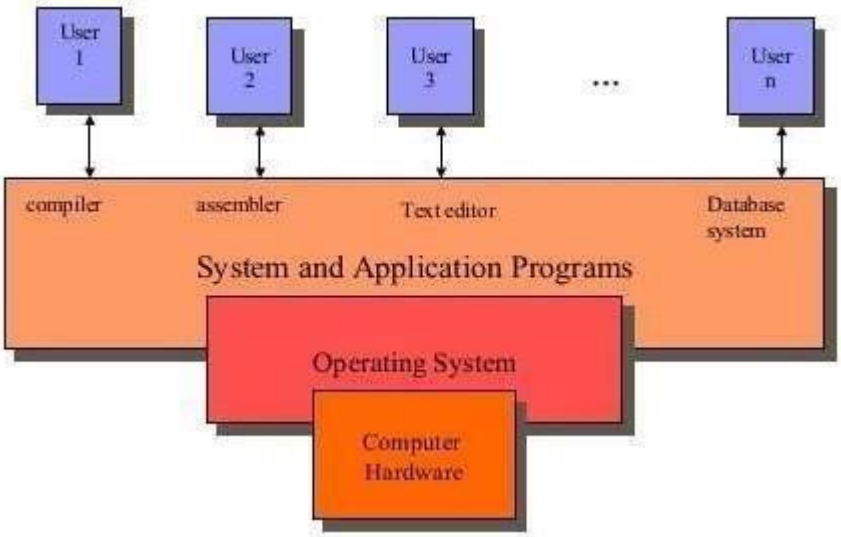
UNIT–1

Operating System Introduction-Structure-Simple batch,Multiprogramming,Time-shared,Personal Computer,Parallel,Distributed System, Real time System,System Components,OS services,System calls.

Process-Processconceptsandscheduling,Operationsonprocess,Cooperatingprocess, Threads

IntroductionofOperatingSystem

- AnOSactasaninterfacebetweenuserandsystemhardware.
- Computerconsistsofthehardware,OperatingSystem,systemprograms, application programs.
- The hardwareconsists of memory,CPU,ALU,I/O device,storage deviceand peripheral device.
- Systemprogramconsistsofcompilers,loaders,editors,OS etc.
- Applicationprogramconsistsofdatabaseprograms, businessprograms.
- EverycomputermusthaveanOStorunotherprograms.
- TheOScontrols&coordinatestheuseofthehardwareamongthevarious system programs and application programs for various tasks.
- Itsimplyprovidesanenvironmentwithinwhichotherprogramscando useful work.



OPERATINGSYSTEM

Definition

- In the 1960's one might have defined OS as **"The software that controls thehardware"**.
- OperatingSystemperformsallthebasictaskslikemanagingfiles,processes,

andmemory.Thusoperating systemactsasthemanager ofalltheresources,i.e. **resourcemanager**.

- Operating system becomes an interface between the user and the machine.It is one of the most required software that is present in the device.
- Operating System is a type of software that works as an interface betweenthesystem program and the hardware.

ConceptofOS

- The OS is a set of special programs that runon a computer systemthatallow it to work properly.
- It performs basic task as recognizing inputfromthe keyboard, keeping track of files and directories on the disk, sending output to thedisplay screen and controlling a peripheral device.
- TheOSmustsupportthefollowingtasks. Theyare,
  - Providesthefacilitiestocreate,modificationofprogramanddatafileusing an editor.
  - Accesstothecompilerfortranslatingtheuserprogramfromhighlevel language to machine language.
  - Providealoaderprogramtomovethecompiledprogramcodetothe computer memory for execution.

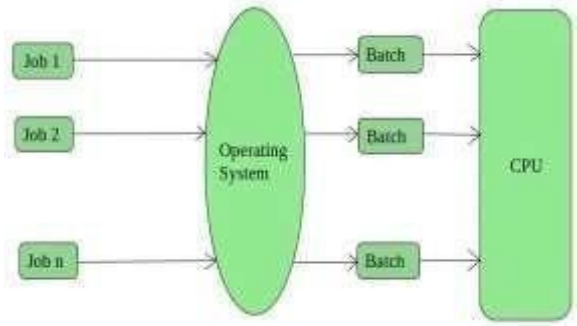
TypesofOperatingSystems

ThereareseveraltypesofOperatingSystemswhicharementionedbelow.

- BatchOperatingSystem
- Multi-ProgrammingSystem
- Time-SharingOperatingSystem
- PersonalComputers
- ParallelOperatingSystem
- DistributedOperatingSystem
- Real-TimeOperatingSystem

1. BatchOperatingSystem

Thistypeofoperatingsystem doesnotinteractwiththecomputer directly.Thereisanoperatorwhich takessimilarjobshavingthesame requirement and groups them into batches.Itistheresponsibilityofthe operator to sort jobs withsimilarneeds.



Advantages

- Processors of the batch systems know how long the job would be when itis inthe queue.

- Multipleuserscansharethebatchsystems.
- Theidletimeforthebatchsystemisvery less.
- Itiseasytomanagelargeworkrepeatedly inbatchsystems.

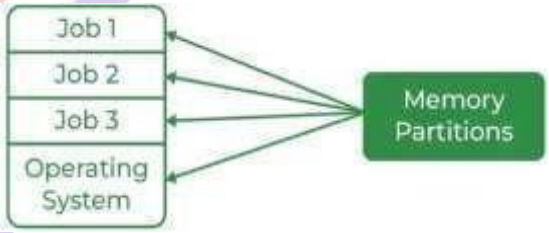
Disadvantages

- Thecomputeroperatorsshouldbewellknownwithbatchsystems.
- Batch systemsarehardtodebug.
- Itissometimescostly.
- Theotherjobswillhavetowaitforanunknowntimeifanyjobfails.
- Itisverydifficulttoguessorknowthetimerequiredforanyjobtocomplete.

Examples

Payroll Systems,BankStatements,etc.

2. Multi -Programming Operating System



Multiprogramming Operating Systemscanbesimplyillustratedasmore thanoneprogramispresentinthemain memoryand anyone of themcan be kept in execution.Thisisbasicallyusedforbetter executionofresources.

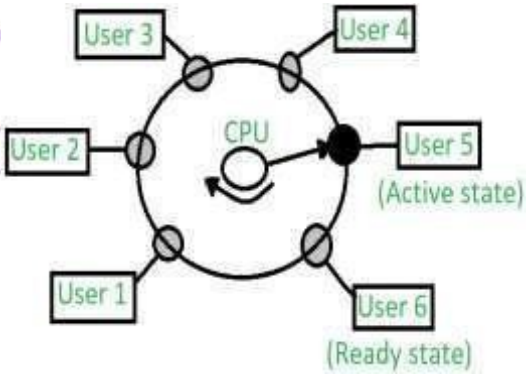
AdvantagesofMulti-ProgrammingOperatingSystem

- MultiProgrammingincreasestheThroughputoftheSystem.
- Ithelpsinreducingtheresponsetime.

DisadvantagesofMulti-ProgrammingOperatingSystem

- Thereisnotanyfacilityforuserinteractionofsystemresourceswiththesystem.

3. Time-SharingOperatingSystems



Each task is given some time toexecute so that all the tasks work smoothly. Each user gets thetime ofthe CPU as they use asingle system. Thesesystemsarealso known asMultitasking Systems. Thetaskcanbe from a single user or different users also. The time that each task gets to execute is called quantum. After this time interval is over OS switches over to the next task.

Advantages

- Eachtaskgetsanequalopportunity.
- Fewerchancesofduplicationofsoftware.
- CPUidletimecanbereduced.
- **Resource Sharing:** Time-sharing systems allow multiple users to share hardware resources such as the CPU, memory, and peripherals,reducing the cost of hardware and increasing efficiency.
- **Improved Productivity:** Time-sharing allows users to work concurrently, thereby reducing the waiting time for their turn to use the computer. This increased productivity translates to more work getting done in less time.
- **Improved UserExperience:** Time-sharingprovidesaninteractiveenvironment that allows users to communicate with the computer in real time, providing a better user experience than batch processing.

Disadvantages

- Reliabilityproblem.
- Onemusthavetotakecare ofthesecurity andintegrity of userprogramsand data.
- Datacommunicationproblem.
- **High Overhead:** Time-sharing systems have a higher overhead than other operating systems due to the need for scheduling, context switching, and other overheads that come with supporting multiple users.
- **Complexity:** Time-sharing systems are complexand require advanced software to manage multiple users simultaneously. This complexity increases the chance of bugs and errors.
- **Security Risks:** With multiple users sharing resources, the risk of security breaches increases. Time-sharing systems require careful management of user access, authentication, and authorization toensure the security of data and software.
- - sharing operating system that allows multiple users to access a Windowsserver remotely. Users can run their own applications and access shared resources, such as printers and network storage, in real-time.

4. PersonalComputer

A personal computer (PC) is a microcomputer designed for use by one person at a time.

Prior to the PC, computers were designed for -- and only affordable for -- companiesthatattachedterminalsformultipleuserstoasingle large mainframe computerwhoseresourceswereshareddamongallusers.By the

1980s,technologicaladvancesmadeitfeasibletobuildasmallcomputerthatan

individualcouldownanduse asaword processorandforothercomputingfunctions.

Whether they are home computers or business ones, PCs can be used to store, retrieve and process data of all kinds. A PC runs firmware that supports anoperating system (OS), which supports a spectrum of other software.This software lets consumers and business users perform a range of general-purpose tasks, such as the following:

- wordprocessing
- spreadsheets
- email
- instantmessaging
- accounting
- databasemanagement
- internetaccess
- listeningtomusic
- network-attachedstorage
- graphicdesign
- musiccomposition
- videogaming
- softwaredevelopment
- networkreconnaissance
- multimediaservers
- wirelessnetworkaccesshotspots
- videoconferencing

Types

Personalcomputersfallintovariouscategories,suchasthefollowing:

- **Desktopcomputers**usuallyhaveatower, monitor,keyboardandmouse.
- **Tablets**aremobiledeviceswithatouchscreendisplay.
- **Smartphones**arephoneswithcomputing capabilities.
- **Wearables** are devices users wear,such as smartwatches andvarious typesofsmart clothing.
- **Laptop computers** are portable personal computers that usually come withanattached keyboard and trackpad.
- **Notebookcomputers**arelightweightlaptops.
- **Handheldcomputers**includeadvancedcalculatorsandvariousgamingdevices.

5. ParallelOperatingSystem

Parallel Systems are designed to speed up the execution of programs bydividing the programs into multiple fragments and processing these fragments at the same time.

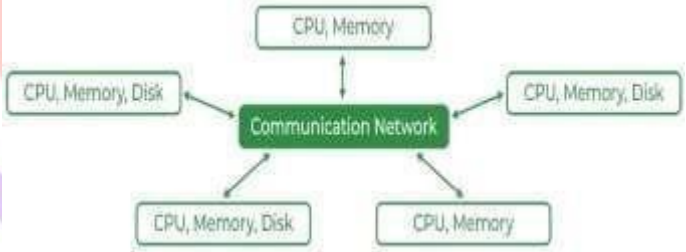
Advantages

- **High Performance:** Parallel systems can executecomputationally intensive tasks more quickly compared to single processor systems.
- **Cost Effective:** Parallel systems can be more cost-effective compared to distributed systems, as they do not require additional hardware for communication.

Disadvantages

- **LimitedScalability:**Parallelsystemshavelimitedscalabilityasthenumber of processors or cores in a single computer is finite.
- **Complexity:**Parallelsystemsaremorecomplextoprogramanddebug compared to single processor systems.
- **SynchronizationOverhead:**Synchronizationbetween processors in aparallel system can add overhead and impact performance.

6. DistributedOperatingSystem



These types of operating systems are a recent advancement in the world of computer technology and are being widely accepted all over the world and, that too, at agreat pace.

Various autonomous interconnected computers communicate with each other using a shared communication network. Independent systems possess their own memory unit and CPU. These are referred to as loosely coupled systems or distributed systems. These systems' processors differ in size and function.

The major benefit of working with these types of the operating system isthat it is always possible that one user can access the files or software which are not actually present on his system but some other systemconnected within this network i.e., remote access is enabled within the devices connected in that network.

TypesofDistributed Systems

The nodes in the distributed systems can be arranged in the form ofclient/server systems or peer to peer systems. Details about these are as follows −

Client/ServerSystems

In client server systems, the client requests a resource and the server provides thatresource. A server may serve multiple clients at the same time whilea client is in contact with only one server. Both the client and server usually communicate via a computer network and so they are a part ofdistributed systems.

PeertoPeerSystems

The peer to peer systems contains nodes that are equal participants in data sharing. All the tasks are equally divided between all the nodes. Thenodesinteract witheachother as required as share resources. This is done withthe help ofa network.

Advantages

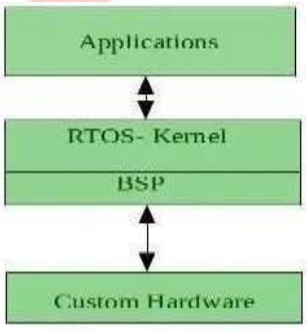
- Failureofonewillnotaffecttheothernetworkcommunication,asall systems are independent of each other.
- Electronicmailincreasesthedataexchangespeed.
- Sinceresourcesarebeingshared,computationishighly fastanddurable.
- Loadonhostcomputerreduces.
- Thesesystemsareeasily scalableasmany systemscanbeeasily addedtothe network.
- Delayindataprocessingreduces.

Disadvantages

- Failureofthemainnetworkwillstoptheentirecommunication.
- Toestablishdistributedsystemsthelanguageisusednotwell-definedyet.
- These types of systems are not readily available astheyarevery expensive.Notonly thattheunderlyingsoftwareishighly complexandnot understood well yet.

Example:LOCUS

7. Real-TimeOperatingSystem



These types of OSs serve real-time systems. The time interval required to process and respond to inputs is very small. This time interval is called **response time**.

Real-time systems are used when there are time requirements that are very strict like missile systems, air traffic control systems, robots, etc.

Types:

1. **HardReal-TimeSystems**

   Hard Real-Time OSs are meant for applications where time constraintsare very strictand even the shortestpossible delay is not acceptable.These systemsare built for saving life like automatic parachutes or airbags whicharerequiredto be readily available in case of an accident. Virtual memory is rarely found in these systems.

2. **SoftReal-TimeSystems**

   TheseOSsareforapplicationswheretime-constraintislessstrict.

Advantages

- **MaximumConsumption:** Maximumutilizationofdevicesandsystems,thus more output from all the resources.
- **TaskShifting:**Thetimeassignedforshiftingtasksinthesesystemsis very less. For example, in older systems, it takes about 10 microsecondsin shifting from one task to another, and in the latest systems, it takes 3 microseconds.

- **Focus on Application:** Focus on running applications and less importance on applications that are in the queue.
- Real-time **operating system in** the **embedded system:** Since the size of programs is small, RTOS can also be used in embedded systems like intransport and others.
- **ErrorFree:**Thesetypesofsystemsareerror-free.
- **Memory Allocation:** Memory allocation is best managed in these typesof systems.

Disadvantages

- **Limited Tasks:** Very few tasks run at the same time and theirconcentration is very less on a few applications to avoid errors.
- **Useheavysystemresources:**Sometimesthesystemresourcesarenotsogood and they are expensive as well.
- **ComplexAlgorithms:**Thealgorithmsareverycomplexanddifficultforthe designer to write on.
- **Devicedriverandinterruptsignals:**Itneedsspecificdevicedriversand interrupts signal to respond earliest to interrupts.
- **ThreadPriority:**Itisnotgoodtosetthreadpriorityasthesesystemsare very less prone to switching tasks.

Examples

Scientific experiments, medical imaging systems, industrial control systems, weapon systems,robots,airtrafficcontrolsystems,etc.

OperatingSystemServices

- **UserInterface -** User interface is essentialand all operating systems provide it. Users either interface with the operating system through command-lineinterface (CUI) or graphical userinterface(GUI). Command interpreter executes next user-specifiedcommand.AGUI offers the user a mouse-based window and menu system as an interface.
- **Program execution** - The system must be able to load a program into memory and to run that program, end execution, either normally or abnormally (indicating error)
- **I/O operations** -A running program may require I/O, which may involvea file or an I/O device.

- **File-systemmanipulation** - The file system isofparticular interest. Obviously, programs need to read and write files and directories, create anddelete them, search them, list file Information, permission management.
- **Communications**–Processesmayexchangeinformation,onthesame

computer or between computers over a network. Communications may be via shared memoryor through message passing (packets moved by the OS)

- **Error detection** – OS needs to be constantly aware of possible errors mayoccur in the CPU and memory hardware, in I/O devices, in user program. For each type of error, OS should take the appropriate action to ensure correct and consistent computing. Debugging facilities can greatly enhance the user's and programmer's abilities to efficiently use the system.

AnothersetofOSfunctionsexistsforensuringtheefficientoperationofthe system itself via resource sharing

- **Resource allocation -** When multiple users or multiple jobs running concurrently, resources must be allocated to each of them. Many types of resources such as CPU cycles, main memory, and file storage may have special allocation code, others such as I/O devices may havegeneral request andrelease code.
- **Accounting -** To keep track of which users use how much and what kinds of computer resources
- **Protection and security -** The owners of information stored in a multiuser or networked computer system may want to control use of that information, concurrent processes should not interfere with each other. **Protection** involves ensuring that all access to systemresources is controlled. **Security**ofthe system from outsiders requires user authentication, extends to defending external I/O devices from invalid access attempts. If a system is to be protected and secure, precautions must be instituted throughout it. A chain is only as strong as its weakest link.

SystemCalls

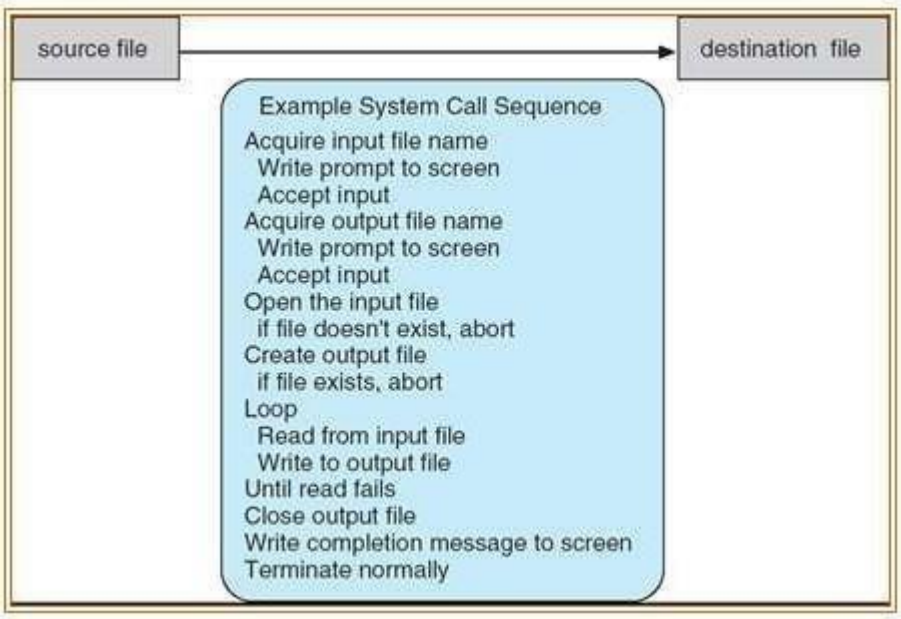
- Asystemcallisawayforauserprogramto interfacewiththeoperating system. The program requests several services, and the OS responds by invoking aseries of system calls to satisfy the request.
- Asystemcallcanbewritteninassemblylanguageorahigh-levellanguagelike **C,C++**or**Pascal**.
- System callsarepredefinedfunctionsthattheoperatingsystemmaydirectly invoke if a high-level language is used.
- A system call is a method for a computer program to request a service fromthe kernel of the operating system on which it is running.
- Asystemcallisamethod ofinteracting withtheoperating systemviaprograms.
- Asystemcallisarequestfromcomputersoftwaretoanoperatingsystem's kernel.

- A simple system call may take few nanoseconds to provide the result, like retrieving the system date and time. A more complicated system call, such as connecting to a network device, may take a few seconds. Most operating systems launch a distinct kernel thread for each system call to avoidbottlenecks. Modern operating systems are multi-threaded, which means they can handle various system calls at the same time.
- The **Application Program Interface (API)** connects theoperating system's functions to user programs. It acts as a link between the operating systemand a process, allowing user-levelprograms to request operating systemservices. The kernel system can only be accessedusing system calls. System calls arerequired for any programs that use resources.

- Whencomputersoftwareneedsto accesstheoperatingsystem'skernel, it makes a system call. The system call uses an API to expose the operating system's services touserprograms. It istheonlymethodto accessthe kernelsystem. All programs or processes that require resources for execution must use system calls, as they serve as an interface between the operating system and user programs.

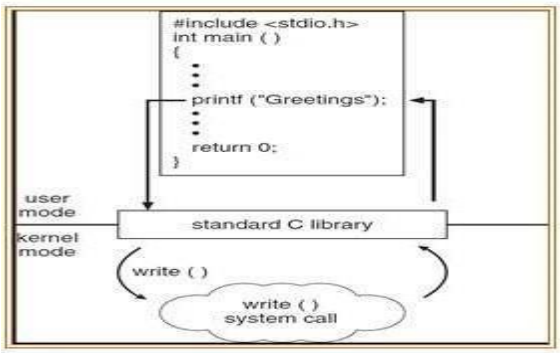
ExampleofSystemCalls

Systemcallsequence tocopythecontentsofonefileto anotherfile



StandardCLibrary Example

Cprograminvokingprintf()librarycall,whichcallwrite()systemcall



Therearevarioussituationswherewemustrequiresystemcallsintheoperating system. Following of the situations are as follows:

1. Itismust requirewhenafilesystemwantstocreateordeletea file.
2. Networkconnectionsrequirethesystemcallstosendingandreceivingdata packets.
3. Ifyouwanttoreadorwriteafile,youneedtosystemcalls.
4. Ifyouwanttoaccesshardwaredevices,includingaprinter,scanner,youneed a system call.
5. Systemcallsareusedtocreateandmanagenewprocesses.

TypesofSystem Calls

Thereare commonlyfivetypesofsystemcalls.Theseareas follows:

1. **ProcessControl**
2. **FileManagement**
3. **DeviceManagement**
4. **InformationMaintenance**
5. **Communicatio**

nProcess Control

Process control is the system call that is used to direct the processes. Some process control examples include creating, load, abort, end, execute, process,terminate the process, etc.

FileManagement

Filemanagementisasystemcallthatisusedtohandlethefiles.Somefile management examples include creating files, delete files, open, close, read, write, etc.

DeviceManagement

Device management is a system call that is used to deal with devices. Some examples of device management include read, device, write, get device attributes, release device, etc.

InformationMaintenance

Information maintenance is a system call that is used to maintain information. Therearesomeexamplesofinformationmaintenance, includinggettingsystemdata, set time or date, get time or date, set system data, etc.

Communication

Communication isa systemcallthat isused forcommunication. There are some examples of communication, including create, delete communication connections,send, receive messages, etc.

ExamplesofWindowsand Unixsystemcalls

Process	Windows	Unix
ProcessControl	CreateProcess() ExitProcess() WaitForSingleObject()	Fork() Exit() Wait()
FileManipulation	CreateFile()ReadFile() WriteFile() CloseHandle()	Open() Read() Write() Close()
DeviceManagement	SetConsoleMode() ReadConsole()WriteConsole()	Ioctl() Read() Write()
InformationMaintenance	GetCurrentProcessID() SetTimer() Sleep()	Getpid() Alarm() Sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	Pipe() Shmget() Mmap()
Protection	SetFileSecurity() InitializeSecurityDescriptor() SetSecurityDescriptorgroup()	Chmod() Umask() Chown()

open()

The**open()**systemcallallowsyoutoaccessafileonafilesystem.It allocates resourcestothe file and provides a handlethatthe process mayreferto. Many processes can open a file at once or by a singleprocess only. It's all based on the file system and structure.

read()

It is used to obtain data from a file on the file system. It accepts threearguments in general:

- Afiledescriptor.
- Abuffertostorereaddata.
- Thenumberofbytesto readfromthefile.

Thefiledescriptorofthefiletobereadcouldbeusedtoidentifyitandopenit using **open()** before reading.

wait()

Insomesystems,aprocessmayhavetowaitforanotherprocessto completeitsexecutionbeforeproceeding.Whenaparentprocessmakesachild process, the parentprocess execution is suspended until the child process isfinished.The**wait()** systemcallisusedto suspendtheparent process. Oncethe childprocesshas completed its execution, control is returned to the parent process.

write()

Itisusedtowritedatafromauserbuffertoadevicelikeafile.Thissystemcall is one way for a programto generate data. It takes three arguments in general:

- Afiledescriptor.
- Apointertothebufferinwhichdatais saved.
- Thenumberofbytestobewrittenfromthebuffer.

fork()

Processesgenerateclonesofthemselvesusingthe**fork()**systemcall.Itis one ofthe most common ways to create processes in operating systems. When a parent process spawns a child process, execution of the parent process is interrupted until the child process completes. Once the child processhas completed its execution, controlis returned to the parent process.

close()

Itisusedtoendfilesystemaccess.Whenthissystemcall isinvoked,itsignifies thattheprogramnolongerrequiresthefile,andthebuffersareflushed,thefileinformation is altered, and the file resources are de-allocated as a result.

exec()

Whenanexecutablefilereplacesanearlierexecutablefileinanalready executing process,thissystem function isinvoked.As a new processisnotbuilt,the oldprocess identificationstays,butthenewprocessreplacesdata,stack,data,head,etc.

exit()

The**exit()**isasystemcallthatisusedtoendprogramexecution.Thiscall indicates that the thread execution is complete, which is especially useful in multi-threaded environments. The operating system reclaims resources spent by the process following the use of the **exit()** system function.

SystemcomponentsinOS:-

An operating system is a large and complex system that can only be created by partitioning intosmall parts. These pieces should be a well-defined part of the system, carefully defining inputs,outputs,andfunctions.

Although Windows, Mac, UNIX, Linux, and other OS do not have the same structure, mostoperating systems share similar OS system components, such as file,memory, process, I/Odevicemanagement.

Thecomponentsofanoperatingsystemplayakeyroletomakeavarietyof computer systempartsworktogether.There arethefollowingcomponentsofanoperatingsystem,suchas:

1. ProcessManagement
2. FileManagement
3. NetworkManagement
4. MainMemoryManagement
5. SecondaryStorageManagement
6. I/ODeviceManagement
7. SecurityManagement
8. CommandInterpreterSystem

Operatingsystemcomponentshelp yougetthecorrect computingbydetectingCPU and memoryhardware errors.

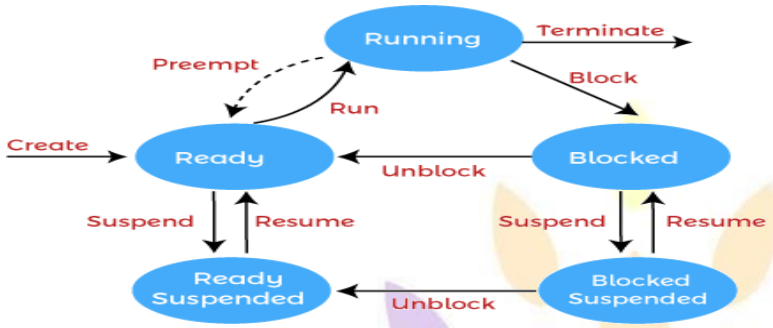
1. ProcessManagement

The process management component is a procedure for managing many processes running simultaneously on the operating system. Every running softwareapplication program has one or more processes associated with them.

For example, when you use a search engine like Chrome, there is a process running for thatbrowserprogram.

Processmanagementkeepsprocessesrunningefficiently.Italsousesmemoryallocated to themandshuttingthemdownwhenneeded.

The execution of a process must be sequential so, at least one instruction should be executed onbehalfoftheprocess.



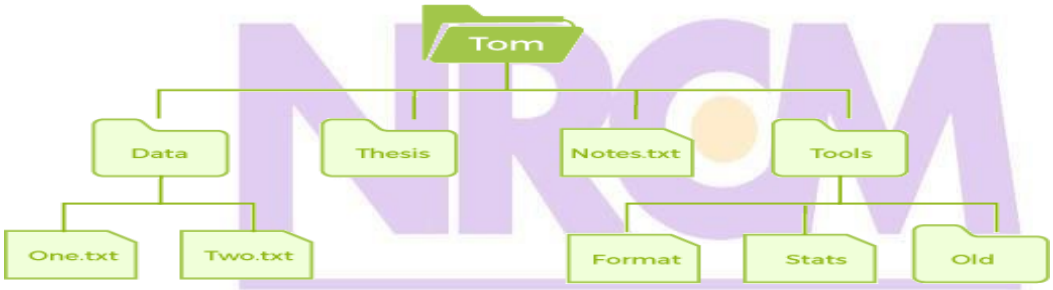
Functionsofprocessmanagement

Herearethefollowingfunctionsofprocessmanagementintheoperatingsystem,suchas:

- Processcreationanddeletion.
- Suspensionandresumption.
- Synchronizationprocess
- Communicationprocess

2. FileManagement

A file is a set of related information defined by its creator. It commonly represents programs(bothsourceandobjectforms)anddata. Datafiles canbealphabetic, numeric, oralphanumeric.



Functionoffilemanagement

Theoperatingsystemhasthefollowingimportantactivitiesinconnectionwithfilemanagement:

- Fileanddirectorycreationanddeletion.
- Formanipulatingfilesanddirectories.
- Mappingfilesontosecondarystorage.
- Backupfilesonstablestoragemedia.

3. NetworkManagement

Networkmanagementis the process of administeringandmanagingcomputernetworks.Itincludesperformance management, provisioning of networks, fault analysis, and maintaining thequalityofservice.

A distributed system is a collection of computers or processors thatnever share theirmemoryand clock. In this type of system, all the processors have their local memory, and the processorscommunicate with each other using differentcommunication cables, such as fibre optics ortelephonelines.

Thecomputersinthenetworkareconnectedthroughacommunicationnetwork, which canconfigureinmanydifferentways.Thenetworkcanfullyorpartiallyconnectinnetwork management,whichhelpsusersdesignroutingandconnectionstrategiesthatovercomeco nnectionandsecurityissues.

FunctionsofNetworkmanagement

Networkmanagementprovidesthefollowingfunctions,suchas:

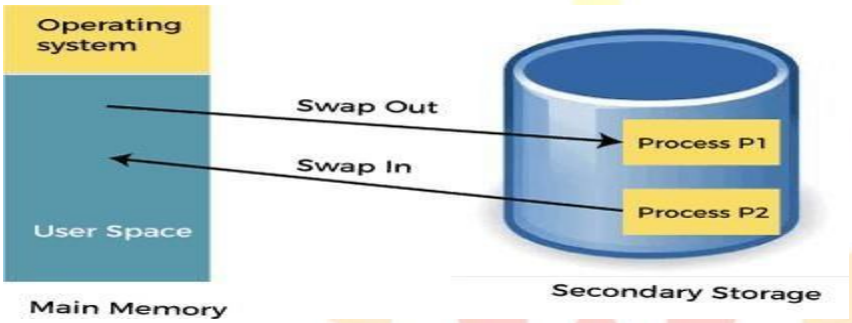
- Distributed systems help you to various computing resources in size and function. Theymayinvolveminicomputers,microprocessors,andmanygeneral-purposecomputersystems.
- A distributed system also offers the user access to the various resources the networkshares.
- It helps to access shared resources that helpcomputation to speed up oroffers dataavailabilityandreliability.

4. MainMemorymanagement

Mainmemoryisalargearrayofstorageorbytes,whichhasanaddress.Thememorymanage mentprocessisconductedbyusingasequenceofreadsorwritesofspecificmemoryaddresses.

It should be mapped to absolute addressesand loaded inside the memoryto execute a program.The selectionofamemorymanagementmethoddependsonseveralfactors.

However, it is mainly based on the hardware design of the system. Each algorithm requirescorresponding hardware support. Main memory offers fast storage that can be accessed directlybythe CPU. It is costlyand hence has a lower storage capacity. However, for a program to beexecuted,itmustbeinthemainmemory.



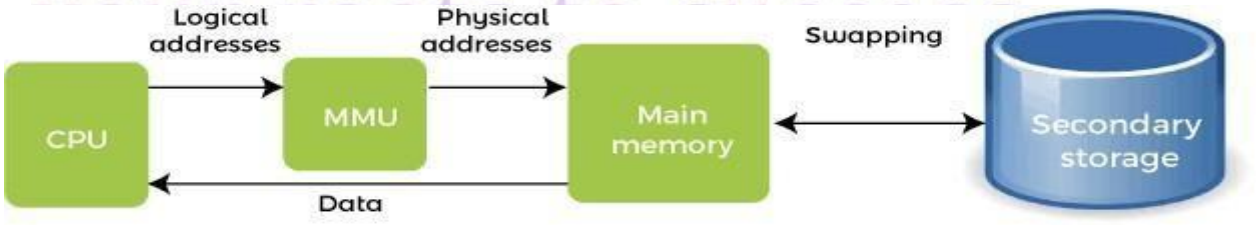
FunctionsofMemorymanagement

AnOperatingSystemperformsthefollowingfunctionsforMemoryManagementintheoperatingsystem:

- Ithelpsyoutokeeptrackofprimarymemory.
- Determinewhatpartofitareinusebywhom,whatpartisnotinuse.
- Inamultiprogrammingsystem,theOSdecideswhichprocesswillgetmemoryand howmuch.
- Allocatesthememorywhenaprocessrequests.
- Italsode-allocatesthememorywhenaprocessnolongerrequiresorhasbeenterminated.

5. Secondary-StorageManagement

The most important task of a computer system is to execute programs. These programs help youto access the data from the main memory during execution. This memory of the computer is verysmall to store all data and programs permanently. The computer system offers secondary storagetobackupthe mainmemory.



Todaymoderncomputersuseharddrives/SSDastheprimarystorageofboth

programsanddata.However,thesecondary storagemanagementalsoworkswith storage devices, such as USB flashdrivesandCD/DVDdrives.Programslikeassemblersandcompilersarestoredon thediskuntilitisloadedintomemory,andthenusethediskisusedasasourceanddestinationf orprocessing.

FunctionsofSecondarystoragemanagement

Herearesomemajorfunctionsofsecondarystoragemanagementintheoperatingsystem:

- Storageallocation
- Freespace management
- Diskscheduling
-

6. I/ODeviceManagement

One of the important use of an operating system that helps to hide the variations of specifichardware devicesfromtheuser.

FunctionsofI/Omanagement

TheI/Omanagementsystemoffersthefollowingfunctions,suchas:

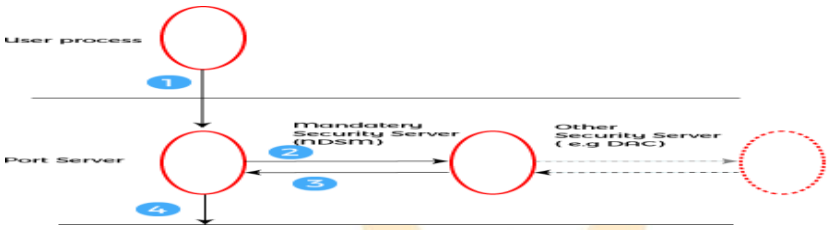
- Itoffersabuffercachingsystem
- Itprovidesgeneraldevicedrivercode
- Itprovidesdriversforparticularhardwaredevices.
- I/Ohelpsyoutoknowtheindividualitiesofaspecificdevice.

7. SecurityManagement

The various processes in an operating system need to be secured from other activities. Therefore,various mechanisms can ensure those processes that want to operate files, memory CPU, andotherhardwareresourcesshouldhaveproperauthorizationfromtheoperatingsystem.

Securityrefers to a mechanism for controlling the access ofprograms, processes, or users to

theresourcesdefinedbycomputercontrolstobeimposed,togetherwithsomemeansofenfo rcement.

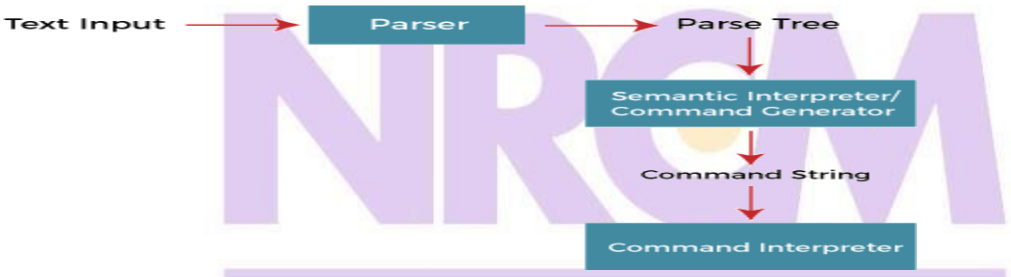


For example, memory addressing hardware helps to confirm that a process can be executedwithin its own address space. The time ensures that no process has control ofthe CPU withoutrenouncing it. Lastly, no process is allowed to do its own I/O to protect, which helps you to keepthe integrityofthevarious peripheraldevices.

Security can improve reliability by detecting latent errors at the interfaces between componentsubsystems.Earlydetection of interface errors can prevent thefoulness of ahealthysubsystembya malfunctioningsubsystem. Anunprotectedresourcecannot misuse by an unauthorized orincompetentuser.

9. CommandInterpreterSystem

One of the most important components of an operating system is its command interpreter.Thecommandinterpreteristheprimaryinterfacebetweentheuserandtherest ofthesystem.



Many commands are given to the operating system by control statements. A program that readsand interprets control statements is automatically executed when a new job is started in a batchsystemorauserlogsintoatime-sharedsystem.Thisprogramisvariouslycalled.

- Thecontrolcardinterpreter,
- Thecommand-lineinterpreter,
- Theshell(inUNIX),andsoon.

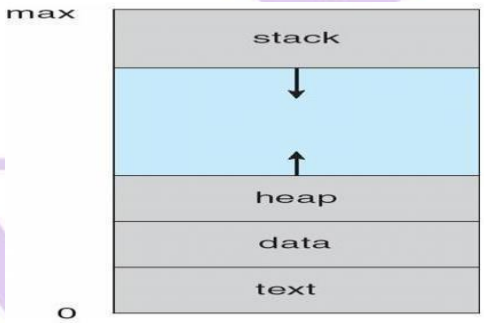
Its function is quite simple, get the next command statement, and execute it. The commandstatements deal with process management, I/O handling, secondarystorage management, mainmemorymanagement,file systemaccess,protection,andnetworking.

PROCESS: A process can be thought of as a program in execution. A process is the unit of workin most systems.

A process will need certain resources—such as CPU time, memory, files, and I/O devices to accomplish its task. These resources are allocated to the process either when it is created or while it is executing.

StructureofaProcessinMemory

- Aprocessismore thanthe programcode,whichissometimesknownasthe**textsection**.
- Italsoincludesthecurrentactivity,asrepresentedbythevalueofthe**programcounter** andthecontentsofthe processor'sregisters.
- A process generally alsoincludes the process **stack**, which contains temporary data (suchasfunction parameters, return addresses, and local variables).
- A**data section**,whichcontainsglobalvariables.
- Aprocessmayalsoincludea**heap**,whichismemorythatisdynamicallyallocated during process run time.



WhenaProgrambecomesProcess?

A program is a ***passive*** entity, such as a file containing a list of instructions stored on disk (Oftencalled as **executable file**). Incontrast, a process is an ***active*** entity,witha programcounter specifying the nextinstructionto execute and a set ofassociated resources. Aprogrambecomes a process when an executable file is loaded into memory.

Two common techniques for loading executable files are double-clicking an icon representing the executable file and entering the name of the executable file on the command line (as in prog.exe or a.out).

Iftwoprocessesareassociatedwiththesameprogram,aretheysameordifferent?(Or) Explain if you run same program twice, what section would be shared in memory?

Althoughtwoprocessesmaybeassociatedwiththesameprogram,theyarenevertheless

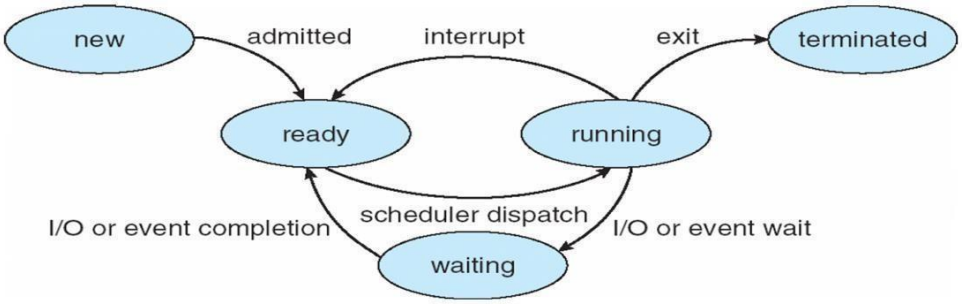
considered two separate execution sequences. For instance, severalusers may berunning different copies of the mail program, or the same usermay invoke many copies of the web browser program. Each of these is a separate process; and although the text sections are equivalent, the data, heap, and stack sections vary. It is also common to have a process that spawns many processes as it runs.

1. ProcessState

As a process executes, it changes **state**. The state ofa process is defined in part bythecurrent activity of that process.

Aprocessmaybeinoneofthefollowingstates:

- **New:**Theprocessisbeingcreated.
- **Running:**Instructionsarebeingexecuted.
- **Waiting**:Theprocessiswaitingforsomeeventtooccur(suchasanI/Ocompletionor reception of a signal).
- **Ready:**Theprocessiswaitingtobeassignedtoaprocessor.
- **Terminated**:Theprocesshasfinishedexecution.



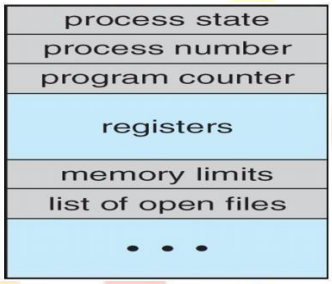
2. ProcessControlBlock

Eachprocess is represented inthe operating system bya **Process ControlBlock (PCB)** or **Task Control Block.** It contains many pieces of information associated with a specific process, including these:

- **Processstate:**Thestatemaybenew,ready,running,andwaiting,halted,andsoon.
- **Programcounter**. Thecounter indicatestheaddressofthe next instructionto beexecuted for this process.
- **CPU registers**. The registers vary in number and type, depending on the computer architecture. They include accumulators, index registers, stack pointers, and general- purpose registers, plus any condition-code information. Along with the program counter, this state information must be saved when an interrupt occurs, to allow the process to be continued correctly afterward.
- **CPU-scheduling information**. This information includes a process priority, pointers to scheduling queues, and any other scheduling parameters.
- **Memory-managementinformation**.Thisinformationmayincludesuchitemsasthevalue

ofthebaseandlimitregistersandthepagetables,orthesegmenttables,dependingon the memory system used by the operating system.

- **Accounting information**. This information includes the amount of CPU and real time used, time limits, account numbers, job or process numbers, and so on.
- **I/Ostatusinformation**.ThisinformationincludesthelistofI/Odevicesallocatedtothe process, a list of open files, and so on.



ProcessScheduling

Theobjectiveofmultiprogrammingistohavesomeprocessrunningatalltimes,to maximize CPU utilization.

TheobjectiveoftimesharingistoswitchtheCPUamongprocessesofrequentlythat users can interact with each programwhile itis running.

To meet these objectives, the **process scheduler** selects an available process (possiblyfrom aset of several available processes) for programexecution on the CPU.

1. Scheduling Queues

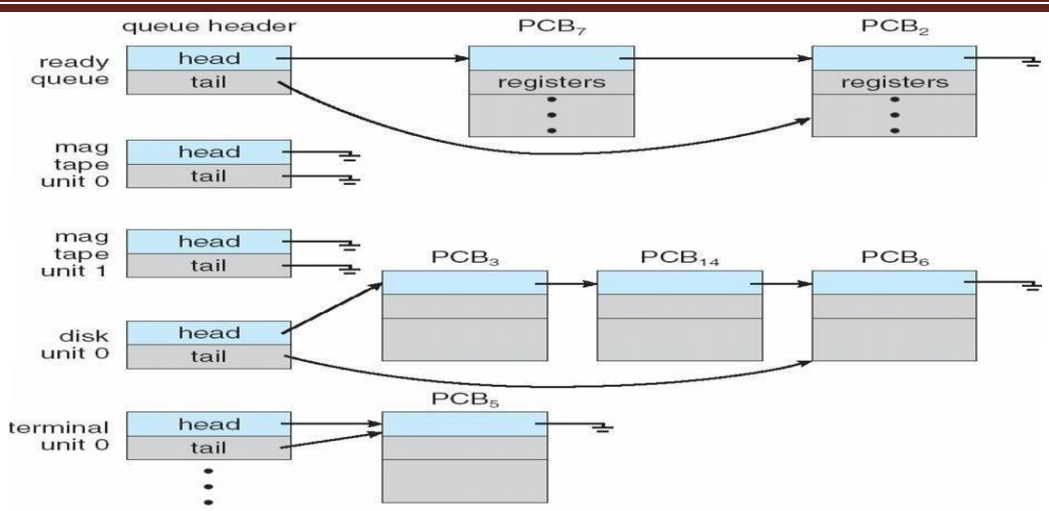
Thefollowingarethedifferentqueuesavailable,

a. JobQueue

- Asprocessesenterthesystem,theyareputintoa**jobqueue**,whichconsistsofall processes in the system.

b. Ready Queue

- The processes that areresiding in main memoryand are readyand waitingto executeare kept on a list called the **ready queue**.
- This queue is generally stored as a linked list. A ready-queue header containspointers to the first and final PCBs in the list. Each PCB includes a pointer field thatpoints to the next PCB in the ready queue.



c. DeviceQueue

- Thelistofprocesseswaitingfor aparticular I/Odeviceiscalleda**devicequeue**.
- Eachdevice hasitsowndevicequeue.

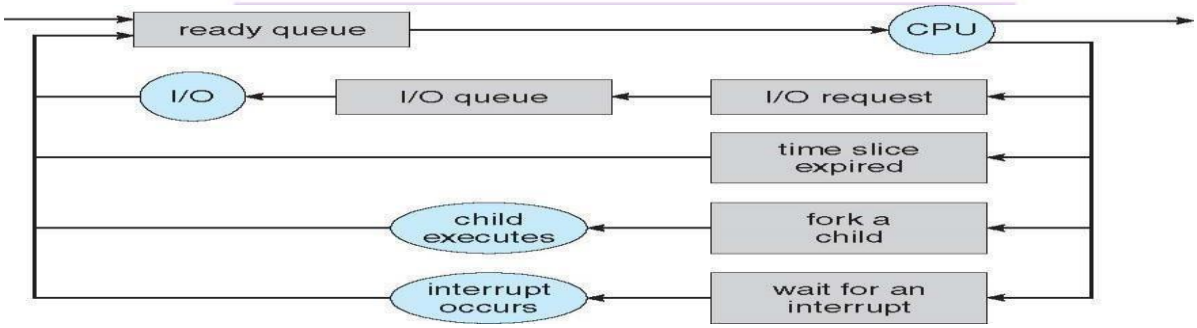
Queuing-diagramrepresentationofprocessscheduling

Acommonrepresentationofprocessschedulingisa**queuingdiagram**.Eachrectangular box represents aqueue. Twotypes of queuesare present: theready queueanda set of device queues. The circles represent the resources that serve the queues, and the arrows indicate the flow of processes in the system.

A new process is initially put in the ready queue. It waits there until it is selected for execution, or **dispatched**. Once the process is allocated the CPU and is executing, one of several events could occur:

- TheprocesscouldissueanI/OrequestandthenbeplacedinanI/Oqueue.
- Theprocesscouldcreateanewchildprocessandwaitforthechild'stermination.
- The process could be removed forcibly from the CPU, as a result of an interrupt, andbeput back in the ready queue.

In the first two cases, the process eventually switches from the waiting state to the ready state and is then put back in the readyqueue. A process continues this cycle until it terminates, at which time it is removed fromall queues and has its PCB and resources deallocated.



2. Schedulers

Definition: A process migrates among the various scheduling queues throughout its lifetime. The operating system must select, for scheduling purposes, processes from these queues in some fashion. The selection process is carried out bythe appropriate **scheduler**.

TypesofSchedulers

a. Long-TermScheduleror**JobScheduler**

- Often, in a batch system, more processes are submitted than can be executedimmediately. These processes are spooled toa mass-storage device (typically a disk), where they are kept for later execution.
- The **long-term scheduler**, or **job scheduler**, selects processes from this pool and loads them into memory for execution.
- The long-term scheduler executes much less frequently; minutes may separate the creation of one new process and the next.
- The long-term scheduler controls the **degree of multiprogramming** (the number of processes in memory).
- Ifthedegreeofmultiprogramming isstable, thentheaveragerateofprocesscreationmust be equal to the average departure rate of processes leaving the system. Thus, the long- term scheduler may need to be invoked only when a process leaves the system.
- Because of the longer interval between executions, the long-term scheduler can afford to take more time to decide which process should be selected for execution.It is important that the long-term scheduler select a good ***process mix*** of I/O-bound and CPU-bound processes.
- Onsomesystems,thelong-termschedulermaybeabsent orminimal.

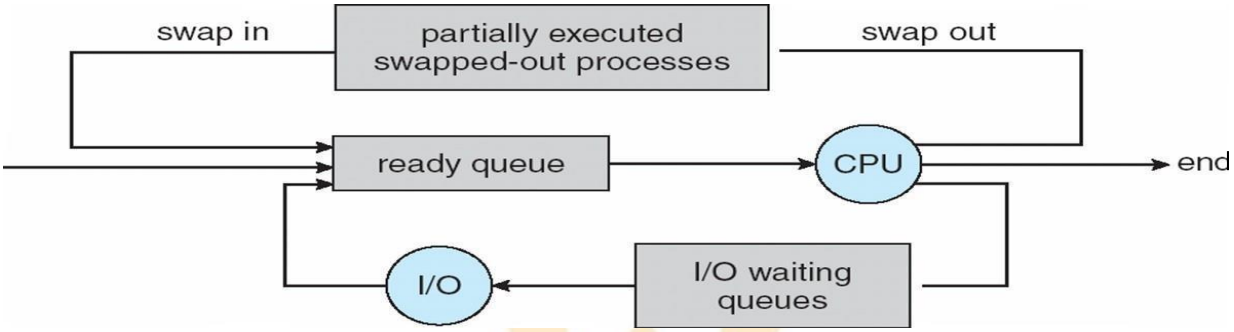
b. Short-TermScheduler,Or**CPUScheduler**

- The **short-termscheduler**, or**CPU scheduler**, selectsfromamong the processes that are readyto execute and allocates the CPU to one of them.
- Theshort-termschedulermustselectanewprocessfortheCPUfrequently.
- AprocessmayexecuteforonlyafewmillisecondsbeforewaitingforanI/Orequest. Often, the short-term scheduler executes at least once every 100 milliseconds.
- Becauseoftheshorttimebetweenexecutions,theshort-termschedulermustbefast.

c. Medium-TermScheduler

- Some operating systems, such as time-sharing systems, may introduce an additional, intermediate level of scheduling.
- The key idea behind a medium-term scheduler is that sometimes it can be advantageousto remove a process from memory (and from active contention for the CPU) and thus reduce the degree of multiprogramming.
- Later, the process can be reintroduced into memory, and its execution can be continued where it left off. This scheme is called **swapping**.
- The process is swapped out, and is later swapped in, by the medium-term scheduler. Swappingmay be necessary to improve the process mix or because a change inmemory

requirementshasovercommittedavailablememory,requiringmemorytobe freedup.



3. Context Switch

Definition: Switching the CPU to another process requires performing a state save of the current process and a state restoreof a different process. This task is known as a **context switch**.

Whenacontextswitchoccurs,thekernelsavesthecontextoftheoldprocessinitsPCB and loads the saved context ofthe new process scheduled to run.

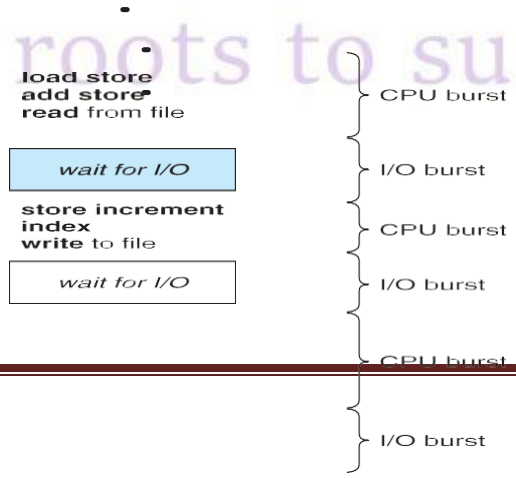
Overhead: Context-switch time is pure overhead, because the system does no useful workwhile switching.

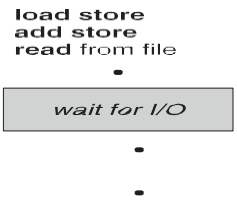
Switching Speed: Switching speed varies from machine to machine, depending on the memory speed, the number of registers that must be copied, and the existence of special instructions (such as a single instruction to load or store all registers). A typical speed is a fewmilliseconds.

Hardware Support: Context-switch times are highly dependent on hardware support. A context switchhere simply requires changing the pointer to the current register set. Of course,if there are moreactiveprocessesthantherearеregister sets, thesystemresortsto copying register datato and from memory, as before. Also, the more complex the operating system, the greater the amount of work that must be done during a context switch

4. CPU–I/OBurstCycle

The success of CPU scheduling depends on an observed property ofprocesses: process execution consists of a **cycle** of CPU execution and I/O wait. Processes alternate between these two states.Processexecutionbeginswitha **CPU burst**. That isfollowed byan**I/O burst**, which is followed byanother CPU burst, thenanother I/Oburst, and so on. Eventually, the finalCPU burst ends with a system request to terminate execution.





Definition ofNonPreemptiveScheduling

Under nonpreemptive scheduling, once the CPU has been allocated to a process, the process keeps the CPU until it releases the CPU either by terminating or by switching to the waiting state. This scheduling method was used by Microsoft Windows 3.x.

DefinitionofPreemptiveScheduling

Under this, a running process may be replaced by higher priority process at any time. Used from Windows 95 to till now. Incurs the cost associated with access to shared data. It also affects the design of OS.

Dispatcher

Another component involved in the CPU-scheduling function is the **dispatcher**. The dispatcher is the module that gives control of the CPU to the process selected by the short- term scheduler. This function involves the following:

- Switchingcontext
- Switchingtousermode
- Jumpingtotheproper locationintheuser programtorestartthatprogram

Thedispatchershouldbeasfastaspossible,sinceitisinvokedduringeveryprocess switch.

DispatchLatency:Thetimeittakesforthedispatchertostop oneprocessandstartanotherrunning is known as the **dispatch latency**.

Operationsonprocesses(OR)Systemcallinterfaceforprocessmanagement-fork,exit,wait, waitpid,exec

The processes in most systems can execute concurrently, and they may be created and deleted dynamically. Thus, these systems must provide a mechanism for process creation and termination.

a. ProcessCreation

During the course of execution, a process may create several new processes. The creating process is called a parent process, and the new processes are called the children of that process. Each of these new processes may in turn createother processes, forming a **tree** ofprocesses.

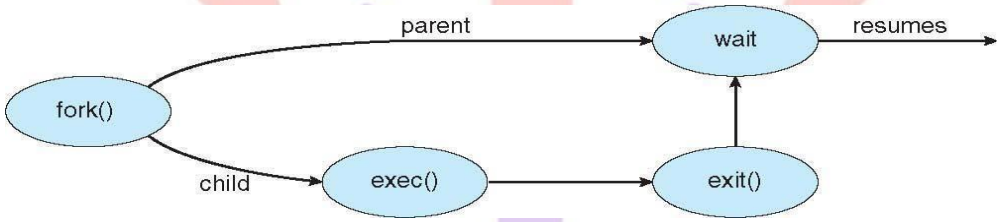
SystemCalls

- **fork()**
  - Most operating systems (including UNIX, Linux, and Windows) identify processes according to a unique **process identifier** (or **pid**), which is typically an integer number.

- Anew process iscreated bythe fork () systemcall. The new processconsistsofa copyof the address space of the original process.
- This mechanism allows the parent process to communicate easily with its child process. Both processes (the parent and the child) continue execution at the instruction after the fork (), with one difference: thereturn code for thefork () is zero for the new (child) process, whereas the (nonzero) process identifier of the childis returned to the parent.

- **exec()**
  - After a fork () system call, one of the twoprocesses typically uses the exec () system callto replace the process's memoryspace with a new program.
  - The exec () system call loads a binary file into memory and starts its execution. In this manner, thetwo processes are able to communicate and then go their separate ways.

- **wait()**
  - The parent can then create more children; or, if ithas nothing else to do while the child runs, it can issue a wait () system call to move itself off the ready queue until the termination of the child. Because the call to exec () overlays the process's address space with a new program, the call to exec () does not return control unless an error occurs.



b. ProcessTermination

A process terminates when it finishes executing its final statement and asks the operating system to delete it by using the exit () system call. At that point, the process may return a status value (typicallyan integer) to its parent process (via the wait() systemcall). Allthe resources of the process—including physical and virtual memory, open files and I/O buffers—are deallocated by the operating system.

Termination can occur in other circumstances as well. A process can cause thetermination of another process via an appropriate system call (for example, TerminateProcess() in Windows). Usually, such a system call can be invoked only by the parent of the process that is to be terminated. Otherwise, users could arbitrarily kill each other's jobs.

COOPERATINGPROCESSES

- TheconcurrentprocessexecutingintheOSmaybeeitherindependent process or cooperating process.
- Independentprocess**cannot**affectorbeaffectedby theexecutionofanother process.
- Cooperatingprocess**can**affectorbeaffectedbytheexecutionofanother

process.

Advantagesofprocesscooperation

1. **Informationsharing:**severalusersmaybeinterestinthesamepieceof information.
2. **Computationspeed-up:**Ifwewantaparticulartasktorunfaster,wemust break it into subtasks and run in parallel.
3. **Modularity:**Constructingthesystem inmodularfashion,dividingthesystem functions into separate process.
4. **Convenience:**Userwillhavemanytaskstoworkinparallel(Editing, compiling, printing).

Processescancommunicatewitheachotherthroughboth:

- SharedMemory
- Messagepassing

The following figure shows a basic structure of communicationbetweenprocesses via the shared memory method and via the message passing method.

Figure 1 - Shared Memory and Message Passing

(i) SharedMemory

Communication between processes using shared memory requires processestosharesomevariable, and it completelydependsonhowtheprogrammer willimplement it.

One way of communication using shared memory can be imagined like this: Suppose process1 and process2 are executing simultaneously, and theyshare some resources or use some information from another process. Process1 generates informationabout certaincomputationsorresources beingusedand keeps it as arecord in shared memory. Whenprocess2 needs to use the shared information, it willcheck in the record stored in shared memory and take note of the information generated by process1 and act accordingly.

Processes can use shared memory for extracting information as a record from another process as well as for delivering any specific information to other processes.

Ex:Producer-Consumerproblem

Aproducerprocessproducesinformationthatisconsumedbyaconsumer

process. For example, a print program produces characters that are consumed by the printer driver.

A producer can produce one item while the consumer is consuming another item. The Producer and Consumer must be synchronized. The consumer does not tryto consume an item, the consumer must wait until an item is produced.

Unbounded-Buffer

- nopractical limitonthesizeofthe buffer.
- Producer canproduceanynumberof items.
- Consumermayhavetowait

Bounded-Buffer

- assumesthatthereisafixedbuffer size.

Bounded-Buffer–Shared-MemorySolution:

Shareddata

```
#define BUFFER_SIZE 10
Typedef struct
{
. . .
}item;
itembuffer[BUFFER_SIZE];
int in = 0;
intout=0;
```

Bounded-Buffer–ProducerProcess:

```
itemnextProduced; while
(1)
{
 while(((in+1)%BUFFER_SIZE)==out);          /*donothing
 */buffer[in]=nextProduced;
 in= (in+ 1)%BUFFER_SIZE;
}
```

Bounded-Buffer–ConsumerProcess:

```
itemnextConsumed;
while (1)
{
 while(in==out);                            /*donothing
 */next Consumed = buffer[out];
 out=(out+1)%BUFFER_SIZE;
}
```

(ii) MessagingPassingMethod

In this method, processes communicate with eachother without usinganykind of shared memory.Iftwo processeswant tocommunicatewitheachother, theyproceed as follows

:

- Establish a communication link (if a link alreadyexists,noneedto establish it again.)
- Startexchangingmessagesusingbasicprimitives.
- Themessagesizecanbeoffixedsizeorofvariablesize.Ifitisoffixed size,itiseasy foran OSdesignerbutcomplicatedforaprogrammerandifit is of variable size then it is easy for a programmer but complicated for the OS designer.
- Cooperating process to communicate with each other via an inter process communication (IPC).
- IPC provides a Mechanism to allow processes to communicate and tosynchronize their actions.
- If P and Q want to communicate, a communication link exists between themand exchange messages via send/receive. OS provides this facility.
- IPCfacilityprovidestwooperations:

  Send(*message*)–messagesizefixedorvariable.

  Receive(*message*)

- Implementationofcommunicationlinkbyfollowing.
  - physical(e.g.,shared memory, hardware bus)
  - logical(e.g.,logicalproperties)

Methodsforlogicalimplementationofalink

i. Directcommunication.
ii. Indirectcommunication.

DirectCommunication

- Eachprocesses mustnameeachotherexplicitly:
  - Send(*P,message*)–sendamessagetoprocessP.
  - Receive(*Q,message*)–receiveamessagefromprocessQ.
- Linksareestablishedautomatically.

- Alinkisassociatedwithexactlyonepair ofcommunicating processes.
- Betweeneachpairthereexistsexactlyonelink.
- Thelink maybeunidirectional,but isusuallybi-directional.
- Thisexhibitsbothsymmetryand asymmetryinaddressing

  Symmetry:

  Boththesenderandthereceiverprocessesmustnametheotherto communicate.

  Asymmetry:

  Onlysendernamestherecipient,therecipientisnotrequiredtonamethe sender. The send and receive primitives are as follows.

  - Send(P, message)–sendamessagetoprocessP.
  - Receive(id,message)–receiveamessagefromanyprocess.

Disadvantageofdirectcommunication

Changing anameoftheprocesscreatesproblems.

IndirectCommunication

- Themessagesaresentandreceivedfrommailboxes(alsoreferredtoas ports).
- Amailbox isanobject
- Processcanplacemessages.
- Processcanremovemessages.
- Twoprocessescancommunicateonlyiftheyhaveasharedmailbox.
- Primitivesaredefinedas:

  send(*A,message*) –sendamessagetomailboxA

  receive(*A,message*)–receiveamessagefrommailbox A.
- AmailboxmaybeownedeitherbyaprocessorbytheOS.
- If the mailbox is owned by a process, then we distinguish b/w the owner (who can only receive msg through this mailbox) and the user (who can only send msg to the mailbox).
- Amailbox maybeownedbytheOSisindependentandprovideamechanism,
  - createamailbox
  - receivemessagesthroughmailbox
  - destroyamailbox.

Mailboxsharingproblem

TheprocessesP1,*P2*,and*P3*allsharemailboxA.*ProcessesP1*,sends;*P2* and*P3*receivethemessagefromA.Whogetsa message?

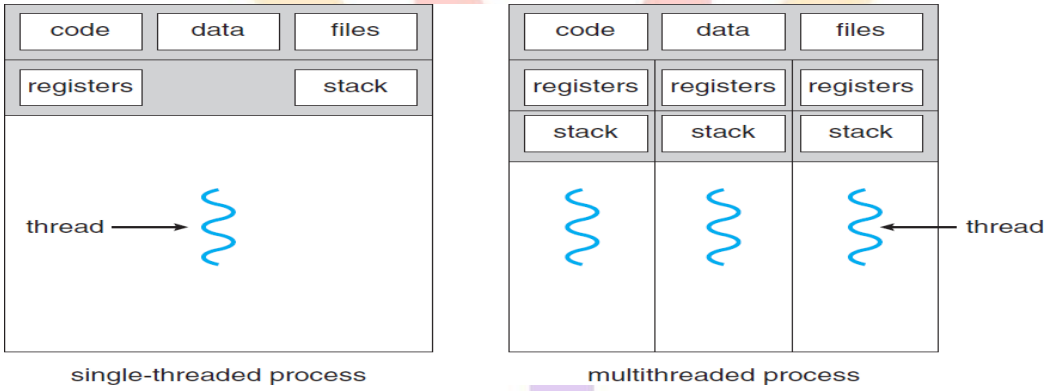
Solutions

- Allowalinktobeassociatedwithat mosttwoprocesses.
- Allowonlyoneprocessatatimetoexecutereceiveoperation.
- Allow the system to select arbitrarily the receiver. The system may identifythe receiver to the sender.

DefiningThread

Thread is a basic unit of CPU utilization; it comprises a thread ID, a program counter, a register set, and astack. It shares with other threads belonging to the same process its code section, data section, and other operating-system resources, such as open files and signals.

A traditional (or ***heavyweight***) process has a single thread of control. If a process has multiple threads ofcontrol,it can perform more than one task at atime.



SingleThread

- Aprocessisaprogramthatperformsasingle**thread**ofexecution.
- For example, when a process is running a word-processor program, a single thread of instructions is being executed.

- Thissinglethreadofcontrolallowstheprocesstoperformonlyonetaskata time. The user cannot simultaneously type in characters and run the spell checker within the same process, for example.Multi Thread
  - Most modernoperating systems have extended the process concept to allow a process tohave multiple threads ofexecution and thus to performmore than one task at a time.
  - This feature is especially beneficial on multicore systems, where multiple threads can runin parallel.
  - Onasystemthatsupportsthreads,thePCBisexpandedtoincludeinformationforeach thread. Other changes throughout the systemare also needed to support threads.

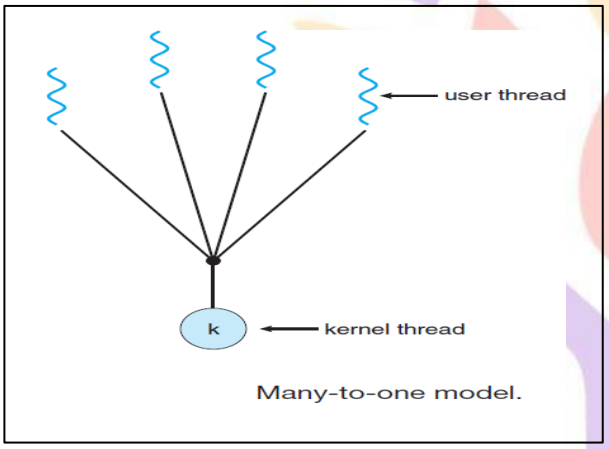
MultithreadingModels

Support for threads may be provided either at the user level, for **user threads**, or by the kernel,for**kernelthreads**.Userthreadsaresupportedabovethekernelandaremanagedwithout

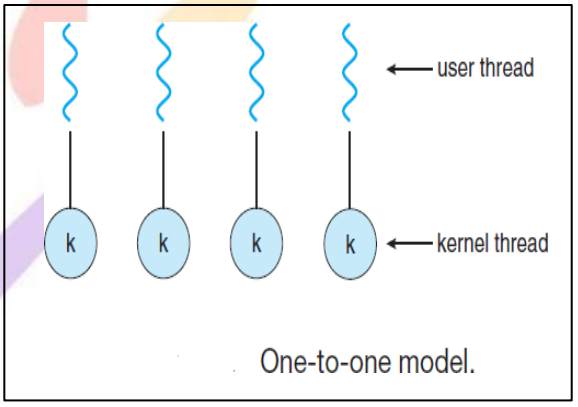
kernel support, whereas kernel threads are supported and managed directly by the operating system. Virtually all contemporary operating systems—including Windows, Linux, Mac OS X, and Solaris support kernel threads.

Ultimately, a relationship must exist between user threads and kernel threads. The following are the three common ways ofestablishing such a relationship: the many-to-one model, the one-to-one model, and the many-to many models.

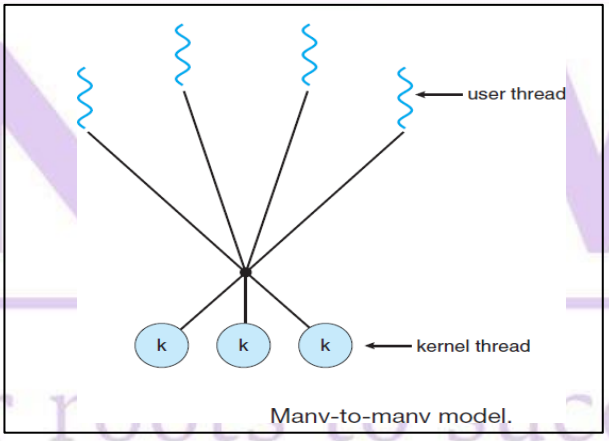
1. **Many-to-OneModel**
   - Themany-to-onemodelmapsmanyuser-levelthreadstoonekernelthread.
2. **One-to-OneModel**
   - Theone-to-onemodelmapseachuserthread toakernelthread.
3. **Many-to-ManyModel**
   - Itmultiplexesmanyuser-levelthreadsto asmallerorequalnumberofkernelthreads.



Many-to-one model.



One-to-one model.



Many-to-many model.