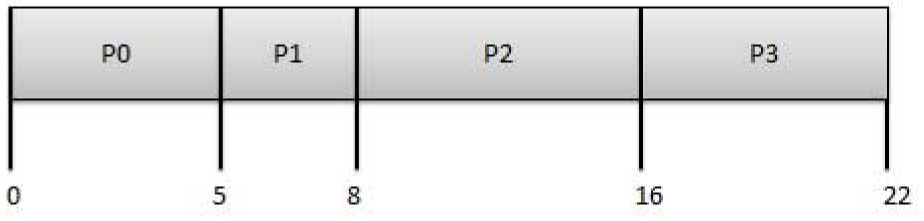
- First-Come, First-Served (FCFS) Scheduling
- Shortest-Job-Next (SJN) Scheduling
- Priority Scheduling
- Shortest Remaining Time
- Round Robin(RR) Scheduling
- Multiple-Level Queues Scheduling

> These algorithms are either non-preemptive or preemptive. Non-preemptive algorithms are designed so that once a process enters the running state, it cannot be preempted until it completes its allotted time, whereas the preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

First Come First Serve (FCFS)



Process	Arrival Time	Execute Time	Service Time
P0	0	5	0
P1	1	3	5
P2	2	8	8
P3	3	6	16



- Jobs are executed on first come, first serve basis.
- Its implementation is based on FIFO queue.
- Poor in performance as average wait time is high.

> Wait time of each process is as follows −

Process	Wait Time : Service Time - Arrival Time
P0	0 - 0 = 0
P1	5 - 1 = 4
P2	8 - 2 = 6
P3	16 - 3 = 13

> Average Wait Time: (0+4+6+13) / 4 = 5.75

Shortest Job Next (SJN)

- This is also known as shortest job first, or SJF
- Best approach to minimize waiting time.
- Easy to implement in Batch systems where required CPU time is known in advance.

Priority Based Scheduling

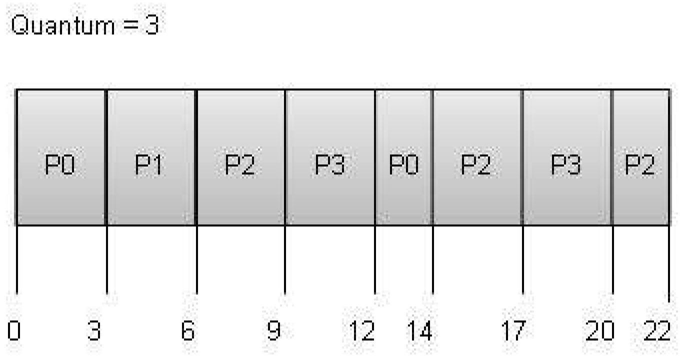
- Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems.
- Each process is assigned a priority. Process with highest priority is to be executed first and so on.
- Processes with same priority are executed on first come first served basis.
- Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Shortest Remaining Time

- Shortest remaining time (SRT) is the preemptive version of the SJN algorithm.
- The processor is allocated to the job closest to completion but it can be preempted by a newer ready job with shorter time to completion.
- Impossible to implement in interactive systems where required CPU time is not known.
- It is often used in batch environments where short jobs need to give preference.

Round Robin Scheduling

- Round Robin is the preemptive process scheduling algorithm.
- Each process is provided a fix time to execute, it is called a quantum.
- Once a process is executed for a given time period, it is preempted and other process executes for a given time period.
- Context switching is used to save states of preempted processes.



UNIX Introduction

> UNIX is an operating system which was first developed in the 1960s, and has been under constant development ever since.
>
> UNIX systems also have a graphical user interface (GUI) similar to Microsoft Windows.
>
> There are many different versions of UNIX, although they share common similarities. The most popular varieties of UNIX are Sun Solaris, GNU/Linux, and MacOS X. Linux in its turn is packaged in a form known as a Linux distribution. There are several Linux distributions, both free and commercial.
>
> **Linux Operating System**
>
> Linux is one of popular version of UNIX operating System. It is open source as its source code is freely available. It is free to use. Linux was designed considering UNIX compatibility. Its functionality list is quite similar to that of UNIX.

Components of Linux System

> Linux Operating System has primarily three components

- Kernel − Kernel is the core part of Linux. It is responsible for all major activities of this operating system. It consists of various modules and it interacts directly with the underlying hardware. Kernel provides the required abstraction to hide low level hardware details to system or application programs.
- System Library − System libraries are special functions or programs using which application programs or system utilities accesses Kernel's features. These libraries implement most of the functionalities of the operating system and do not requires kernel module's code access rights.
- System Utility − System Utility programs are responsible to do specialized, individual level tasks.

Basic Features

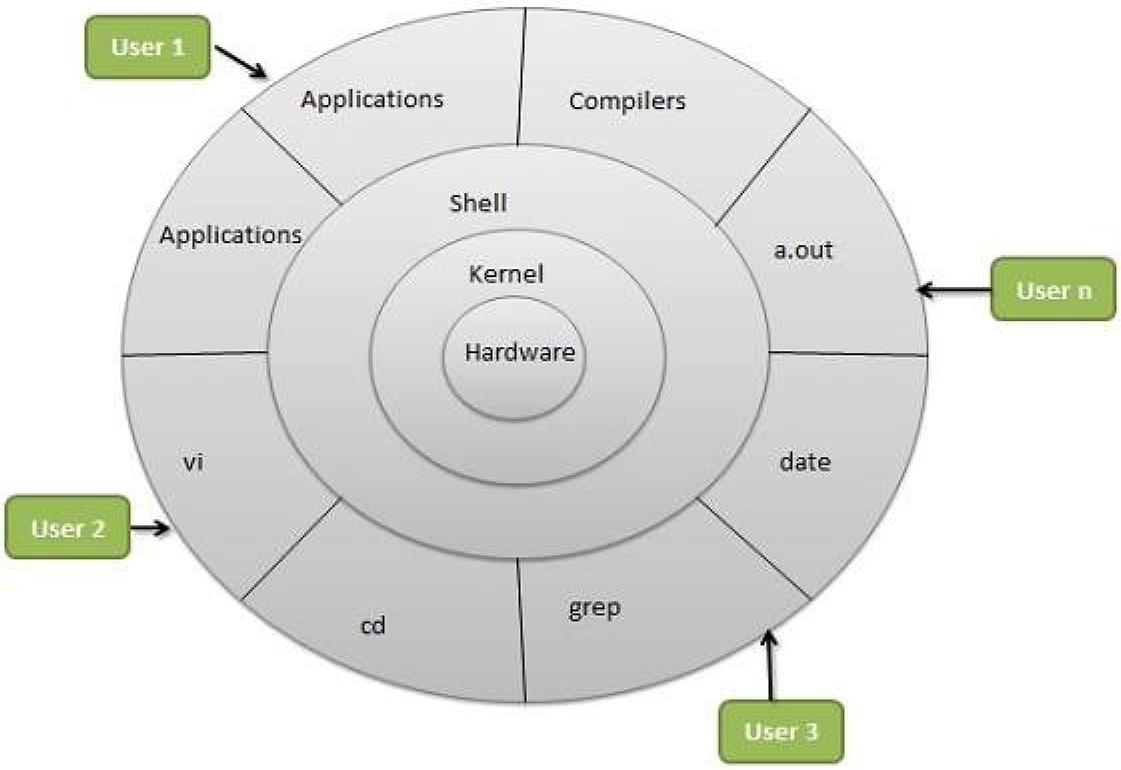
> Following are some of the important features of Linux Operating System.

- Portable − Portability means software can works on different types of hardware in same way. Linux kernel and application programs supports their installation on any kind of hardware platform.
- Open Source − Linux source code is freely available and it is community based development project. Multiple teams work in collaboration to enhance the capability of Linux operating system and it is continuously evolving.
- Multi-User − Linux is a multiuser system means multiple users can access system resources like memory/ ram/ application programs at same time.
- Multiprogramming − Linux is a multiprogramming system means multiple applications can run at same time.
- Hierarchical File System − Linux provides a standard file structure in which system files/ user files are arranged.
- Shell − Linux provides a special interpreter program which can be used to execute commands of the operating system. It can be used to do various types of operations, call application programs. etc.

- Security − Linux provides user security using authentication features like password protection/ controlled access to specific files/ encryption of data.

Architecture

> The following illustration shows the architecture of a Linux system −



> The architecture of a Linux System consists of the following layers −

- Hardware layer − Hardware consists of all peripheral devices (RAM/ HDD/ CPU etc).
- Kernel − It is the core component of Operating System, interacts directly with hardware, provides low level services to upper layer components.
- Shell − An interface to kernel, hiding complexity of kernel's functions from users. The shell takes commands from the user and executes kernel's functions.

- Utilities − Utility programs that provide the user most of the functionalities of an operating systems.

Deadlock

Every process needs some resources to complete its execution. However, the resource is granted in a sequential order.

1. The process requests for some resource.
2. OS grant the resource if it is available otherwise let the process waits.
3. The process uses it and release on the completion.

A Deadlock is a situation where each of the computer process waits for a resource which is being assigned to some another process. In this situation, none of the process gets executed since the resource it needs, is held by some other process which is also waiting for some other resource to be released.

Let us assume that there are three processes P1, P2 and P3. There are three different resources R1, R2 and R3. R1 is assigned to P1, R2 is assigned to P2 and R3 is assigned to P3.

After some time, P1 demands for R1 which is being used by P2. P1 halts its execution since it can't complete without R2. P2 also demands for R3 which is being used by P3. P2 also stops its execution because it can't continue without R3. P3 also demands for R1 which is being used by P1 therefore P3 also stops its execution.

In this scenario, a cycle is being formed among the three processes. None of the process is progressing and they are all waiting. The computer becomes unresponsive since all the processes got blocked.

Strategies for handling Deadlock

1. Deadlock Ignorance

Deadlock Ignorance is the most widely used approach among all the mechanism. This is being used by many operating systems mainly for end user uses. In this approach, the Operating system assumes that deadlock never occurs. It simply ignores deadlock. This approach is best suitable for a single end user system where User uses the system only for browsing and all other normal stuff.

In these types of systems, the user has to simply restart the computer in the case of deadlock. Windows and Linux are mainly using this approach.

2. Deadlock prevention

Deadlock happens only when Mutual Exclusion, hold and wait, No preemption and circular wait holds simultaneously. If it is possible to violate one of the four conditions at any time then the deadlock can never occur in the system.

3. Deadlock avoidance

In deadlock avoidance, the operating system checks whether the system is in safe state or in unsafe state at every step which the operating system performs. The process continues until the system is in safe state. Once the system moves to unsafe state, the OS has to backtrack one step.

In simple words, The OS reviews each allocation so that the allocation doesn't cause the deadlock in the system.

4. Deadlock detection and recovery

This approach let the processes fall in deadlock and then periodically check whether deadlock occur in the system or not. If it occurs then it applies some of the recovery methods to the system to get rid of deadlock.

Race Condition

A race condition occurs in the software when the computer program depends on the threads or processes of a program.

In software, we cannot debug the race condition because the final result is non-deterministic and based on the timing of multiple threads.

A Race Condition typically occurs when two or more threads try to read, write and possibly make the decisions based on the memory that they are accessing concurrently.

```
e.g. (((x+5)*10)/2)-1

add 5
mul 10
dev 2
sub 1
```

Critical Section

The regions of a program that try to access shared resources and may cause race conditions are called critical section. To avoid race condition among the processes, we need to assure that only one process at a time can execute within the critical section.

Critical Section is the part of a program which tries to access shared resources. That resource may be any resource in a computer like a memory location, Data structure, CPU or any IO device.

The critical section cannot be executed by more than one process at the same time; operating system faces the difficulties in allowing and disallowing the processes from entering the critical section.

The critical section problem is used to design a set of protocols which can ensure that the Race condition among the processes will never arise.

In order to synchronize the cooperative processes, our main task is to solve the critical section problem. We need to provide a solution in such a way that the following conditions can be satisfied.

Conditions:
Mutual Exclusion
Progress

Necessary conditions for Deadlocks

1. **Mutual Exclusion**

A resource can only be shared in mutually exclusive manner. It implies, if two process cannot use the same resource at the same time.

1. **Hold and Wait**

A process waits for some resources while holding another resource at the same time.

1. **No preemption**

The process which once scheduled will be executed till the completion. No other process can be scheduled by the scheduler meanwhile.

1. **Circular Wait**

All the processes must be waiting for the resources in a cyclic manner so that the last process is waiting for the resource which is being held by the first process.

Optimistic Locking

is a strategy where you read a record, take note of a version number (other
methods to do this involve dates, timestamps or checksums/hashes) and check that the version
hasn't changed before you write the record back. When you write the record back you filter the
update on the version to make sure it's atomic. (i.e. hasn't been updated between when you check
the version and write the record to the disk) and update the version in one hit.

If the record is dirty (i.e. different version to yours) you abort the transaction and the user can re-
start it.

This strategy is most applicable to high-volume systems and three-tier architectures where you do
not necessarily maintain a connection to the database for your session. In this situation the client
cannot actually maintain database locks as the connections are taken from a pool and

you may not
be using the same connection from one access to the next.

Pessimistic Locking

is when you lock the record for your exclusive use until you have finished with it.
It has much better integrity than optimistic locking but requires you to be careful with your
application design to avoid Deadlocks. To use pessimistic locking you need either a direct connection
to the database (as would typically be the case in a __two tier client server__application) or an
externally available transaction ID that can be used independently of the connection.
In the latter case you open the transaction with the TxID and then reconnect using that ID. The DBMS
maintains the locks and allows you to pick the session back up through the TxID.

Some popular Linux Commands:

1. ls

List directory contents. If you know windows you would know that the command dir is used to list the contents in a directory. In Linux, the *ls* command is used to list out files and directories. Some versions may support color-coding. The names in blue represent the names of directories.

ls -l | *more* this helps to paginate the output so you can view page by page. Otherwise the listing scrolls down rapidly. You can always use *ctrl c* to go back to the command line.

$ ls -l filename

2. cd /var/log –

Change the current directory. The forward slash is to be used in Linux. The example is a Linux directory that comes with all versions of Linux.

When you use *ls –l* you will be able to see more details of the contents in the directory

It will list down the

- Permissions associated with the file
- The owner of the file
- The group associated with the file
- The size of the file
- The time stamp
- The name of the file

$ cd /var/log

3. grep –

Find text in a file. The *grep* command searches through many files at a time to find a piece of text you are looking for.

grep PATTERN [FILE]

grep failed transaction.log

The above command will find all of the words in the files that matched the word ‘failed’.

$ grep ‘failed’ transaction.log

4. su / sudo command –

There are some commands that need elevated rights to run on a Linux system. So you run them as a System Administrator which normal users cannot do.

su command changes the shell to be used as a super user and until you use the exit command you can continue to be the super user

sudo – if you just need to run something as a super user, you can use the *sudo* command. This will allow you to run the command in elevated rights and once the command is executed you will be back to your normal rights and permissions.

Example – shutdown command the shutdown command safely turns off the computer system.

- *sudo shutdown 2* – shutdown and turns of the computer after 2 minutes
- *sudo shutdown –r 2* – shuts down and reboots in 2 minutes

- Using *ctrl C* or *shutdown –c* helps in stopping the shutdown process.

$ sudo **shutdown** 2

$ sudo **shutdown** –r 2

5. pwd – Print Working Directory

One way to identify the directory you are working in is the *pwd* command

It displays the current working directory path and is useful when directory changes are often

$ pwd

6. passwd –

Though looks similar to the *pwd* command the role it plays is different.

This command is used to change the user account password. You could change your password or the password of other users. Note that the normal system users may only change their own password, while *root* may modify the password for any account.

passwd [username] - changes the password for the user.

$ passwd admin

7. mv – Move a file

To move a file or rename a file you would use the *mv* command.

Here the file name gets changed from *first.txt* to *second.txt.*

Type *ls* to view the change

$ mv first.txt second.txt

8. cp – Copy a file

cp source file destination file. In case you need a copy of the file *second.txt* in the same directory you have to use the *cp* command

$ cp second.txt third.txt

You can use *ls – l* to see the new file created. The two files will be exactly of the same size.

9. rm –

This command is used to remove files in a directory or the directory itself. A directory cannot be removed if it is not empty.

rm [name of the file]

rm –r removes all the contents in a directory and the directory as well.

```
$ rm file1
```

```
$ rm -r myproject
```

10. mkdir – to make a directory.

mkdir [directory name] if you would like to create a directory in the name 'myproject' type

mkdir myproject

$ **mkdir** myproject

14. echo –

This command is used to display a text or a string to the standard output or a file.

```
$ echo "This is an article on basic linux commands"
```

```
This is an article on basic linux commands
```

The *echo –e* option acts as an interpretation of escape characters that are back-slashed.

$ echo –e "This **is** an article **is for** beginners. \nIt **is on** basic linux commands

Will display the output as

This **is** an article **is for** beginners.

It **is on** basic linux commands

\n the newline character is interpreted by the echo –e command

15. wc -

The *wc* (word count) command in Linux operating system is used to find out the number of new lines, word count, byte and characters count in a file specified by the file arguments.

wc [options] filenames.

$ wc –l readme.txt

Shows the output as - 120 readme.txt

- **wc -l** : Prints the number of lines in a file.
- **wc -w** : prints the number of words in a file.
- **wc -c** : Displays the count of bytes in a file.
- **wc -m** : prints the count of characters from a file.
- **wc -L** : prints only the length of the longest line in a file.

16. man –

This command is used to view the on-line reference manual pages for commands/programs.

$ man grep

$ man mkdir

17. history –

This command is used to show previously used commands or to get information about the commands executed by a user.

$ history

18. clear –

This command lets you clear the terminal screen.

$ clear

20. reboot –

This command may be used to halt, power-off or reboot a system as follows.

$ reboot

References:

https://www.tutorialspoint.com/operating_system/os_overview.htm

https://hackr.io/blog/basic-linux-commands

https://www.hpc.iastate.edu/guides/unix-introduction