GANAPATI INSTITUTE OF ENGINEERING AND TECHNOLOGY
(POLYTECHNIC)

Mathasahi , Jagatpur , Cuttack - 754200, Odisha

"Operating System"

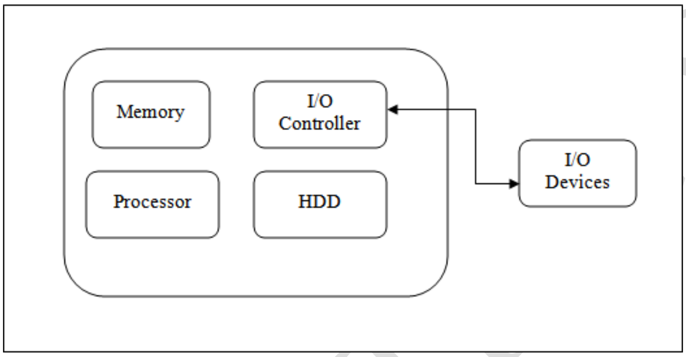
PREPARED BY:

Miss Smaranika Moharana.

OperatingSystem:

- Anoperating system is a program which manages all the computerhardwares.
- Itprovidesthebaseforapplicationprogramandactsasanintermediarybetweenauserandthe computer hardware.
- The operating system has two objectives such as:
  - Firstly,anoperatingsystemcontrolsthecomputer’shardware.
  - Thesecondobjectiveistoprovideaninteractiveinterfacetotheuserandinterpret commands so that it can communicate with the hardware.
- Theoperatingsystemisveryimportantpartofalmosteverycomputersystem.

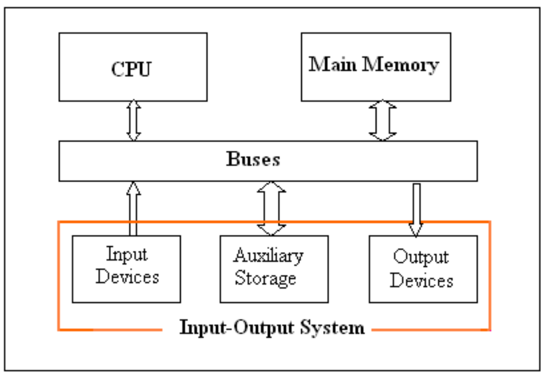
ManagingHardware



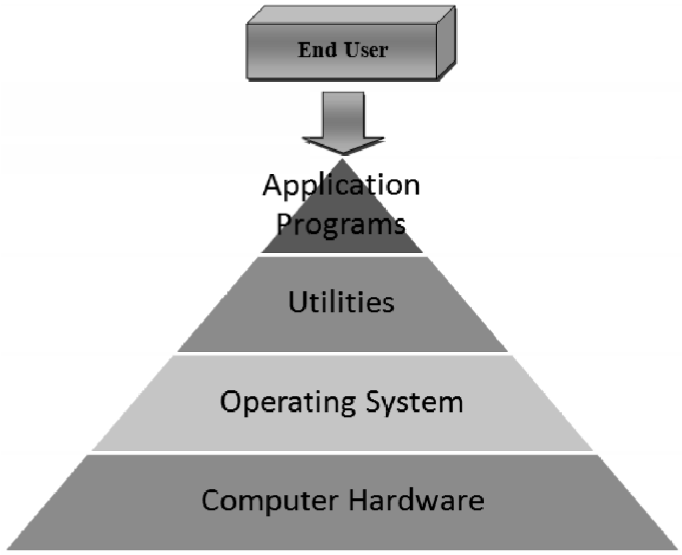
- Theprimeobjectiveofoperatingsystemistomanage&controlthevarioushardwareresources of a computer system.
- Thesehardwareresourcesincludeprocesser,memory,anddiskspaceandsoon.
- Theoutputresultwasdisplayinmonitor.Inadditiontocommunicatingwiththehardware theoperatingsystemprovidesonerrorhandlingprocedureanddisplayanerrornotification.
- Ifadevicenotfunctioningproperly,theoperatingsystemcannotbecommunicatewiththe device.

ProvidinganInterface

- Theoperatingsystemorganizesapplicationsothat users can easily access, use and store them.



- It provides a stable and consistent way for applications to deal with the hardware without the user having known details of the hardware.
- If the program is not functioning properly, the operating system again takes control, stops the application and displays the appropriate error message.
- Computersystemcomponentsaredividedinto5parts
  - Computerhardware
  - operatingsystem
  - utilities
  - Applicationprograms
  - Enduser



- Theoperatingsystemcontrolsandcoordinateauserofhardwareandvariousapplication programs for various users.
- Itisaprogramthatdirectlyinteractswiththehardware.
- Theoperatingsystemis thefirstencodedwiththeComputeranditremainsonthememory all time thereafter.

Systemgoals

- Thepurposeofanoperatingsystemistobeprovidedanenvironmentinwhichanusercan execute programs.
- Itsprimarygoalsaretomakethecomputersystemconveniencefortheuser.
- Itssecondarygoalsaretousethecomputerhardwareinefficientmanner.

Viewofoperatingsystem

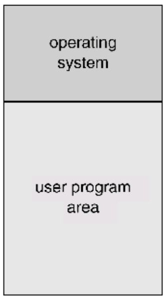
- **Userview:**Theuserviewofthe computervariesbytheinterfacebeingused.The examples are-windowsXP,vista,windows7etc.Mostcomputerusersitintheinfrontofpersonal computer (pc) in this case the operating system is designed mostly for easy use with some attention paid to resource utilization. Some user sit at a terminal connected to a mainframe/minicomputer.Inthiscaseotherusersareaccessingthesamecomputerthrough the other terminals. There user are share resources and may exchange the information. The operatingsysteminthiscaseisdesignedtomaximizeresourcesutilizationtoassumethatall availableCPUtime,memoryandI/Oareusedefficientlyandnoindividualusertakesmore than his/her fair and share.The other users sit at workstations connected to network of other workstations and servers. These users have dedicated resources but they share resources such as networking and servers like file, compute and print server. Here the operating systemis designedto compromise betweenindividual usability and resource utilization.
- **Systemview:** From the computer point of view the operating system is the program which ismostintermediatewiththehardware.Anoperatingsystemhasresourcesashardwareand softwarewhichmayberequiredtosolveaproblemlikeCPUtime,memoryspace,file storage space andI/Odevicesandsoon.That's why the operating system acts as manager oftheseresources.Anotherviewoftheoperatingsystemisitisacontrolprogram.Acontrol program manages the execution of user programs to present the errors in proper use of the computer. It is especially concerned of the user the operation and controls the I/O devices.

TypesofOperatingSystem

1. **MainframeSystem:**Itisthesystemwherethefirstcomputerusedtohandlemanycommercial scientificapplications.Thegrowthofmainframesystemstracedfromsimplebatchsystem where the computer runsone andonly oneapplicationtotime sharedsystems whichallowed for user interaction with the computer system
   a. **Batch /Early System:**Early computers were physically large machine.The common inputdevices werecardreaders,tapedrivers.Thecommonoutputdeviceswereline printers,tapedriversandcardpunches.Inthesesystemstheuserdidnotinteractdirectly with the computer system. Instead the user preparing a job which consists of programming data and some control information and then submitted it to the computer

operatoraftersometimetheoutputisappeared.Theoutputintheseearlycomputerwas fairlysimpleismaintaskwastotransfercontrolautomaticallyfromonejobtonext.The operatingsystemalwaysresidesinthememory.Tospeedupprocessingoperators batchedthejobswithsimilarneedsandranthentogetherasagroup.Thedisadvantages ofbatchsystemarethatinthisexecutionenvironmenttheCPUisoftenidlebecausethe speed up of I/O devices is much slower than the CPU.

Memory Layout for a Simple Batch System



b. **MultiprogrammedSystem:**MultiprogrammingconceptincreasesCPUutilizationby organization jobs so that the CPU always has one job to execute the idea behind multiprogrammingconcept.Theoperatingsystemkeepsseveraljobsinmemory simultaneously as shown in below figure.

Operating System
Job 1
Job 2
Job 3
Job 4

This set of job is subset of the jobs kept in the jobpool. The operatingsystem picks and beginning to execute one of the jobs in the memory. In this environment the operating systemsimplyswitchesandexecutes anotherjob.WhenajobneedstowaittheCPUis simply switchedtoanother jobandsoon. The multiprogramming operating systemis sophisticatedbecausetheoperatingsystemmakesdecisionsfortheuser.Thisisknownas scheduling.Ifseveraljobsarereadytorunatthesametimethesystemchooseoneamong them.ThisisknownasCPUscheduling.Thedisadvantagesofthemultiprogrammed system are

- It does not provide user interaction with the computer system during the program execution.
- The introduction of disk technology solved these problems rather than reading the cards from card reader into disk. This form of processing is known as spooling.

SPOOL stands for simultaneous peripheral operations online. It uses the disk as a huge buffer for reading from input devices and for storing output data until the output devices accept them. It is also use for processing data at remote sides. The remote processing is done and its own speed with no CPU intervention. Spooling overlaps the input,outputonejobwithcomputationofotherjobs.Spoolinghasabeneficialeffecton theperformanceofthesystemsbykeepingbothCPUandI/Odevicesworkingatmuch higher time.

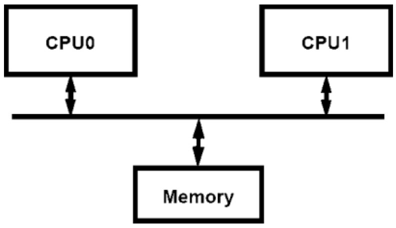
c. **TimeSharingSystem:**Thetimesharingsystemisalsoknownasmultiusersystems.The CPU executes multiple jobs by switching among them but the switches occurs so frequentlythattheusercaninteractwitheachprogramwhileitisrunning.Aninteractive computer system provides direct communication between a user and system. The user gives instruction to the operating systems or to a program directly using keyboard or mouse and wait for immediate results. So the response time will be short. The time sharingsystemallowsmanyuserstosharethecomputersimultaneously.Sinceeachaction inthissystemisshort,onlyalittleCPUtimeisneededforeachuser.Thesystemswitches rapidly from one user to the next so each user feels as if the entire computer system is dedicatedtohis use,eventhoughitis beingsharedbymanyusers.Thedisadvantagesof time sharing system are:

- Itismorecomplexthanmultiprogrammedoperatingsystem
- Thesystemmusthavememorymanagement&protection,sinceseveraljobsarekept in memory at the same time.
- Timesharingsystemmustalsoprovideafilesystem,sodiskmanagementisrequired.
- ItprovidesmechanismforconcurrentexecutionwhichrequirescomplexCPUscheduling schemes.

2. **Personal Computer System/Desktop System:** Personal computer appeared in 1970's. They are microcomputers that are smaller & less expensive than mainframe systems. Instead of maximizingCPU&peripheralutilization,thesystemsoptformaximizinguserconvenience& responsiveness. At first file protection was not necessary on a personal machine. But when other computers 2ndotherusers can access the fileson a pc file protection becomes necessary. The lack of protection made if easy for malicious programs to destroy data on such systems. Theseprogramsmaybeselfreplicating&theyspreadviawormorvirusmechanisms.Theycan disrupt entire companies or even world wide networks. E.g : windows 98, windows 2000, Linux.
3. **Microprocessor Systems/ Parallel Systems/ Tightly coupled Systems:**These Systemshave more than one processor in close communications which share the computer bus, clock, memory & peripheral devices. Ex: UNIX, LINUX. Multiprocessor Systems have 3 main advantages.
   a. **Increased throughput:** No. of processes computed per unit time. By increasing the no. ofprocessors moveworkcanbedone inlesstime.The speedupratiowithN processors is not N, but it is less than N. Because a certain amount of overhead is incurred in keeping all the parts working correctly.
   b. **IncreasedReliability:**Iffunctionscanbeproperlydistributedamongseveralprocessors, thenthefailureofoneprocessorwillnothaltthesystem,butslowitdown.Thisabilityto continue to operate in spite of failure makes the system fault tolerant.
   c. **Economic scale:** Multiprocessorsystems can save money as they can share peripherals, storage & power supplies.

Thevarioustypesofmultiprocessingsystemsare:

- **Symmetric Multiprocessing (SMP):** Each processor runs an identical copy of the operatingsystem&thesecopiescommunicatewithoneanotherasrequired.Ex:Encore's version of UNIX for multi max computer. Virtually, all modern operating system includingWindowsNT,Solaris,DigitalUNIX,OS/2&LINUXnowprovidesupportfor SMP.



- **Asymmetric Multiprocessing (Master – Slave Processors):** Each processor is designed for a specific task. A master processor controls the system & schedules & allocatestheworktotheslaveprocessors.Ex-Sun'sOperatingsystemSUNOSversion4 provides asymmetric multiprocessing.

4. **Distributed System/Loosely Coupled Systems:** In contrast to tightly coupled systems, the processorsdonotsharememoryoraclock.Instead,eachprocessorhasitsownlocalmemory. The processors communicate with each other by various communication lines such as high speed buses or telephone lines. Distributed systems depend on networking for their functionalities. By being able to communicate distributed systems are able to share computational tasks and provide a rich set of features to the users. Networks vary by the protocols used, the distances between the nodes and transport media. TCP/IP is the most commonnetworkprotocol.Theprocessorisadistributedsystemvariesinsizeandfunction.It may microprocessors, work stations, minicomputer, and large general purpose computers. Networktypesarebasedonthedistancebetweenthenodessuchas LAN(withinaroom,floor or building) and WAN (between buildings, cities or countries). The advantages of distributed systemareresourcesharing,computationspeedup,reliability,communication.

5. **Real time Systems:** Real time system is used when there are rigid time requirements on the operation of a processor or flow of data. Sensors bring data to the computers. The computer analyzesdataandadjustscontrolstomodifythesensorsinputs.Systemthatcontrolsscientific experiments, medical imaging systems and some display systems are real time systems. The disadvantages of real time system are:

   a. A real time system is considered to function correctly only if it returns the correct result within the time constraints.

   b. Secondary storage is limited or missing instead data is usually stored in short term memory or ROM.

   c. AdvancedOS featuresareabsent.

   Realtimesystemisoftwotypessuchas:

- **Hardrealtimesystems:**Itguaranteesthatthecritical taskhasbeencompletedontime. The sudden task is takes place at a sudden instant of time.
- **Softrealtimesystems:**Itisalessrestrictivetypeofrealtimesystemwhereacriticaltask getspriorityoverothertasksandretainsthatpriorityuntilitcomputes.Thesehavemore limitedutilitythan hardreal time systems. Missing anoccasional deadlineis acceptable e.g.QNX,VXworks.Digitalaudioormultimediaisincludedinthiscategory.

It is a special purpose OS in which there are rigid time requirements on the operation of a processor. A real time OS has well defined fixed time constraints. Processing must be done withinthetimeconstraintorthesystemwillfail.Arealtimesystemissaidtofunctioncorrectly only if it returns the correct result within the time constraint. These systems are characterized by having time as a key parameter.

BasicFunctionsofOperation System:

Thevariousfunctionsofoperatingsystemareasfollows:

1. **ProcessManagement:**

- Aprogramdoesnothingunlesstheirinstructionsareexecutedbya CPU.Aprocessisaprogram in execution. A time shared user program such as a complier is a process. A word processing program being run by an individual useron a pc is a process.
- A system task such as sending output to a printer is also a process. A process needs certain resources includingCPU time,memory files &I/Odevicestoaccomplishitstask.
- Theseresources are either given to the process when it is created or allocated to it while it is running.TheOSisresponsibleforthefollowingactivitiesofprocessmanagement.
- Creating&deletingbothuser&systemprocesses.
- Suspending&resumingprocesses.
- Providing mechanismforprocess synchronization.
- Providing mechanismforprocess communication.
- Providingmechanismfordeadlockhandling.

2. **MainMemoryManagement:**

The main memory is central to the operation of a modern computer system. Main memory is a large array of words or bytes ranging in size from hundreds of thousand to billions. Main memory stores the quickly accessible data shared by the CPU &I/Odevice. The central processorreadsinstructionfrommainmemoryduringinstructionfetchcycle&itbothreads

&writesdatafrommainmemoryduringthedatafetchcycle.Themainmemoryisgenerallythe onlylargestoragedevicethattheCPUisabletoaddress&accessdirectly.Forexample,forthe CPU to process data from disk. Those data must first be transferred to main memory by CPU generated E/O calls. Instruction must be in memory for the CPU to execute them. The OS is responsiblefor thefollowingactivitiesinconnectionwithmemorymanagement.

- Keepingtrackofwhichpartsofmemoryarecurrentlybeingused&bywhom.
- Decidingwhichprocessesaretobeloadedintomemorywhenmemoryspacebecomes available.
- Allocating&deallocatingmemoryspaceasneeded.

3. **FileManagement:**

File management is one of the most important components of an OS computer can store informationonseveraldifferenttypesofphysicalmediamagnetictape,magneticdisk&optical disk are the most common media.Each medium is controlled by a device such as disk drive or tapedrivethosehasuniquecharacteristics.Thesecharacteristicsincludeaccessspeed,capacity, data transfer rate & access method (sequential or random).For convenient use of computer system the OS provides a uniform logical view of information storage. The OS abstracts from the physical properties of its storage devices to define a logical storage unit the file. A file is collectionofrelatedinformationdefinedbyitscreator.TheOSisresponsibleforthefollowing activities of file management.

- Creating&deleting files.
- Creating&deleting directories.
- Supportingprimitivesformanipulatingfiles&directories.
- Mappingfilesintosecondarystorage.
- Backingupfilesonnon-volatilemedia.

4. **I/OSystemManagement:**

One of the purposes of an OS is to hide the peculiarities of specific hardware devices from the user.Forexample,inUNIXthe peculiarities ofI/Odevicesare hiddenfromthe bulkofthe OS itself by the I/O subsystem. The I/O subsystem consists of:

- Amemorymanagementcomponentthatincludesbuffering,catching&spooling.
- A general device- driver interfaces drivers for specific hardware devices. Only the device driverknowsthepeculiaritiesofthespecificdevicetowhichitisassigned.

5. **SecondaryStorageManagement:**

The main purpose of computer system is to execute programs. These programs with the data they access must be in main memory during execution. As the main memory is too small to accommodate all data & programs & because the data that it holds are lost when power is lost. The computer system must provide secondary storage to back-up main memory. Most modern computer systems are disks as the storage medium to store data & program. The operating systemisresponsibleforthefollowingactivitiesofdiskmanagement.

- Freespacemanagement.
- Storageallocation.
- Diskscheduling

Becausesecondarystorageisusedfrequentlyitmustbeusedefficiently.

Networking:

A distributed system is a collection of processors that don't share memory peripheral devices or a clock. Each processor has its ownlocal memory & clockand the processor communicate with one anotherthroughvariouscommunicationlinessuchashighspeedbusesornetworks.Theprocessors inthesystemareconnectedthroughcommunicationnetworkswhichareconfiguredinanumberof different ways. The communication network design must consider message routing & connection strategies are the problems of connection & security.

Protectionorsecurity:

If a computer system has multi users & allow the concurrent execution of multiple processes then the various processes must be protected from one another's activities. For that purpose, mechanismsensurethatfiles,memorysegments,CPU&otherresourcescanbeoperatedonbyonly those processes that have gained proper authorization from the OS.

Commandinterpretation:

One of the most important functions of the OS is connected interpretation where it acts as the interface between the user & the OS.

SystemCalls:

System calls provide the interface between a process & the OS. These are usually available in the formofassemblylanguageinstruction.Somesystemsallowsystemcallstobemadedirectlyfroma high level language program like C, BCPL and PERL etc. systems calls occur in different ways dependingonthecomputerinuse.Systemcallscanberoughlygroupedinto5majorcategories.

1. **ProcessControl:**

- **End,abort:**Arunningprogramneedstobeabletohasitsexecutioneithernormally(end)or abnormally (abort).
- **Load, execute:**A process or job executing one program may want to load and executes another program.
- **Create Process, terminate process:** There is a system call specifying for the purpose of creatinganewprocessorjob(createprocessorsubmitjob).Wemaywanttoterminateajob or process that we created (terminates process, if we find that it is incorrect or no longer needed).
- **Getprocessattributes,setprocessattributes:**Ifwecreateanewjoborprocessweshould able to control its execution. This control requires the ability to determine & reset the attributes of a job orprocesses (get process attributes, set process attributes).
- **Waittime:**Aftercreatingnewjobsorprocesses,wemayneedtowaitforthemtofinishtheir execution (wait time).
- **Waitevent,signalevent:**Wemaywaitforaspecificeventtooccur(waitevent).Thejobsor processes thensignal when that event has occurred(signal event).

2. **FileManipulation:**

- **Createfile,deletefile:**Wefirstneedtobeabletocreate&deletefiles.Boththesystemcalls require the name of the file & some of its attributes.
- **Open file, close file:** Once the file is created, we need to open it & use it. We close the file when we are no longer using it.
- **Read, write, reposition file:** After opening, we may also read, write or reposition the file (rewind or skip to the end of the file).
- **Get file attributes, set file attributes:** For either files or directories, we need to be able to determinethevaluesofvariousattributes&resetthemifnecessary.Twosystemcallsgetfile attribute & set file attributes are required for their purpose.

3. **DeviceManagement:**

- **Request device, release device:** If there are multiple users of the system, we first request thedevice.Afterwefinishedwiththedevice,wemustreleaseit.
- **Read,write,reposition:**Oncethedevicehasbeenrequested&allocatedtous,wecanread, write & reposition the device.

4. **Informationmaintenance:**

- **Get time or date, set time or date:**Most systems have a system call to return the currentdate & time or set the current date & time.
- **Getsystemdata,setsystemdata:**Othersystemcallsmayreturninformationaboutthe system like number of current users, version number of OS, amount of free memory etc.
- **Getprocessattributes,setprocessattributes:**TheOSkeepsinformationaboutallits processes& therearesystem callstoaccess thisinformation.

5. **Communication:**Therearetwomodesofcommunicationsuchas:

- **Message passing model:** Information is exchanged through an inter processcommunication facility provided by operating system. Each computer in a network has a namebywhichitisknown.Similarly,eachprocesshasaprocessnamewhichistranslatedto an equivalent identifier by which the OS can refer to it. The get hostid and get processed systems calls to do this translation. These identifiers are then passed to the general purpose open&closecallsprovidedbythefilesystemortospecificopenconnectionsystemcall.The recipient process must give its permission for communication to take place with an accept connection call. The source of the communication known as client & receiver known as server exchange messages by read message & write message system calls. The close connection call terminates the connection.
- **Shared memory model:**processes use map memory system calls to access regions of memoryownedbyotherprocesses.Theyexchangeinformationbyreading&writingdatain the shared areas. The processes ensure that they are not writing to the same location simultaneously.

SYSTEMPROGRAMS:

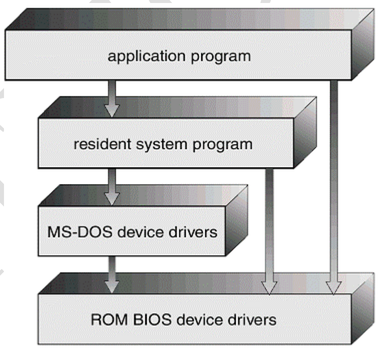
System programs provide a convenient environment for program development & execution. They are divided into the following categories.

- **File manipulation:**These programs create, delete, copy, rename, print & manipulate files and directories.
- **Status information:** Some programs ask the system for date, time & amount of available memory or disk space, no. of users or similar status information.
- **Filemodification:**Severaltexteditorsareavailabletocreateandmodifythecontentsoffile stored on disk.

- **Programming language support:**compliers, assemblers & interpreters are provided to the user with the OS.
- **Programmingloadingandexecution:**Onceaprogramisassembledorcompiled,itmust be loaded into memory to be executed.
- **Communications:**Theseprogramsprovidethemechanismforcreatingvirtualconnections among processes users 2nd different computer systems.
- **Application programs:**Most OS are supplied with programs that are useful to solve common problems or perform common operations. Ex: web browsers, word processors & text formatters etc.

Systemstructure:

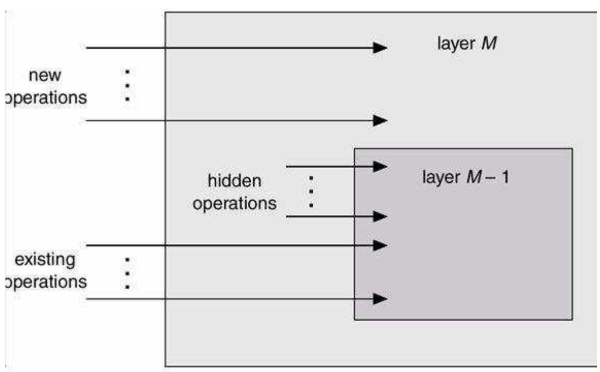
1. **Simplestructure:** There are several commercial system that don't have a well- defined structure such operating systems begins as small, simple & limited systems and then grow beyondtheiroriginalscope.MS-DOSisanexampleofsuchsystem.Itwasnotdividedinto modulescarefully.AnotherexampleoflimitedstructuringistheUNIXoperatingsystem.



(MSDOSStructure)

2. **Layered approach:**In the layered approach, the OS is broken into a number of layers (levels)eachbuiltontopoflower layers.The bottom layer(layero)is thehardware & top most layer (layer N) is the user interface. Themainadvantageofthelayeredapproach is modularity.

- The layers are selected such that each users functions (or operations) & services of only lower layer.



- This approach simplifies debugging & system verification, i.e. the first layer can be debugged without concerning the rest of the system. Once the first layer is debugged, its correct functioningisassumedwhilethe2ndlayerisdebugged&soon.
- If an error is found during the debugging of a particular layer, the error must be on that layer because the layers below it are already debugged. Thus the design & implementation of the systemaresimplifiedwhenthesystemisbrokendownintolayers.
- Eachlayerisimplementedusingonlyoperationsprovidedbylowerlayers.Alayerdoesn't need toknowhowtheseoperationsareimplemented;itonlyneedstoknowwhattheseoperationsdo.
- Thelayerapproachwasfirstusedintheoperatingsystem.Itwasdefinedinsixlayers.

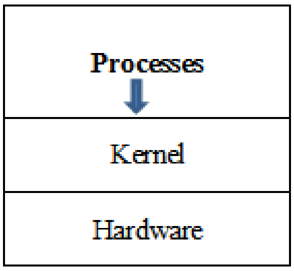
Layers	Functions
5	UserProgram
4	I/OManagement
3	OperatorProcess Communication
2	MemoryManagement
1	CPUScheduling
0	Hardware

Themaindisadvantageofthelayeredapproachis:

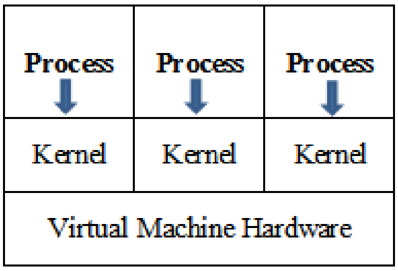
- The main difficulty with this approach involves the careful definition of the layers, becausealayercanuseonlythoselayersbelowit.Forexample,thedevicedriverforthe disk space used by virtual memory algorithm must be at a level lower than that of the memorymanagementroutines,becausememorymanagementrequirestheabilitytouse the disk space.
- Itislessefficientthananonlayeredsystem(Eachlayeraddsoverheadtothesystemcall &thenetresultisasystemcallthattakelongertimethanonanonlayeredsystem).

VirtualMachines:

ByusingCPUscheduling&virtualmemorytechniquesanoperatingsystemcancreatetheillusion ofmultipleprocesses,eachexecutingonitsownprocessors&ownvirtualmemory.Eachprocessor is provided a virtualcopyof the underlying computer. The resources ofthe computer are shared to create the virtual machines. CPU scheduling can be used to create the appearance that users have their own processor.



(Non virtual Machine)



(Virtual Machine)

Implementation: Although the virtualmachine concept is useful, itis difficultto implementsince mucheffort isrequired toprovidean exactduplicate of theunderlyingmachine. TheCPU is being multiprogrammed among several virtual machines, which slows down the virtual machines in various ways.

Difficulty: A major difficulty with this approach is regarding the disk system. The solution is to providevirtualdisks,whichareidenticalinallrespectsexceptsize.Theseareknownasminidisksin IBM'sVMOS.Thesumofsizesofallminidisksshouldbelessthantheactualamountofphysical disk space available.

I/OStructure

A general purpose computer system consists of a CPU and multiple device controller which is connectedthroughacommonbus.Eachdevicecontrollerisinchargeofaspecifictypeofdevice.A device controller maintains some buffer storage and a set of special purpose register. The device controllerisresponsible formovingthedatabetweenperipheraldevicesandbufferstorage.

I/O Interrupt: To start an I/O operation the CPU loads the appropriate register within the device controller.Inturnthedevicecontrollerexaminesthecontentoftheregistertodeterminetheactions which will be taken. For example, suppose the device controller finds the read request then, the controller will start the transfer of data from the device to the buffer. Once the transfer of data is complete thedevice controller informs theCPU that the operationhas been finished. Oncethe I/O is started, two actions are possible such as

- InthesimplestcasetheI/OisstartedthenatI/Ocompletioncontrolisreturntotheuser process.ThisisknownassynchronousI/O.

- Theotherpossibilityis asynchronous I/O inwhichthecontrol is returntotheuserprogram withoutwaitingfortheI/Ocompletion.TheI/Othencontinueswithotheroperations.

When an interrupt occurs first determine which I/Odevice is responsible for interrupting. After searching the I/Odevice table the signal goes to the each I/Orequest. If there are additional request waiting inthe queue forone device the operating systemstarts processing the nextrequest. Finally control is return from the I/O interrupt.

DMA controller: DMA is used for high speed I/O devices. In DMA access the device controller transfers on entire block of data to of from its own buffer storage to memory. In this access the interrupt is generated per block rather than one interrupt per byte. The operating system finds a buffer from the pool of buffers for the transfer. Then a portion of the operating system called a devicedriversetstheDMAcontrollerregisterstouseappropriatesource anddestinationaddresses and transfer length. The DMA controller is then instructed to start the I/O operation. While the DMA controller is performing the data transfer, the CPU is free to perform other tasks. Since the memory generally can transfer only one word at a time, the DMA controller steals memory cycles from the CPU. This cycle stealing can slow down the CPU execution while a DMA transfer is in progress. The DMA controller interrupts the CPU when the transfer has been completed.

StorageStructure

Thestoragestructureofacomputersystemconsistsoftwotypesofmemorysuchas

- Mainmemory
- Secondarymemory

Basicallytheprograms&dataareresidedinmainmemoryduringtheexecution.Theprogramsand data are not stored permanently due to following two reasons.

- Mainmemoryistoosmalltostoreallneededprogramsanddatapermanently.
- Mainmemoryisavolatilestoragedevicewhichlostitscontentswhenpoweristurnedoff.

MainMemory: ThemainmemoryandtheregistersaretheonlystorageareathattheCPUcan accessthedatadirectlywithoutanyhelpofotherdevice.Themachineinstructionwhichtake memoryaddressasargumentsdonottakediskaddress.Thereforeinexecutionanyinstructionsand any data must be resided in any one of direct access storage device. If the data are not in memory they must be moved before the CPU can operate on them. There are two types of main memorysuch as:

- **RAM (Random Access Memory):** The RAM is implemented in a semiconductor technologyiscalledD-RAM(DynamicRAM)whichformsanarrayofmemorywords/cells. Each&everywordshouldhaveitsownaddress/locator.Instructionis performedthrougha sequence of load and store instruction to specific memory address. Each I/Ocontroller includesregistertoholdcommandsofthedatabeingtransferred.Toallowmoreconvenient accesstoI/Odevice many computer architecture provide memory mappedI/O.Inthe case ofmemory mapped I/Oranges ofmemoryaddress are mapped tothe device register.Read and write to this memory addressbecause the data to be transferred to and from the device register.

SecondaryStorage: Themostcommonsecondarystoragedevicesaremagneticdiskandmagnetic tape which provide permanent storage of programs and data.

MagneticDisk: Itprovidesthebulkofsecondarystorageformoderncomputersystems.Eachdisk platter has flat circular shape like a CD. The diameter of a platter range starts from 1.8 to 5.25 inches. The two surfaces of a platter are covered with a magnetic material which records the information/dataisgivenbytheuser.Theread,writeheadareattachedtoadiskarm,whichmoves alltheheadsasaunit.Thesurfaceofaplatterislogicallydividedintocirculartrackswhicharesub dividedintosectors.Thesetoftrackswhichareatonearmpositionformsacylinder.Therearemay bethousandsofcylindersinadiskdrive&eachtrackcontains100ofsectors.Thestoragecapacity of a common disk drive is measured in GB. When the disk is in use a drive motor spins it at high speed. Most drives rotated 62 to 200 time/sec.Thedisk speed has two parts such as transfer rate & positioningtime.Thetransferrate istherateatwhichdataflow betweenthedrive&the computer. The positioning time otherwise called as random access time. It consists of two parts such as seek time & rotational latency. The seek time is the time taken to move the disk arc to the desired cylinder. The rotationallatency is the time takentorotate the disk head.

Magnetic Tape: It was used as early secondary storage medium. It is also permanent and can hold largequantityofdata.Itsaccesstimeisslower,comparisontomainmemorydevices.Magnetictapes are sequential innature.That'swhy randomaccess to magnetic tapeis thousand times slower than the random access to magnetic disk. The magnetic tapes are used mainly for backup the data. The magnetic tape must be kept in a non dusty environment and temperature controlled area. But the main advantage of the secondary storage device is that it can hold 2 to 3 times more data than a largediskdrive.Thereare4typesofmagnetictapessuchas:

- ½Inch

- ¼Inch
- 4 mm
- 8 mm

OperatingSystemServices

An operating system provides an environment for the execution of the program. It provides some servicestotheprograms.Thevariousservicesprovidedbyanoperatingsystemareasfollows:

- **Program Execution:** The system must be able to load a program into memory and to run that program. The program must be able to terminate this execution either normally or abnormally.
- **I/OOperation:** A running program mayrequire I/O.This I/Omayinvolve afile ora I/O device for specific device. Some special function can be desired. Therefore the operating system must provide a means to do I/O.
- **FileSystemManipulation:**Theprogramsneedtocreateanddeletefilesbynameandread andwritefiles.Thereforetheoperatingsystemmustmaintaineachandeveryfilescorrectly.
- **Communication:** The communication is implemented via shared memory or by the technique of message passing in which packets of information are moved between the processes by the operating system.
- **Error detection:** The operating system should take the appropriate actions for the occurrencesofanytypelikearithmeticoverflow,accesstotheillegalmemorylocationand too large user CPU time.
- **Research Allocation:** When multiple users are logged on to the system theresources must be allocated to each of them. For current distribution of the resource among the various processes the operating system uses the CPU scheduling run times which determine which process will be allocated with the resource.
- **Accounting:**Theoperatingsystemkeeptrackofwhichusersusehowmanyandwhichkind of computer resources.
- **Protection:** The operating system is responsible for both hardware as well as software protection. The operating system protects the information stored in a multiuser computer system.

ProcessManagement:

Process:A process or task is an instance of a program in execution. The execution of a process mustprogramsinasequentialmanner.Atanytimeatmostoneinstructionisexecuted.Theprocess includes the current activity as represented by the value of the program counter and the content of the processors registers. Also it includes the process stack which contain temporary data (such as method parameters return address and local variables) & a data section which contain global variables.

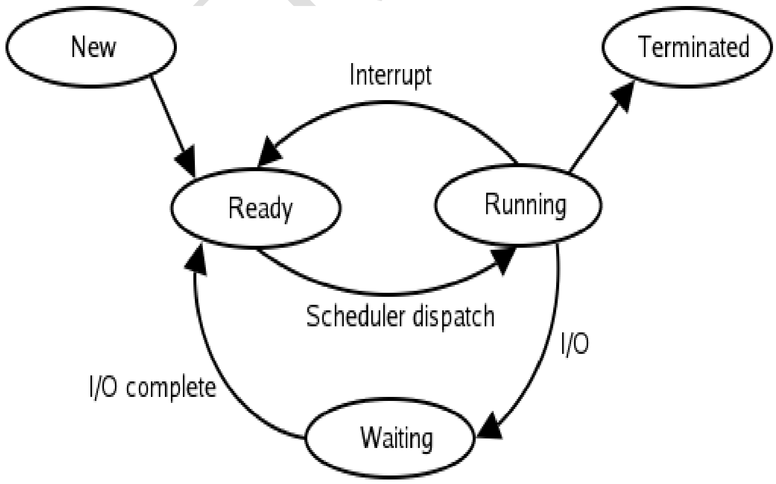
Differencebetweenprocess&program:

Aprogramby itselfisnotaprocess.A programinexecutionisknownas aprocess.Aprogramisa passiveentity,suchasthecontentsofafilestoredondiskwhereasprocessisanactiveentitywitha programcounterspecifyingthenextinstructiontoexecuteandasetofassociatedresourcesmaybe sharedamong several process with some scheduling algorithmbeing used to determinate when the stop work on one process and service a different one.

Process state: As a process executes, it changes state. The state of a process is defined by the correctactivity ofthat process. Eachprocess maybeinone ofthe following states.

- **New:**Theprocessisbeingcreated.
- **Ready:**Theprocessiswaitingtobeassignedtoaprocessor.
- **Running:**Instructionsarebeingexecuted.
- **Waiting:**Theprocessiswaitingforsomeeventtooccur.
- **Terminated:**Theprocesshasfinishedexecution.

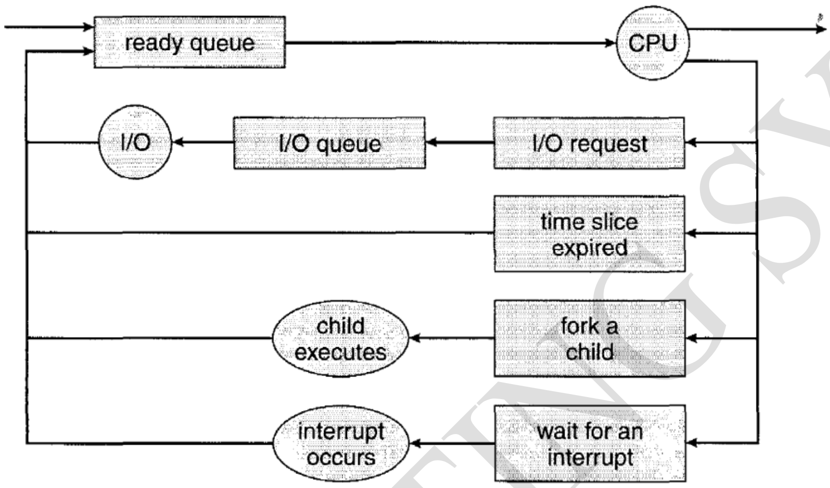
Many processes may be in ready and waiting state at the same time. But only one process can be running on any processor at any instant.



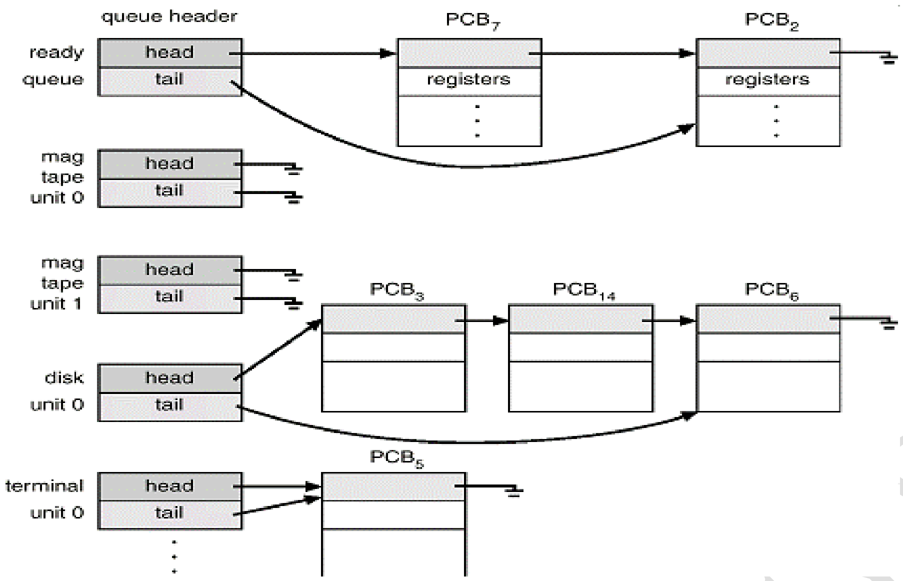
Processscheduling:

SchedulingisafundamentalfunctionofOS.Whenacomputerismultiprogrammed,ithasmultiple processescompletingfortheCPUatthesametime.IfonlyoneCPUisavailable,thenachoicehas to be made regarding which process to execute next. This decision making process is known as schedulingandthepartoftheOSthatmakesthischoiceiscalledascheduler.Thealgorithmituses inmakingthischoiceiscalledschedulingalgorithm.

Schedulingqueues: Asprocessesenterthesystem,theyareputintoajobqueue.Thisqueue consistsofallprocessinthesystem.Theprocessthatareresidinginmainmemoryandareready& waiting toexecute or keptona list calledready queue.



Thisqueueisgenerallystoredasalinkedlist.Areadyqueueheadercontainspointerstothefirst& final PCB in the list. The PCB includes a pointer field that points to the next PCB in the ready queue. The lists of processes waiting for a particular I/Odevice are kept on a list called device queue. Each device has its own device queue. A new process is initially put in the ready queue. It waitsinthereadyqueueuntilitisselectedforexecution&isgiventheCPU.



SCHEDULERS:

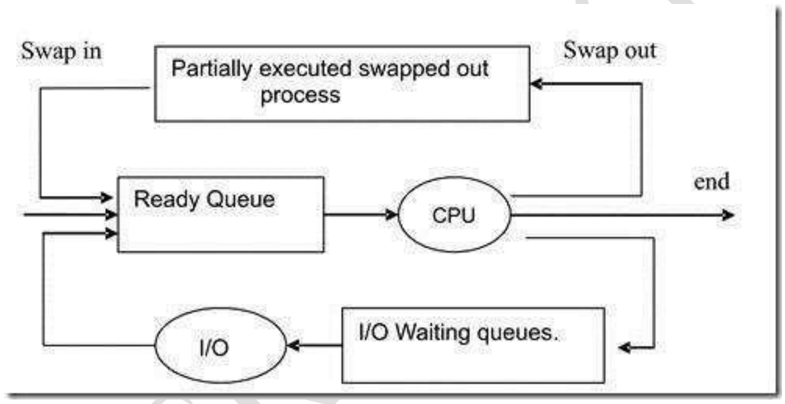
A process migrates between the various scheduling queues throughout its life-time purposes. The OSmustselectforschedulingprocessesfromthesequeuesinsomefashion.This selectionprocess iscarriedoutbytheappropriatescheduler.Inabatchsystem,moreprocessesaresubmittedandthen executedimmediately.Sotheseprocessesarespooledtoamassstoragedevicelikedisk,wherethey are kept for later execution.

Typesofschedulers:

Thereare3typesofschedulersmainlyused:

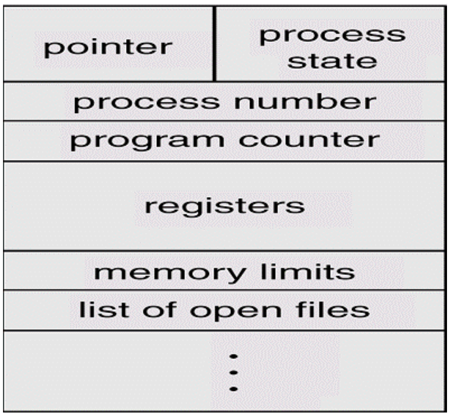
1. **Longtermscheduler:**Longtermschedulerselectsprocessfromthedisk&loadsthemintomemory for execution. It controls the degreeof multi-programming i.e. no. of processes in memory. It executes less frequently than other schedulers. If the degree of multiprogrammingisstablethantheaveragerateofprocesscreationisequaltotheaverage departure rate of processes leaving the system. So, the long term scheduler is needed to be invokedonlywhenaprocessleavesthesystem.Duetolongerintervalsbetweenexecutions it can afford to take more time to decide which process should be selected for execution. MostprocessesintheCPUareeither I/OboundorCPUbound.An I/Oboundprocess (an interactive 'C' program is one that spends most of its time in I/Ooperation than it spends in doing I/Ooperation. A CPU bound process is one thatspends more ofits time in doing computations thanI/Ooperations (complex sorting program). Itis important thatthe long term scheduler should select a good mix of I/O bound & CPU bound processes.

2. **Short-termscheduler:**Theshorttermschedulerselectsamongtheprocessthatareready to execute & allocates the CPU to one of them. The primary distinction between these two schedulers is the frequency of their execution. The short-term scheduler must select a new process for the CPU quite frequently. It must execute at least one in 100ms. Due to the short duration of time between executions, it must be very fast.
3. **Medium - term scheduler:** some operating systems introduce an additional intermediate levelofschedulingknownasmedium-termscheduler.Themainideabehindthisscheduler is that sometimes it is advantageous to remove processes from memory & thus reduce the degree of multiprogramming. At some later time, the process can be reintroduced into memory & its execution can be continued from where it had left off. This is called as swapping. The process is swapped out & swapped in later by medium term scheduler. Swapping is necessary to improve theprocess miss or due to some change in memory requirements, the available memory limit is exceeded which requires some memory to be freed up.



Processcontrolblock:

Eachprocessisrepresentedinthe OSbyaprocesscontrol block. Itisalsobyaprocesscontrol block. It is also known as task control block.



Aprocesscontrolblockcontainsmanypiecesofinformationassociatedwithaspecificprocess. It includes the following informations.

- **Processstate:**Thestatemaybenew,ready,running,waitingorterminatedstate.
- **Programcounter:**itindicatestheaddress ofthenextinstructiontobeexecutedforthis purpose.
- **CPU registers:** The registers vary in number & type depending on the computer architecture. It includes accumulators, index registers, stack pointer & general purpose registers, plus any condition- code information must be savedwhen an interrupt occurs to allow the process to be continued correctly after- ward.
- **CPU scheduling information:**This information includes process priority pointers to scheduling queues & anyother scheduling parameters.
- **Memory management information:** This information may include such information asthevalueofthebar&limitregisters,thepagetablesorthesegmenttables,depending upon the memory system used by the operating system.
- **Accounting information:** This information includes the amount of CPU and real time used, time limits, account number, job or process numbers and so on.
- **I/OStatus Information:** This information includes the list of I/Odevices allocated to this process, a list of open files and so on. The PCB simply serves as the repository for any information that may vary from process to process.

CPUSchedulingAlgorithm:

CPUSchedulingdealswiththeproblemofdecidingwhichoftheprocessesinthereadyqueueisto be allocated firstto the CPU.There are four types of CPUscheduling that exist.

1. **FirstCome,FirstServedScheduling(FCFS)Algorithm:**ThisisthesimplestCPUscheduling algorithm.Inthisscheme,theprocesswhichrequeststheCPUfirst,thatisallocatedtotheCPU first.TheimplementationoftheFCFSalgorithmiseasilymanagedwithaFIFOqueue.Whena processentersthereadyqueueitsPCBislinkedontotherearofthequeue.Theaveragewaiting timeunderFCFSpolicyisquietlong.Considerthefollowingexample:

Process	**CPUtime**
P_1	3
P_2	5
P_3	2
P_4	4

UsingFCFSalgorithmfindtheaveragewaitingtimeandaverageturnaroundtimeiftheorderis P_1, P_2, P_3, P_4.

Solution: If the process arrivedinthe orderP_1, P_2, P_3, P_4 thenaccording tothe FCFSthe Gantt chart will be:

P_1	P_2	P_3	P_4

0 3 8 10 14

ThewaitingtimeforprocessP$_1$=0,P$_2$=3,P$_3$=8,P$_4$=10thentheturnaroundtimefor process $P_1 = 0 + 3 = 3$, $P_2 = 3 + 5 = 8$, $P_3 = 8 + 2 = 10$, $P_4 = 10 + 4 = 14$.

Thenaveragewaitingtime=(0+3+8+10)/4=21/4=5.25

Averageturnaroundtime=(3+8+10+14)/4=35/4=8.75

The FCFS algorithm is non preemptive means once the CPU has been allocated to a process then the process keeps the CPU until the release the CPU either by terminating or requesting I/O.

2. **ShortestJobFirstScheduling(SJF)Algorithm:**Thisalgorithmassociateswitheachprocess if the CPU is available. This scheduling is also known as shortest next CPU burst, because the scheduling is done by examining the length ofthe nextCPUburstofthe process rather thanits total length. Consider the following example:

Process	**CPUtime**
P_1	3
P_2	5
P_3	2
P_4	4

Solution: AccordingtotheSJFtheGanttchartwillbe

P_3	P_1	P_2	P_4

0 2 5 9 14

The waiting time forprocess P_1 = 0,P_2 = 2,P_3 = 5,P_4 = 9 then the turnaround time for process P_3 =0+ 2 =2, P_1= 2+ 3= 5,P_4= 5+ 4=9, P_2 =9+5=14.

Thenaverage waiting time= (0 + 2 +5+ 9)/4 =16/4= 4

Averageturnaroundtime=(2+5+9+14)/4=30/4=7.5

TheSJFalgorithmmaybeeitherpreemptiveornonpreemptivealgorithm.ThepreemptiveSJF is also known as shortest remaining time first.

Considerthefollowingexample.

Process	**ArrivalTime**	**CPUtime**
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

InthiscasetheGanttchartwillbe

P_1	P_2	P_4	P_1	P_3

0 1 5 10 17 26

Thewaitingtimeforprocess

P_1 = 10 - 1 = 9

P_2=1–1=0

P_3=17–2=15

P_4=5–3=2

Theaveragewaitingtime=(9+0+15+2)/4=26/4=6.5

3. **Priority Scheduling Algorithm:** In this scheduling a priority is associated with each process and the CPU is allocated to the process with the highest priority. Equal priority processes are scheduledinFCFS manner. Consider the followingexample:

Process	**ArrivalTime**	**CPUtime**
P_1	10	3
P_2	1	1
P_3	2	3

P_4	1	4
P_5	5	2

According tothepriority schedulingtheGanttchartwillbe

P_2	P_5	P_1	P_3	P_4

0 1 6 16 18 19

Thewaitingtimeforprocess

P_1 = 6

P_2=0

P_3=16

P_4=18

P_4=1

Theaveragewaitingtime=(0+1+6+16+18)/5=41/5=8.2

4. **Round Robin Scheduling Algorithm:** This type of algorithm is designed only for the time sharing system. It is similar to FCFS scheduling with preemption condition to switch between processes.Asmallunitoftime calledquantum time or timeslice is usedtoswitchbetween the processes. The average waiting time under the round robin policy is quiet long. Consider the following example:

Process	**CPUtime**
P_1	3
P_2	5
P_3	2
P_4	4

TimeSlice=1millisecond.

P_1	P_2	P_3	P_4	P_1	P_2	P_3	P_4	P_1	P_2	P_4	P_2	P_4	P_2

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

Thewaitingtimeforprocess

P_1=0+(4–1)+(8–5)=0+3+3=6

P_2=1+(5–2)+(9– 6)+(11–10)+(12–11)+(13–12)=1+3+3+1+1+1=10

P_3=2+(6–3)=2+3=5

P_4=3+(7–4)+(10–8)+(12–11)=3+3+2+1=9

Theaveragewaitingtime=(6+10+5 +9)/4=7.5

ProcessSynchronization:

A co-operation process isone that can affectorbe affected by other processesexecuting in the system.Co-operatingprocessmayeitherdirectlysharealogicaladdressspaceorbeallottedtothe shareddataonlythroughfiles.ThisconcurrentaccessisknownasProcesssynchronization.

CriticalSectionProblem:

Considerasystemconsistingofnprocesses(P_0,P_1,………P_{n-1})eachprocesshasasegmentofcode whichisknownascriticalsectioninwhichtheprocessmaybechangingcommonvariable,updating atable,writingafile andsoon.Theimportantfeature ofthesystemisthatwhentheprocess is executinginitscriticalsectionnootherprocessistobeallowedtoexecuteinitscriticalsection.The executionofcriticalsectionsbytheprocessesisamutuallyexclusive.Thecriticalsectionproblemis to design a protocol that the process can use to cooperate each process mustrequest permission to enteritscriticalsection.Thesectionofcodeimplementingthis requestistheentrysection.The criticalsectionisfollowedonexitsection.Theremainingcodeistheremaindersection.

Example:

```
While(1)
{
Entry Section;
        CriticalSection;
Exit Section;
        RemainderSection;
}
```

Asolutiontothecriticalsectionproblemmustsatisfythefollowingthreeconditions.

1. **MutualExclusion:**IfprocessP$_i$isexecutinginitscriticalsectionthennoanyotherprocess can be executing in their critical section.
2. **Progress:**Ifnoprocessisexecutinginitscriticalsectionandsomeprocesswishtoenter their critical sections thenonly those process thatare not executingintheirremainder section can enter its critical section next.
3. **Boundedwaiting:**Thereexistsaboundonthe numberoftimesthatotherprocessesare allowedtoentertheircriticalsectionsafteraprocesshasmadearequest.

Semaphores:

Forthesolutiontothecriticalsectionproblemonesynchronizationtoolisusedwhichisknownas semaphores.Asemaphore‘S’isanintegervariablewhichisaccessedthroughtwostandard operationssuchaswaitandsignal.Theseoperationswereoriginallytermed‘P’(forwaitmeansto test)and‘V’(forsinglemeanstoincrement).Theclassicaldefinitionofwaitis

```
Wait(S)
{
    While(S<=0)
    {
        Test;
    }
    S--;
}
```

Theclassicaldefinitionofthesignalis Signal

```
(S)
{
    S++;
}
```

Incaseofwaitthetestconditionisexecutedwithinterruptionandthedecrementisexecuted without interruption.

BinarySemaphore:

Abinarysemaphoreisasemaphorewithanintegervaluewhichcanrangebetween0and1.

Let‘S’beacountingsemaphore.Toimplementthebinarysemaphoreweneedfollowingthe structure of data.

BinarySemaphoresS_1,S_2;

int C;

InitiallyS_1=1,S2=0andthevalueofCissettotheinitialvalueofthecountingsemaphore‘S’. Then the wait operation of the binary semaphore can be implemented as follows.

```
Wait(S1)
C--;
if(C <0)
{
    Signal(S1);
    Wait(S2);
}
```

Signal(S_1);

Thesignaloperationofthebinarysemaphorecanbeimplementedasfollows: Wait

(S_1);

C++;

if(C<=0)

Signal(S_2);

Else

Signal(S_1);

ClassicalProblemonSynchronization:

Therearevarioustypesofproblemwhicharproposedforsynchronizationschemesuchas

- **Bounded Buffer Problem:** This problem was commonly used to illustrate the power of synchronizationprimitives.Inthisscheme weassumedthatthepoolconsistsof'N'buffer andeachcapableofholdingoneitem.The'mutex'semaphoreprovidesmutualexclusionfor accesstothebufferpoolandisinitializedtothevalueone.Theemptyandfullsemaphores countthenumberofemptyandfullbufferrespectively.Thesemaphoreemptyisinitialized to'N'andthesemaphorefullisinitializedtozero.Thisproblemisknownasprocedureand consumer problem. The code of the producer is producing full buffer and the code of consumeris producing empty buffer. The structure of producer process is as follows:

  do{
  produceaniteminnextp
  ....
  Wait(empty);
  Wait(mutex);
  ....
  addnextptobuffer
  ....
  Signal(mutex);
  Signal (full);
  }While(1);

  Thestructureofconsumerprocessisasfollows:

```
do{
Wait (full);
Wait(mutex);
...........
Removeanitemfrombuffertonextc
...........
Signal(mutex);
Signal(empty);
............
Consumetheiteminnextc;
..............
}While(1);
```

- **ReaderWriterProblem:**Inthistypeofproblemtherearetwotypesofprocessareused suchasReaderprocessandWriterprocess.Thereaderprocessisresponsibleforonly readingandthewriterprocessisresponsibleforwriting.Thisisanimportantproblemof synchronizationwhichhasseveralvariations like
  - The simplestone is referred as first reader writerproblemwhich requires thatno readerwillbekeptwaitingunlessawriterhasobtainedpermissiontousetheshared object. In other words no reader should wait for other reader to finish because a writer is waiting.
  - Thesecondreaderwriterproblemrequiresthatonceawriterisreadythenthewriter performs its write operation as soon as possible.

Thestructureofareaderprocessisasfollows: Wait

```
(mutex);
Readcount++;
if(readcount==1)
Wait (wrt);
Signal(mutex);
...........
Readingisperformed
...........
Wait(mutex);
```

Readcount--;

if(readcount==0) Signal

(wrt);

Signal(mutex);

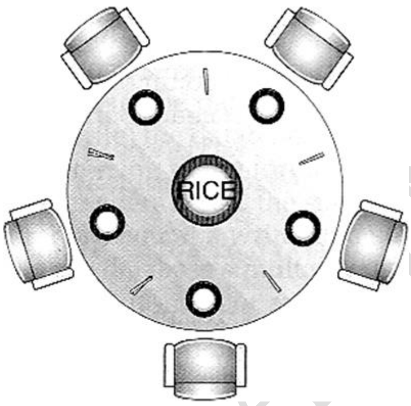
Thestructureofthewriterprocessisasfollows:

Wait(wrt);

Writingisperformed;

Signal (wrt);

- **DiningPhilosopherProblem:**Consider5philosopherstospendtheirlivesinthinking& eating.Aphilosophersharescommoncirculartablesurroundedby5chairseachoccupies by one philosopher. In the center of the table there is a bowl of riceand the table is laid with 6 chopsticks as shown in below figure.



When a philosopherthinks she does not interact with hercolleagues.From time to time a philosopher gets hungry and tries to pickup two chopsticks that are closest to her. A philosophermaypickuponechopstickortwochopsticksatatimebutshecannotpickupa chopstickthatisalreadyinhandoftheneighbor.Whenahungryphilosopherhasbothher chopsticksatthesametime,sheeatswithoutreleasingherchopsticks.Whenshefinished eating, she puts down both of her chopsticksand starts thinking again. This problem is consideredasclassicsynchronizationproblem.Accordingtothis problemeachchopstickis representedbyasemaphore.Aphilosophergrabsthechopsticks byexecutingthewait operationonthatsemaphore.Shereleasesthechopsticksbyexecutingthesignaloperation on the appropriate semaphore. The structure of dining philosopher is as follows:

do{

```
Wait(chopstick[i]);
Wait(chopstick[(i+1)%5]);
.............
Eat
.............
Signal(chopstick[i]);
Signal(chopstick[(i+1)%5]);
.............
Think
.............
}While(1);
```

CriticalRegion:

Accordingtothecriticalsectionproblemusingsemaphoreallprocessesmustshareasemaphore variablemutexwhichisinitializedtoone.Eachprocessmustexecutewait(mutex)beforeentering thecriticalsectionandexecutethesignal(mutex)aftercompletingtheexecutionbutthereare variousdifficultiesmayarisewiththisapproachlike:

Case1: Supposethataprocessinterchangestheorderinwhichthewaitandsignaloperationson thesemaphoremutexareexecuted,resultinginthefollowingexecution:

```
Signal(mutex);
..........
CriticalSection
...........
Wait(mutex);
```

Inthissituationseveralprocessesmaybeexecutingintheircriticalsectionssimultaneously,whichis violating mutual exclusion requirement.

Case2: Supposethataprocessreplacesthesignal(mutex)withwait(mutex).Theexecutionisas follows:

```
Wait(mutex);
...........
CriticalSection
...........
Wait(mutex);
```

Inthissituationadeadlockwilloccur

Case3: Supposethataprocessomitsthewait(mutex)andthesignal(mutex).Inthiscasethe mutualexclusionisviolatedoradeadlockwilloccur.

Toillustratethevarioustypesorerrorgeneratedbyusingsemaphorethereare somehighlevel languageconstructshavebeenintroducedsuchascriticalregionandmonitor.

Criticalregionisalsoknownasconditionalcriticalregions.Itconstructsguardsagainstcertain simpleerrorsassociatedwithsemaphore.Thishighlevellanguagesynchronizationconstruct requiresavariableVoftypeTwhichistobesharedamongmanyprocesses.Itisdeclaredas V: shared T;

ThevariableVcanbeaccessedonlyinsidearegionstatementaslikebelow: Wait (mutex);

```
While (! B) {
First_count++;
if (second_count>0)
        Signal(second_delay);
Else
        Signal(mutex);
Wait(first_delay);
First_count--;
Second_count++;
if(first_count> 0)
        Signal(first_delay);
Else
        Signal(second_delay);
Wait(second_delay);
Second_count --;
}
S;
if(first_count> 0)
        Signal (first_delay);
Else if (second_count> 0)
        Signal(second_delay);
```

Else

Signal(mutex);

(Implementationoftheconditionalregionconstructs)

WhereBisaBooleanvariablewhichgovernstheaccesstothecriticalregionswhichisinitializedto false.Mutex,First_delayandSecond_delayarethesemaphoreswhichareinitializedto1,0,and0 respectively.First_countandSecond_countaretheintegervariableswhichareinitializedtozero.

Monitor:

Itischaracterizedasasetofprogrammerdefinedoperators.Itsrepresentationconsistsofdeclaring ofvariables,whosevaluedefinesthestateofaninstance.Thesyntaxofmonitorisasfollows.

```
Monitormonitor_name
{
        Shared variable declarations
        ProcedurebodyP1(………){
        ........
        }
        ProcedurebodyP2(………) {
        ........
        }
                .
                .
                .
        ProcedurebodyPn(………){
        ........
        }
        {
                InitializationCode
        }
}
```

AtomicTransaction:

Thissectionisrelatedtothefieldofdatabasesystem.Atomictransactiondescribesthevarious techniquesofdatabaseandhowtheyarecanbeusedbytheoperatingsystem.Itensuresthatthe criticalsectionsareexecutedautomatically.Todeterminehowthesystemshouldensureatomicity weneedfirsttoidentifythepropertiesofthedevicesusedtoforstoringthedataaccessedbythe transactions.Thevarioustypesstoringdevicesareasfollows:

- **VolatileStorage:**Informationresidinginvolatilestoragedoesnotsurviveincaseof systemcrash.Exampleofvolatilestorageismainmemoryandcachememory.
- **NonvolatileStorage:**Informationresidinginthistypeofstorageusuallysurvivesincase ofsystemcrash.ExamplesareMagneticDisk,MagneticTapeandHardDisk.
- **StableStorage:**Informationresidinginstablestorageisneverlost.Exampleisnonvolatile cache memory.

Thevarioustechniquesusedforensuringtheatomicityareasfollows:

1. **LogbasedRecovery:**Thistechniqueisusedforachievingtheatomicitybyusingdatastructure calledlog.Aloghasthefollowingfields:
    a. **TransactionName:**Thisistheuniquenameofthetransactionthatperformedthe write operation.
    b. **DataItemName:**Thisistheuniquenamegiventothedata.
    c. **OldValue:**Thisisthevalueofthedatabeforetothewriteoperation.
    d. **Newvalue:**Thisisthevalueofthedataafterthewriteoperation.

    ThisrecoverytechniqueusestwoprocessessuchasUndoandRedo.Undorestoresthevalueof olddataupdatedbyatransactiontotheoldvalues.Redosetsthevalueofthedataupdatedbya transaction to the new values.
2. **Checkpoint:**Inthisprinciplesystemmaintainsthelog.Thecheckpointrequiresthefollowing sequences of action.
    a. Outputallthelogrecordsfromvolatilestorageintostablestorage.
    b. Outputallmodifieddataresidinginvolatiletothestablestorage.
    c. Outputacheckpointontothestablestorage.

T_0	T_1
Read(A)	
Write(A)	
Read(B)	

T0	T1
Write(B)	
	Read(A)
	Write(A)
	Read(B)
	Write(B)

3. **Serializibility:** In this technique the transaction executedseriallyinsomearbitraryorder.Considera system consisting two data items A and B which are both read and writtenbytwotransactionsT_0andT_1.Supposethat their transactionsareexecutedautomaticallyintheorder T_0followedbyT_1.Thisexecutionsequenceisknownasschedulewhichisrepresentedasbelow.

Iftransactionsareoverlappedthentheirexecutionresultingscheduleisknownasnon-serial scheduling or concurrent schedule as like below:

T_0	T_1
Read(A)	
Write(A)	
	Read(A)
	Write(A)
Read(B)	
Write (B)	
	Read(B)
	Write(B)

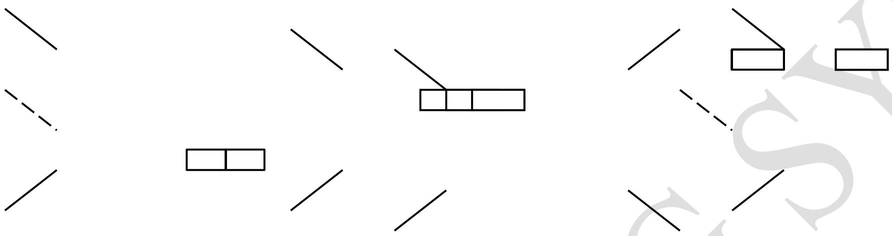
4. **Locking:**Thistechniquegovernshowthelocksareacquiredandreleased.Therearetwotypes oflocksuchassharedlockandexclusivelock.IfatransactionThasobtainedasharedlock(S) on data item Q then T can read this item but cannot write. If a transaction T has obtained an exclusive lock (S) on dataitem Q then T can both read and write in the data item Q.
5. **Timestamp:**Inthistechnique eachtransactioninthe systemisassociatedwithuniquefixed timestampdenotedbyTS.Thistimestampisassignedbythesystembeforethetransaction starts.IfatransactionT$_i$hasbeenassignedwithatimestampTS(T$_i$)andlateranewtransaction T$_j$enters the systemthenTS (T_i) <TS (T_j). Thereare twotypes oftimestampsuchas W- timestampandR-timestamp.W-timestampdenotesthelargesttimestampofanytransactionthat performedwriteoperationsuccessfully.R-timestampdenotesthelargesttimestampofany transaction that executed read operation successfully.

Deadlock:

In a multiprogramming environmentseveral processes maycompete for a finite numberof resources.Aprocessrequestresources;iftheresourceisavailableatthattimeaprocessentersthe waitstate.Waitingprocessmayneverchangeitsstatebecausetheresourcesrequestedareheldby other waiting process. This situation is knownas deadlock.

Example

- Systemhas2diskdrives.
- P1andP2eachholdonediskdriveandeachneedsanotherone.
- 2trainapproacheseachotheratcrossing,bothwillcometofullstopandneithershallstart until other has gone.



- Trafficonlyinonedirection.
- Eachsectionofabridgecanbeviewedasaresource.
- Ifadeadlockoccurs,itcanberesolvedifonecarbacksup(preemptresourcesandrollback).
- Severalcarsmayhavetobebackedupifadeadlockoccurs.
- Starvationispossible

SystemModel:

Asystemconsistsofafinitenumberofresourcestobedistributedamonganumberofcompeting processes.Theresourcesarepartitionedintoseveraltypeseachofwhichconsistsofanumberof identicalinstances.Aprocessmayutilizedaresourcesinthefollowingsequence

- **Request:**Inthisstateonecanrequestaresource.
- **Use:**Inthisstate the processoperates onthe resource.
- **Release:**Inthisstatetheprocessreleasestheresources.

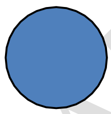
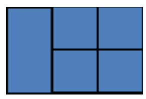
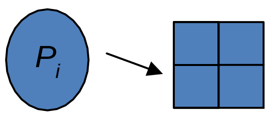
DeadlockCharacteristics: Inadeadlockprocessneverfinishexecutingandsystemresourcesare tiedup.Adeadlocksituationcanariseifthefollowingfourconditionsholdsimultaneouslyina system.

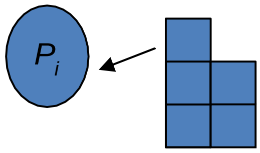
- **MutualExclusion:**Atatimeonlyoneprocesscanusetheresources.Ifanotherprocess requeststhatresource,requestingprocessmustwaituntiltheresourcehasbeenreleased.

- **Holdandwait:**Aprocessmustbeholdingatleastoneresourceandwaitingtoadditional resource that is currently held by other processes.
- **NoPreemption:**Resourcesallocatedtoaprocesscan'tbeforciblytakenoutfromitunless it releases that resource after completing the task.
- **CircularWait:**Aset{P_0,P_1,…….P_n}ofwaitingstate/processmustexistssuchthat$P_0$is waitingforaresourcethatisheldbyP_1,P_1iswaitingfortheresourcethatisheldbyP_2….. $P_{(n-1)}$iswaitingfortheresourcethatisheldby$P_n$and$P_n$iswaitingfortheresourcesthatis held by P_4.

Resource AllocationGraph:

Deadlockcanbedescribedmoreclearlybydirectedgraphwhichiscalledsystemresourceallocation graph.Thegraphconsistsofasetofvertices'V'andasetofedges'E'.Thesetofvertices'V'is partitioned into two different types of nodes such as P = {P_1, P_2, …….P_n}, the set of all theactive processesinthesystemandR={R_1,R_2,…….R_m},thesetofalltheresourcetypeinthesystem.A directededgefromprocess$P_i$toresourcetype$R_j$isdenotedby$P_i \rightarrow R_j$.Itsignifiesthatprocess$P_i$is aninstanceofresourcetype$R_j$andwaitsforthatresource.Adirectededgefromresourcetype$R_j$to theprocess$P_i$whichsignifiesthataninstanceofresourcetype$R_j$hasbeenallocatedtoprocessP_i.A directededge$P_i \rightarrow R_j$iscalledasrequestedgeand$R_j \rightarrow P_i$iscalledasassignededge.

- Process 
- ResourceTypewith4instances 
- $P_i$requestsinstance ofR_j 

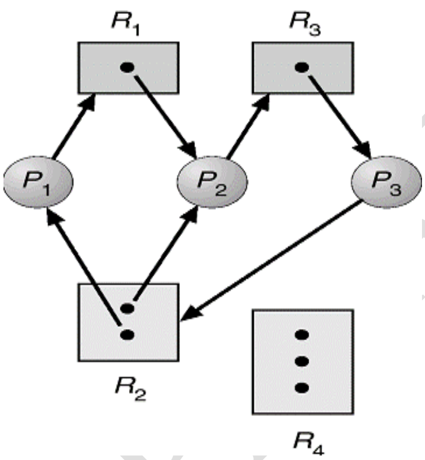
- $P_i$isholdinganinstanceofR_j 

WhenaprocessP$_i$requestsaninstanceofresourcetypeR$_j$thenarequestedgeisinsertedas resourceallocationgraph.Whenthisrequestcanbefulfilled,therequestedgeistransformedtoan assignmentedge.Whentheprocessnolongerneedsaccesstotheresourceitreleasestheresource andasaresulttheassignmentedgeisdeleted.Theresourceallocationgraphshowninbelowfigure has the following situation.

- ThesetsP,R,E
- $P=\{P_1,P_2,P_3\}$
- $R=\{R_1,R_2,R_3,R_4\}$
- $E=\{P_1\rightarrow R_1,P_2\rightarrow R_3,R_1\rightarrow P_2,R_2\rightarrow P_2,R_2\rightarrow P_1,R_3\rightarrow P_3\}$

Theresourceinstancesare

- ResourceR$_1$hasoneinstance
- ResourceR$_2$hastwoinstances.
- ResourceR$_3$hasoneinstance
- ResourceR$_4$hasthreeinstances.

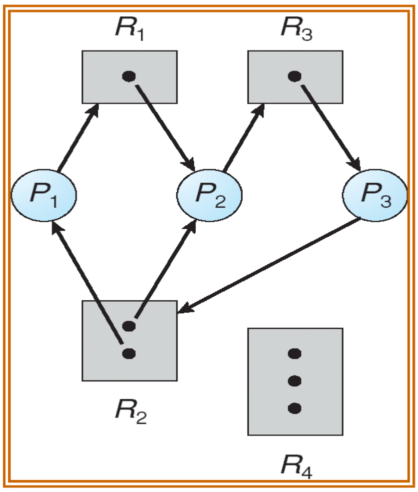


Theprocessstatesare:

- ProcessP$_1$isholdinganinstanceofR$_2$andwaitingforaninstanceofR$_1$.
- ProcessP$_2$isholdinganinstanceofR$_1$andR$_2$andwaitingforaninstanceR$_3$.
- ProcessP$_3$isholdinganinstanceofR$_3$.

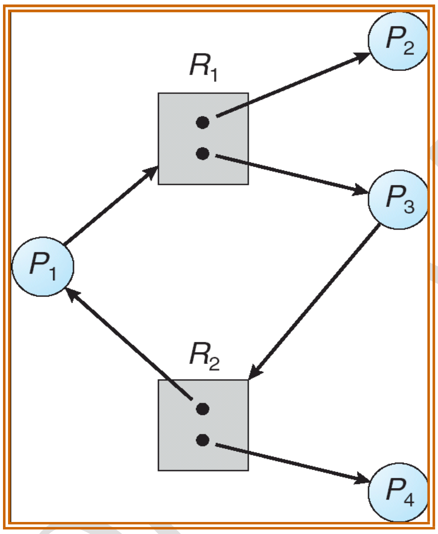
Thefollowingexampleshowstheresourceallocationgraphwithadeadlock.

- P1->R1->P2->R3->P3->R2->P1
- P2->R3->P3->R2->P1



Thefollowingexampleshowstheresourceallocationgraphwithacyclebutnodeadlock.

- P1->R1->P3->R2->P1
- Nodeadlock
- P4mayreleaseitsinstanceofresourceR2
- ThenitcanbeallocatedtoP3



MethodsforHandlingDeadlocks

Theproblemofdeadlockcandealwiththefollowing3ways.

- Wecanuseaprotocoltopreventoravoiddeadlockensuringthatthesystemwillneverenter to a deadlock state.
- Wecanallowthesystemtoenteradeadlockstate,detectitandrecover.
- Wecanignoretheproblemalltogether.

Toensurethatdeadlockneveroccurthesystemcanuseeitheradeadlockpreventionordeadlock avoidance scheme.

DeadlockPrevention:

Deadlockpreventionisasetofmethodsforensuringthatatleastoneofthesenecessaryconditions cannot hold.

- **MutualExclusion:**Themutualexclusionconditionholdsfornonsharable.Theexampleisa printercannotbesimultaneouslysharedbyseveralprocesses.Sharableresourcesdonot requiremutualexclusiveaccessandthuscannotbeinvolvedinadeadlock.Theexampleis readonlyfileswhichareinsharing condition.Ifseveralprocesses attempttoopenthe read onlyfileatthesametimetheycanbeguaranteedsimultaneousaccess.
- **Hold and wait:**To ensure that the hold and wait condition never occurs in the system, we must guaranty that whenevera process requests a resource it does not hold any other resources. Thereare twoprotocols to handle theseproblems suchas one protocol that can be usedrequireseachprocesstorequestandbeallocatedallitsresourcesbeforeitbegins execution.Theotherprotocolallowsaprocesstorequestresourcesonlywhentheprocesshas noresource.Theseprotocolshavetwomaindisadvantages.First,resourceutilizationmaybe low,sincemanyoftheresourcesmaybeallocatedbutunusedforalongperiod.Second, starvationispossible.Aprocessthatneedsseveralpopularresourcesmayhavetowait indefinitely,becauseatleastoneoftheresourcesthatitneedsisalwaysallocatedtosomeother process.
- **NoPreemption:**Toensurethatthisconditiondoesnothold,aprotocolisused.Ifaprocess isholdingsomeresourcesandrequestanotherresourcethatcannotbeimmediatelyallocated to it. The preempted oneadded to a list of resources forwhichthe process iswaiting. The processwillrestartonlywhenitcanregainitsoldresources,aswellasthenewonesthatitis requesting.Alternativelyifaprocessrequestssomeresources,wefirstcheckwhethertheyare available.Iftheyare,weallocatethem.Iftheyarenotavailable,wecheckwhethertheyare allocatedtosomeotherprocessthatiswaitingforadditionalresources.Ifso,wepreemptthe desiredresourcesfromthewaitingprocessandallocatethemtotherequestingprocess.Ifthe resourcesarenoteitheravailableorheldbyawaitingprocess,therequestingprocessmust wait.
- **CircularWait:**Wecanensurethatthisconditionneverholdsbyorderingofallresourcetype andtorequirethateachprocessrequestsresourceinanincreasingorderofenumeration.LetR

= {R_1,R_2,…….R_n}bethesetofresourcetypes.Weassigntoeachresourcetypea unique integer number, which allows us to compare two resources and to determine whether one precedesanotherinourordering.Formally,wedefineaonetoonefunction$F:R\rightarrow N$,where N is the set of natural numbers. Forexample,iftheset of resource types R includes tape drives, disk drivesandprinters,thenthe functionFmight be definedas follows:

F(TapeDrive)=1,

F(DiskDrive)=5, F

(Printer) = 12.

Wecannowconsiderthefollowingprotocoltopreventdeadlocks:Eachprocesscanrequest resourcesonlyinanincreasingorderofenumeration.Thatis,aprocesscaninitiallyrequest anynumberofinstancesofaresourcetype,say R_i.Afterthat,theprocesscanrequest instancesofresourcetype$R_j$ifandonlyif$F(R_j)>F(R_i)$.Ifseveralinstancesofthesame resourcetypeareneeded,definedpreviously,aprocessthatwantstousethetapedriveand printer at the same time must first request the tape drive and then request the printer.

DeadlockAvoidance

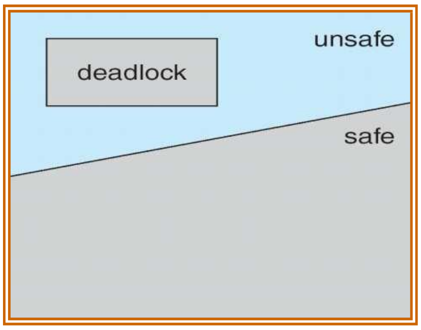
Requiresadditionalinformationabouthowresourcesaretobeused.Simplestandmostusefulmodel requires thateachprocess declare the maximumnumberofresources of eachtype that it may need.Thedeadlock-avoidancealgorithmdynamicallyexaminestheresource-allocationstatetoensure thattherecanneverbeacircular-waitcondition.Resource-allocationstateisdefinedbythenumber ofavailable andallocatedresources,andthemaximumdemandsoftheprocesses.

SafeState

Whenaprocessrequestsanavailableresource,systemmustdecideifimmediateallocationleavesthe systeminasafestate.Systemsareinsafestateifthereexistsasafesequenceofallprocess.A sequence <P_1,P_2, …,P_n> of ALL theprocessesisthe system such thatforeach P_i, the resources that$P_i$canstillrequestcanbesatisfiedbycurrentlyavailableresources+resourcesheldbyalltheP$_j$, withj <i.That is:

- If$P_i$resourceneedsarenotimmediatelyavailable,then$P_i$canwaituntilall$P_j$havefinished.
- When$P_j$isfinished,$P_i$canobtainneededresources,execute,returnallocatedresources,and terminate.
- When$P_i$terminates,P_{i+1}canobtainitsneededresources,andsoon.
- Ifsystemisinsafestate=>Nodeadlock

- Ifsysteminnotinsafestate=>possibilityofdeadlock
- OScannotpreventprocessesfromrequestingresourcesinasequencethatleadstodeadlock
- Avoidance=>ensuethatsystemwillneverenteranunsafestate,preventgettingintodeadlock



Example:

	Maximum Needs	Current Needs
P_0	10	5
P_1	4	2
P_2	9	2

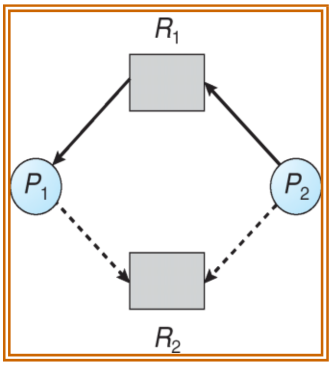
- SupposeprocessesP0,P1,andP2share12magnetictapedrives
- Currently9drivesareheldamongtheprocessesand3areavailable
- Question:Isthissystemcurrentlyinasafestate?
- Answer: Yes!
    - SafeSequence:<P1,P0,P2>

	Maximum Needs	Current Needs
P_0	10	5
P_1	4	2
P_2	9	2

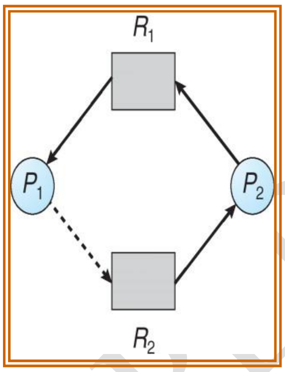
- SupposeprocessP2requestsandisallocated1moretapedrive.
- Question:Istheresultingstatestillsafe?
- Answer:No!Becausetheredoesnotexistasafesequenceanymore.
    - OnlyP1canbeallocateditsmaximumneeds.
    - IFP0andP2request5moredrivesand6moredrives,respectively,thentheresulting state will be deadlocked.

ResourceAllocationGraphAlgorithm

Inthisgraphanewtypeofedgehasbeenintroducedisknownasclaimedge.Claimedge$P_i \rightarrow R_j$indicatesthatprocess$P_j$mayrequestresourceR_j;representedbyadashedline.Claimedgeconverts torequestedgewhenaprocessrequestsaresource.Requestedgeconvertedtoanassignmentedge whentheresourceisallocatedtotheprocess.Whenaresourceisreleasedbyaprocess,assignment edgereconvertstoaclaimedge.Resourcesmustbeclaimedaprioriinthesystem.



- P2requestingR1,butR1isalreadyallocatedtoP1.
- BothprocesseshaveaclaimonresourceR2
- WhathappensifP2nowrequestsresourceR2?



- CannotallocateresourceR2toprocessP2
- Why?Becauseresultingstateisunsafe
  - P1couldrequestR2,therebycreatingdeadlock!

Useonlywhenthereisasingleinstanceofeachresourcetype

- Supposethatprocess$P_i$requestsaresourceR_j
- Therequestcanbegrantedonlyifconvertingtherequestedgetoanassignmentedgedoes not result in the formation of a cycle in the resource allocation graph.

- Herewecheckforsafetybyusingcycle-detectionalgorithm.

Banker'sAlgorithm

Thisalgorithmcanbeusedinbankingsystemtoensurethatthebankneverallocatesallitsavailable cashsuchthatitcannolongersatisfytheneedsofallitscustomer.Thisalgorithmisapplicabletoa systemwithmultipleinstancesofeachresourcetype.Whenanewprocessenterintothesystemit mustdeclare the maximumnumber of instances ofeachresource type thatitmay need.This numbermaynotexceedthetotalnumberofresourcesinthesystem.Severaldatastructuremustbe maintained to implement the banker's algorithm.

Let,

- n=numberofprocesses
- m =numberofresourcestypes

- **Available**:Vectoroflengthm.IfAvailable[j]=k,therearekinstancesofresourcetype $R_j$available.
- **Max**:nxmmatrix.IfMax[i,j]=k,thenprocess$P_i$mayrequestatmostkinstancesofresource type R_j.
- **Allocation**:nxmmatrix.IfAllocation[i,j]=kthen$P_i$iscurrentlyallocatedkinstancesofR_j.
- **Need**:nxmmatrix.IfNeed[i,j]=k,then$P_i$mayneedkmoreinstancesof$R_j$tocompleteits task. Need[i,j]=Max[i,j]–Allocation[i,j].

SafetyAlgorithm

1. LetWorkandFinishbevectorsoflengthmandn,respectively.Initialize:

Work = Available

Finish[i]=falsefori=0,1,…,n-1.

2. Findandisuchthatboth:

(a) Finish[i]=false

(b) $Need_i \leq Work$

Ifnosuchiexists,gotostep4.

3. Work=Work+$Allocation_i$

Finish[i] = true

gotostep2.

4. IfFinish[i]==trueforalli,thenthesystemisinasafestate.

Resource AllocationAlgorithm

Request=requestvectorforprocessP_i.If$Request_i[j]$=kthenprocess$P_i$wantskinstancesof resource type $R_{j.}$

1. If$Request_i \leq Need_i$gotostep2.Otherwise,raiseerrorcondition,sinceprocesshasexceededits maximum claim.

2. If$Request_i \leq$Available,gotostep3.Otherwise$P_i$mustwait,sinceresourcesarenotavailable.

3. PretendtoallocaterequestedresourcestoP$_i$bymodifyingthestateasfollows:

    Available = Available– Request;

    $Allocation_i$=$Allocation_i$+$Request_i$;

    $Need_i$=$Need_i$ – $Request_i$;

- Ifsafe⇒theresourcesareallocatedtoPi.
- Ifunsafe⇒Pimustwait,andtheoldresource-allocationstateisrestored

Example

- 5processesP_0throughP_4;
- 3resourcetypes:
  - A(10instances),B(5instances),andC(7instances).
- SnapshotattimeT$_0$:

	Allocation	Max	Available
	AB C	AB C	AB C
P_0	010	753	332
P_1	200	322	
P_2	302	902	
P_3	211	222	
P_4	002	433	

- ThecontentofthematrixNeedisdefinedtobeMax–Allocation. Need

	AB C
P_0	743
P_1	122
P_2	600
P_3	011

P_4 4 3 1

- Thesystemisinasafestatesincethesequence$<P_1,P_3,P_4,P_2,P_0>$satisfiessafetycriteria.

P_1requests (1, 0, 2)

- CheckthatRequest≤Available(thatis,(1,0,2)≤(3,3,2)⇒true.

Allocation	Need	Available	
	AB C	ABCABC	
P_0	010	743	230
P_1	302	020	
P_2	301	600	
P_3	211	011	
P_4	002	431	

- Executingsafetyalgorithmshowsthatsequence$<P_1,P_3,P_4,P_0,P_2>$satisfiessafety requirement.
- Canrequestfor(3,3,0)byP_4begranted? –NO
- Canrequestfor(0,2,0)byP_0begranted?–NO(ResultsUnsafe)

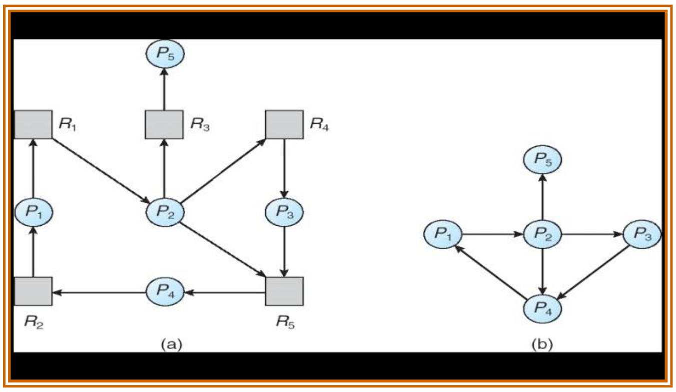
DeadlockDetection

Ifasystemdoesn'temployeitheradeadlockpreventionordeadlockavoidance,thendeadlock situation may occur. In this environment the system must provide

- Analgorithmtorecoverfromthedeadlock.
- Analgorithmtoremovethedeadlockisappliedeithertoasystemwhichpertainssinglein instanceeachresourcetypeorasystemwhichpertainsseveralinstancesofaresourcetype.

SingleInstanceofeachResourcetype

Ifallresourcesonlyasingleinstancethenwecandefineadeadlockdetectionalgorithmwhichusesa newformofresource allocationgraphcalled"Waitfor graph".We obtainthis graphfromthe resource allocationgraphbyremovingthenodesoftyperesourceandcollapsingtheappropriate edges.Thebelowfiguredescribestheresourceallocationgraphandcorrespondingwaitforgraph.



Resource-Allocation Graph

Correspondin wait-forgraph

- Forsingleinstance
- P_i->P_j($P_i$iswaitingfor$P_j$toreleasearesourcethat$P_i$needs)
- P_i->$P_j$existifandonlyifRAGcontains2edgesP_i->$R_q$andR_q->$P_j$forsomeresourceR_q

SeveralInstances ofaResourcetype

Thewaitforgraphschemeisnotapplicabletoaresourceallocationsystemwithmultipleinstances ofreachresourcetype.Forthiscasethealgorithmemploysseveraldatastructureswhicharesimilar tothoseusedinthebanker'salgorithmlikeavailable,allocationandrequest.

- **Available**:Avectoroflengthmindicatesthenumberofavailableresourcesofeachtype.
- **Allocation**:Annxmmatrixdefinesthenumberofresourcesofeachtypecurrently allocated to each process.
- **Request**:Annxmmatrixindicatesthecurrentrequestofeachprocess.IfRequest[i_j]=k, thenprocessP_i is requesting k more instances of resource type. R_j.

1. LetWorkandFinishbevectorsoflengthmandn,respectivelyInitialize:

(a) Work=Available

(b) Fori=1,2,…,n,if$Allocation_i$≠0,then Finish[i] = false;otherwise, Finish[i] = true.

2. Findanindexisuchthatboth:

(a) Finish[i]==false

(b) $\text{Request}_i \leq \text{Work}$

Ifnosuchiexists,gotostep4.

3. Work=Work+Allocation

Finish [i] = true

Gotostep2

4. IfFinish[i]=false,forsomei,1≤i≤n,thenthesystemisinadeadlockstate.Moreover,ifFinish [i]=false,thenprocess$P_i$isdeadlocked.

RecoveryfromDeadlock

Whenadetectionalgorithmdeterminesthatadeadlockexists,severalalternativesexist.One possibility is to inform the operator that a deadlockhas occurred, and to let the operatordealwith thedeadlockmanually.Theotherpossibilityistoletthesystemrecoverfromthedeadlock automatically.Therearetwooptionsforbreakingadeadlock.Onesolutionissimplytoabortoneor more processes to break the circular wait. The second option is to preempt some resources from one or more of the deadlocked processes.

ProcessTermination:

Toeliminatedeadlocksbyabortingaprocess,weuseoneoftwomethods.Inbothmethods,the systemreclaimsallresourcesallocatedtotheterminatedprocesses.

- **Abortalldeadlockedprocesses:**Thismethodclearlywillbreakthedeadlockcycle,butata greatexpense;theseprocessesmayhavecomputedforalongtime,andtheresults ofthese partial computations must be discarded and probably recomputed later.
- **Abortone process ata time until the deadlockcycle is eliminated:**This methodincurs considerableoverhead,sinceaftereachprocessisaborted,adeadlockdetectionalgorithm mustbe invokedtodetermine whetherany processes arestill deadlocked.

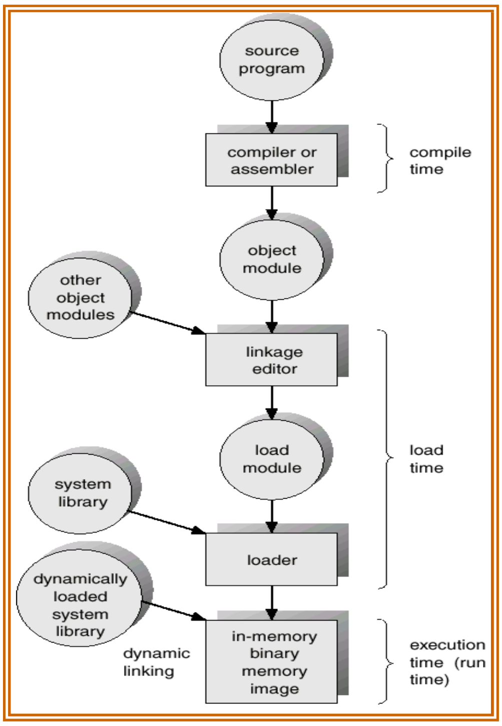
ResourcePreemption:

Toeliminatedeadlocksusingresourcepreemption,wesuccessivelypreemptsomeresourcesfrom processesandgivetheseresources tootherprocessesuntilthedeadlockcycleisbroken.If preemption is required todealwith deadlocks, thenthree issues need to be addressed.

- **Selecting a victim:** Which resources and which processes are to be preempted? As in processtermination,wemustdeterminetheorderofpreemptiontominimizecost.Cost factorsmayincludesuchparametersasthenumbersofresourcesadeadlockprocessis holding,andtheamountoftimeadeadlockedprocesshasthusfarconsumedduringits execution.
- **Rollback:**Ifwepreemptaresourcefromaprocess,whatshouldbedonewiththatprocess? Clearly,itcannotcontinuewithitsnormalexecution;itismissingsomeneededresource.We must rollback the process to some safe state, and restart it from that state.
- **Starvation:**Inasystemwherevictimselectionisbasedprimarilyoncostfactors,itmay happenthatthesameprocessisalwayspickedasavictim.Asaresult,thisprocessnever completesits designatedtask,astarvationsituationthatneeds tobe dealtwithinany practicalsystem.Clearly,wemustensurethataprocesscanbepickedasavictimonlya small finite number of times. The most common solution is to include the number of rollbacks in the cost factor.

MemoryManagement

- Memoryconsistsofalargearrayofwordsorbytes,eachwithitsownaddress.TheCPUfetches instructions from memory according to the value of the program counter. These instructions maycauseadditionalloadingfromandstoringtospecificmemoryaddresses.
- Memoryunitseesonlyastreamofmemoryaddresses.Itdoesnotknowhowtheyaregenerated.
- Programmustbebroughtintomemoryandplacedwithinaprocessforittoberun.
- Inputqueue–collectionofprocessesonthediskthatarewaitingtobebroughtintomemoryfor execution.
- Userprogramsgothroughseveralstepsbeforebeingrun.



Addressbindingofinstructionsanddatatomemoryaddressescanhappenatthreedifferentstages.

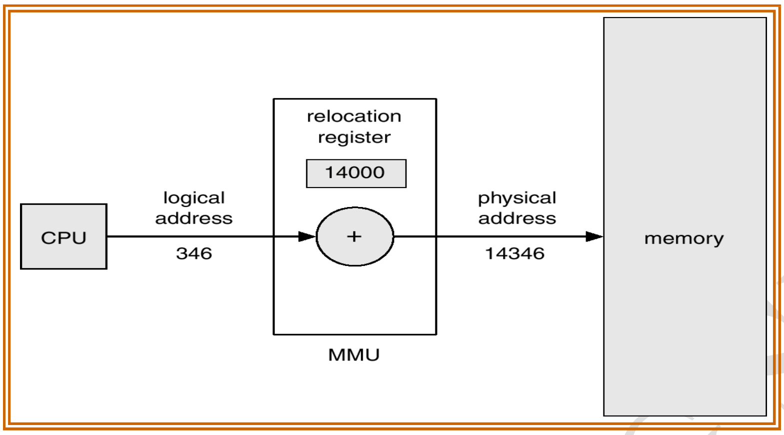
- **Compiletime**:Ifmemorylocationknownapriori,absolutecodecanbegenerated;must recompile code if starting location changes.

  Example:.COM-formatprogramsinMS-DOS.
- **Loadtime**:Mustgeneraterelocatablecodeifmemorylocationisnotknownatcompiletime.
- **Execution time**:Binding delayed until run time if the process can be movedduring its executionfromonememorysegmenttoanother.Needhardwaresupportforaddressmaps (e.g., relocation registers).

LogicalVersusPhysicalAddressSpace

- Theconceptofalogicaladdressspacethatisboundtoaseparatephysicaladdressspaceis central to proper memory management.
  - Logicaladdress–addressgeneratedbytheCPU;alsoreferredtoasvirtualaddress.
  - Physicaladdress–addressseenbythememoryunit.
- Thesetofalllogicaladdressesgeneratedbyaprogramisalogicaladdressspace;thesetofall physicaladdressescorrespondingtotheselogicaladdressesareaphysicaladdressspace.

- Logicalandphysicaladdressesarethesameincompile-timeandload-timeaddress-binding schemes;logical(virtual)andphysicaladdressesdifferinexecution-timeaddress-binding scheme.
- Therun-timemappingfromvirtualtophysicaladdressesisdonebyahardwaredevicecalledthe memory management unit (MMU).



- Thismethodrequireshardwaresupportslightlydifferentfromthehardwareconfiguration.The baseregisterisnowcalledarelocationregister.Thevalueintherelocationregisterisaddedto everyaddressgeneratedbyauserprocess atthetimeitis senttomemory.
- Theuserprogramneversees the realphysicaladdresses.The programcancreateapointerto location346,store itinmemory, manipulate itandcompare it toother addresses.The user programdealswithlogicaladdresses.Thememorymappinghardwareconvertslogicaladdresses intophysicaladdresses.Thefinallocationofareferencedmemoryaddressisnotdetermined until the reference is made.

DynamicLoading

- Routineisnotloadeduntilitiscalled.
- Allroutinesarekeptondiskinarelocatableloadformat.
- Themainprogramisloadedintomemoryandisexecuted.Whenaroutineneedstocallanother routine,thecallingroutinefirstcheckstoseewhethertheotherthedesiredroutineintomemory and to update the program's address tables to reflectthis change.Then control is passed to the newly loaded routine.
- Bettermemory-spaceutilization;unusedroutineisneverloaded.
- Usefulwhenlargeamountsofcodeareneededtohandleinfrequentlyoccurringcases.

- Nospecialsupportfromtheoperatingsystemisrequired.
- Implementedthroughprogramdesign.

DynamicLinking

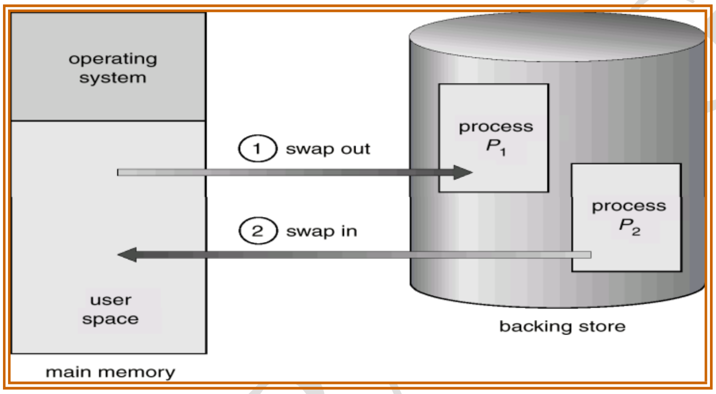
- Linkingispostponeduntilexecutiontime.
- Smallpieceofcode,stub,isusedtolocatetheappropriatememory-residentlibraryroutine,orto load the library if the routine is not already present.
- Whenthisstubisexecuted,itcheckstoseewhethertheneededroutineisalreadyinmemory.If not, the program loads the routine into memory.
- Stubreplacesitselfwiththeaddressoftheroutine,andexecutestheroutine.
- Thusthenexttimethatcodesegmentisreached,thelibraryroutineisexecuteddirectly, incurring no cost for dynamic linking.
- Operatingsystemisneededtocheckifroutineisinprocesses'memoryaddress.
- Dynamic linkingis particularly useful forlibraries.

Swapping

- Aprocesscanbeswappedtemporarilyoutofmemorytoabackingstore,andthenbroughtback intomemory forcontinuedexecution.For example,assume a multiprogramming environment witharoundrobinCPUscheduling algorithm.Whena quantumexpires,thememorymanager will start to swap out the process that just finished, and to swap in another process to the memoryspacethathasbeenfreed.Inthemeantime,theCPUschedulerwillallocateatimeslice tosomeotherprocessinmemory.Wheneachprocessfinisheditsquantum,itwillbeswapped withanotherprocess.Ideally,thememorymanagercanswapprocessesfastenoughthatsome processeswillbeinmemory,readytoexecute,whentheCPUschedulerwantstoreschedulethe CPU.The quantummustalsobesufficientlylarge thatreasonable amounts ofcomputing are done between swaps.
- Rollout,rollin–swappingvariantusedforpriority-basedschedulingalgorithms.Ifahigher priorityprocessarrivesandwantsservice,thememorymanagercanswapoutthelowerpriority processsothatitcanloadandexecute lowerpriorityprocess canbeswappedbackinand continued.Thisvariantissometimescalledrollout,rollin.Normallyaprocessthatisswapped outwillbeswappedbackintothesamememoryspacethatitoccupiedpreviously.This restrictionis dictatedby the process cannot be movedtodifferentlocations. Ifexecutiontime

bindingisbeingused,thenaprocesscanbeswappedintoadifferentmemoryspace,becausethe physical addresses arecomputed during executiontime.

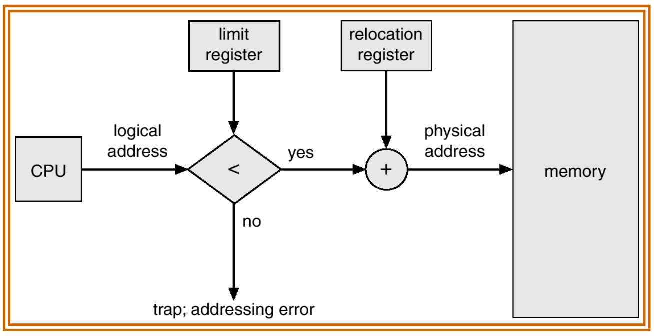
- Backingstore–fastdisklargeenoughtoaccommodatecopiesofallmemoryimagesforall users;must provide directaccess tothese memory images.It mustbe largeenoughto accommodatecopiesofallmemoryimagesforallusers,anditmustprovidedirectaccessto thesememoryimages.Thesystemmaintainsareadyqueueconsistingofallprocesseswhose memoryimagesareschedulerdecidestoexecuteaprocessitcallsthedispatcher.Thedispatcher checks to see whether the next process in the queueis in memory. If not, and there is no free memoryregion,thedispatcherswapsoutaprocesscurrentlyinmemoryandswaps inthe desiredprocess.Itthenreloadsregistersasnormalandtransferscontroltotheselectedprocess.
- Majorpartofswaptimeistransfertime;totaltransfertimeisdirectlyproportionaltothe amount of memory swapped.
- Modifiedversionsofswappingarefoundonmanysystems(i.e.,UNIX,Linux,andWindows).



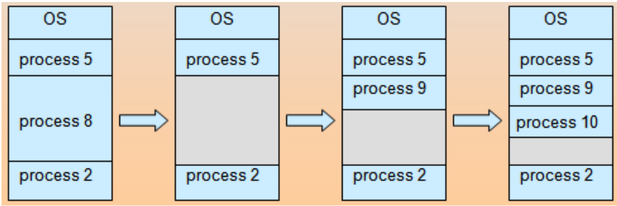
ContiguousMemoryAllocation

- Mainmemoryisusuallydivided intotwo partitions:
  - Residentoperatingsystem,usuallyheldinlowmemorywithinterruptvector.
  - Userprocesses,heldinhighmemory.
- Incontiguousmemoryallocation,eachprocessiscontainedinasinglecontiguoussectionof memory.
- Single-partitionallocation
  - Relocation-registerschemeusedtoprotectuserprocessesfromeachother,andfrom changing operating-system code and data.

- Relocationregistercontainsvalueofsmallestphysicaladdress;limitregistercontains rangeoflogicaladdresses–eachlogicaladdressmustbelessthanthelimitregister.



- Multiple-partitionallocation
  - Hole–blockofavailablememory;holesofvarioussizearescatteredthroughout memory.
  - Whenaprocessarrives,itisallocatedmemoryfromaholelargeenoughto accommodate it.
    - Operatingsystemmaintainsinformationabout:
    a)allocatedpartitions b)freepartitions(hole)
  - Asetofholesofvarioussizes,isscatteredthroughoutmemoryatanygiventime.When aprocessarrivesandneedsmemory,thesystemsearchesthissetforaholethatislarge enoughforthisprocess.Iftheholeistoolarge,itissplitintotwo:onepartisallocated to the arriving process; the other is returned to the set of holes. When a process terminates,itreleasesitsblockofmemory,whichisthenplacedbackinthesetofholes. If the new hold is adjacent to otherholes,these adjacentholesare merged to form one larger hole.
  - Thisprocedureisaparticularinstanceofthegeneraldynamicstorageallocation problem,whichishowtosatisfyarequestofsizenfromalistoffreeholes.Thereare manysolutionstothisproblem.Thesetofholesissearchedtodeterminewhichholeis besttoallocate.Thefirst-fit,best-fitandworst-fitstrategiesarethemostcommonones used to selecta free hole from the set of available holes.



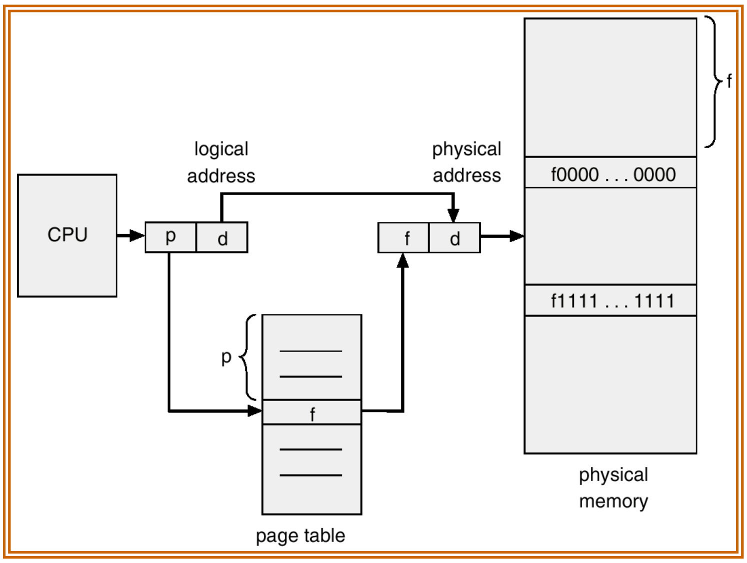
- **First-fit:**Allocatethefirstholethatisbigenough.
- **Best-fit:**Allocatethesmallestholethatisbigenough;mustsearchentirelist,unless ordered by size.
- **Worst-fit:**Allocatethelargesthole;mustalsosearchentirelist.

Fragmentation

- **ExternalFragmentation**–totalmemoryspaceexiststosatisfyarequest,butitisnot contiguous.
- **InternalFragmentation**–allocatedmemorymaybeslightlylargerthanrequestedmemory;this size difference is memory internal to a partition, but not being used.
- Reduceexternalfragmentationbycompaction
  - Shufflememorycontentstoplaceallfreememorytogetherinonelargeblock.
  - Compactionispossibleonlyifrelocationisdynamic,andisdoneatexecutiontime.

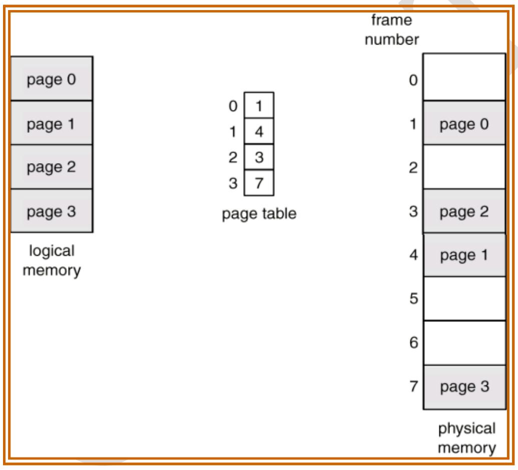
Paging

- Pagingisamemorymanagementschemethatpermitsthephysicaladdressspaceofaprocessto be non contiguous.
- Dividephysicalmemoryintofixed-sizedblockscalled**frames**(sizeispowerof2,forexample 512 bytes).
- Dividelogicalmemoryintoblocksofsamesizecalled**pages**.Whenaprocessistobeexecuted, itspagesareloadedintoanyavailablememoryframesfromthebackingstore.Thebackingstore isdividedintofixedsizedblocksthatareofthesamesizeasthememoryframes.
- Thehardwaresupportforpagingisillustratedinbelowfigure.
- EveryaddressgeneratedbytheCPUisdividedintotwoparts:apagenumber(p)andapage offset(d).Thepagenumberisusedasanindexintoapagetable.Thepagetablecontainsthe baseaddressofeachpageinphysicalmemory.Thisbaseaddressiscombinedwiththepage offset to define the physical memory address that issent to the memory unit.



- Thepagingmodelofmemoryisshowninbelowfigure.Thepagesizeisdefinedbythe hardware.Thesizeofapageistypicallyofapowerof2,varyingbetween512bytesand16MB perpage, depending onthe computer architecture.The selectionof a powerof2 as a pagesize makesthetranslationofalogicaladdressintoapagenumberandpageoffsetparticularlyeasy.If thesizeoflogicaladdressis2^m,andapagesizeis2^naddressingunits,thenthehighorderm-n bitsofalogicaladdressdesignatethepagenumber,andthenlowordebitsdesignatethepage offset.

- Keeptrackofallfreeframes.

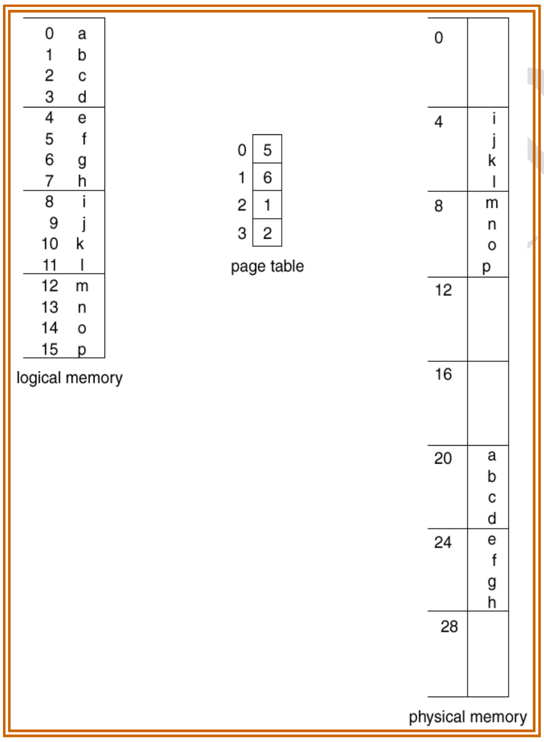


- Torunaprogramofsizenpages,needtofindnfreeframesandloadprogram.
- Setupapagetabletotranslatelogicaltophysicaladdresses.
- Internalfragmentationmayoccur.

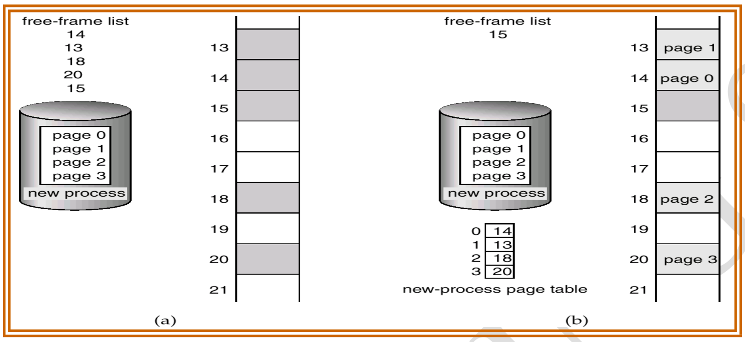
Letustakeanexample.Supposeaprogramneeds32KBmemoryforallocation.Thewhole programisdividedintosmallerunitsassuming4KBandisassignedsomeaddress.Theaddress consists of two parts such as:

- Alargenumberinhigherorderpositionsand
- Displacementoroffsetinthelowerorderbits.

Thenumbersallocatedtopagesaretypicallyinpowerof2tosimplifyextractionofpagenumbers andoffsets.Toaccessapieceofdataatagivenaddress,thesystemfirstextractsthepagenumber andtheoffset.Thenittranslatesthepagenumbertophysicalpageframeandaccessdataatoffsetin physicalpage frame.Atthis moment,thetranslationofthe addressbytheOSisdoneusingapage table.Pagetableisalineararrayindexedbyvirtualpagenumberwhichprovidesthephysicalpage frame that contains the particularpage. It employs a lookup process that extracts the page number andthe offset.The systeminadditionchecks thatthe page number is withinthe address space of processandretrievesthepagenumberinthepagetable.Physicaladdresswillcalculatedbyusingthe formula. Physicaladdress=pagesizeoflogicalmemoryXframenumber+offset

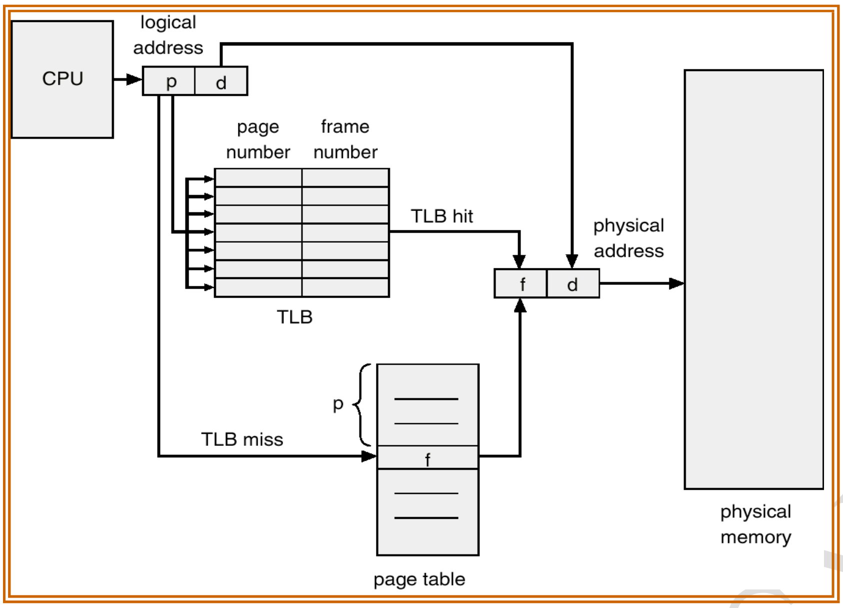


Whenaprocessarrivesinthesystemtobeexecuted,itssizeexpressedinpagesisexamined.Each pageoftheprocessneedsoneframe.Thusiftheprocessrequiresnpages,atleastnframesmustbe availableinmemory.Ifnframesareavailable,theyareallocatedtothisarrivingprocess.Thefirst page of the process is loaded into one of the allocated frames, and the frame number is put in the pagetableforthisprocess.Thenextpageisloadedintoanotherframe,anditsframenumberisput intothepagetableandsoonas inbelowfigure.Animportantaspectofpagingisthe clear separationbetweentheuser’sviewofmemoryandtheactualphysicalmemory.Theuserprogram viewsthatmemoryasonesinglecontiguousspace,containingonlythisoneprogram.Infact,the userprogram is scatteredthroughoutphysical memory,whichalso holds otherprograms.The differencebetweentheuser’sviewofmemoryandtheactualphysicalmemoryisreconciledbythe address-translationhardware.Thelogicaladdressesaretranslatedintophysicaladdresses.This mapping is hidden from the user and is controlledby the operating system.



ImplementationofPageTable

- Pagetableiskeptinmainmemory.
- Page-tablebaseregister(PTBR)pointstothepagetable.
- Inthisschemeeverydata/instruction-byteaccessrequirestwomemoryaccesses.Oneforthe page-table entry and one for the byte.
- Thetwomemoryaccessproblemcanbe solvedbytheuseofaspecialfast-lookuphardware cachecalledassociativeregistersorassociativememoryortranslationlook-asidebuffers(TLBs).
- Typically,thenumberofentriesinaTLBisbetween32and1024.



- TheTLBcontainsonlyafewofthepagetableentries.Whenalogicaladdressisgenerated bytheCPU,itspagenumberispresentedtotheTLB.Ifthepagenumberisfound,itsframe numberisimmediatelyavailableandisusedtoaccessmemory.Thewholetaskmaytakeless than 10 percent longerthan it would if an unmapped memory reference were used.
- IfthepagenumberisnotintheTLB(knownasaTLBmiss),amemoryreferencetothe page table mustbe made.Whenthe frame numberis obtained,we canuse it toaccess memory.

HitRatio

- HitRatio:thepercentageoftimesthatapagenumberisfoundintheassociativeregisters.
- Forexample,ifittakes20nanosecondstosearchtheassociativememoryand100nanoseconds to access memory; for a 98-percent hit ratio, we have

  Effectivememory-accesstime=0.98x120+0.02x220

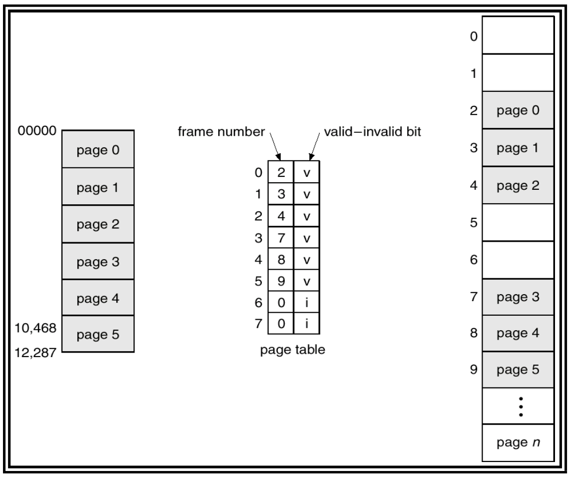
  =122nanoseconds.

- TheIntel80486CPUhas32associativeregisters,andclaimsa98-percenthitratio.

Validorinvalidbitinapagetable

- Memoryprotectionimplementedbyassociatingprotectionbitwitheachframe.
- Valid-invalidbitattachedtoeachentryinthepagetable:
  - "Valid"indicatesthattheassociatedpageisintheprocess'logicaladdressspace,andis thus a legal page.
  - "Invalid"indicatesthatthepageisnotintheprocess'logicaladdressspace.

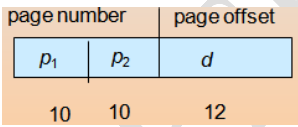
- Payattentiontothefollowingfigure.Theprogramextendstoonlyaddress10,468,anyreference beyondthataddressisillegal.However,referencestopage5areclassifiedasvalid,soaccessesto addressesupto12,287arevalid.Thisreflectstheinternalfragmentationofpaging.



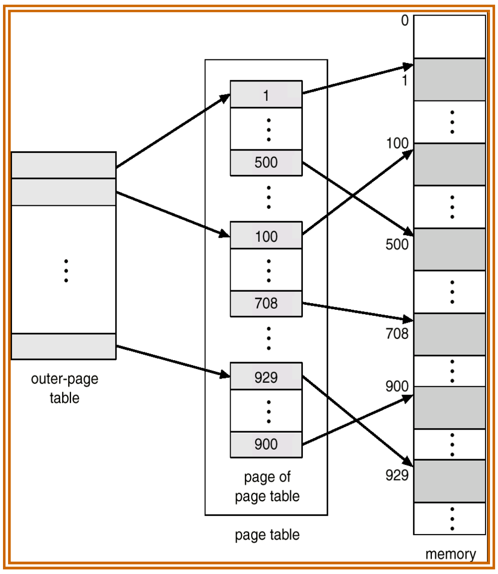
StructureofthePageTable

HierarchicalPaging:

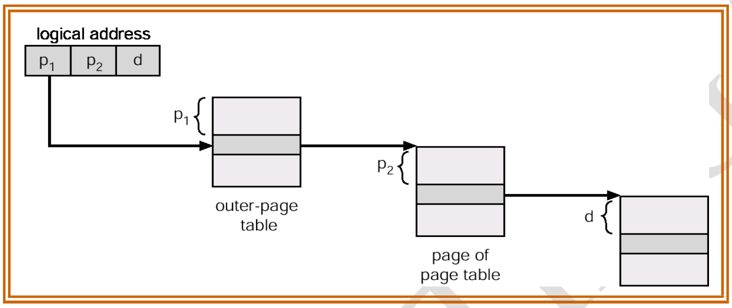
- Alogicaladdress(on32-bitmachinewith4Kpagesize)isdividedinto:
  - Apagenumberconsistingof20bits.
  - Apageoffsetconsistingof12bits.
- Sincethepagetableispaged,thepagenumberisfurtherdividedinto:
  - A10-bitpagenumber.
  - A10-bitpageoffset.
- Thus,alogicaladdressisasfollows:



Where$p_1$isanindexintotheouterpagetable,and$p_2$isthedisplacementwithinthepageofthe outerpagetable.Thebelowfigureshowsatwolevelpagetablescheme.

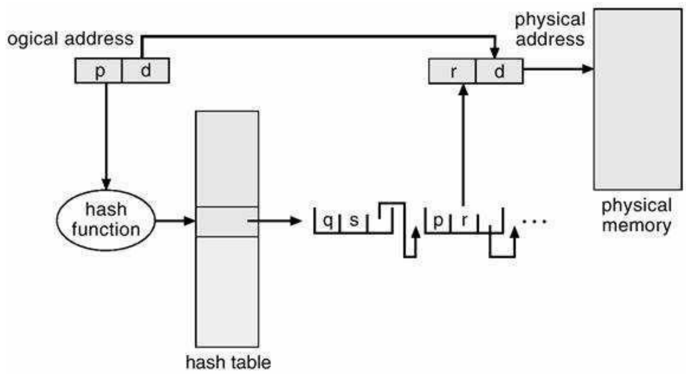


Address-translationschemeforatwo-level32-bitpagingarchitectureisshowninbelowfigure.



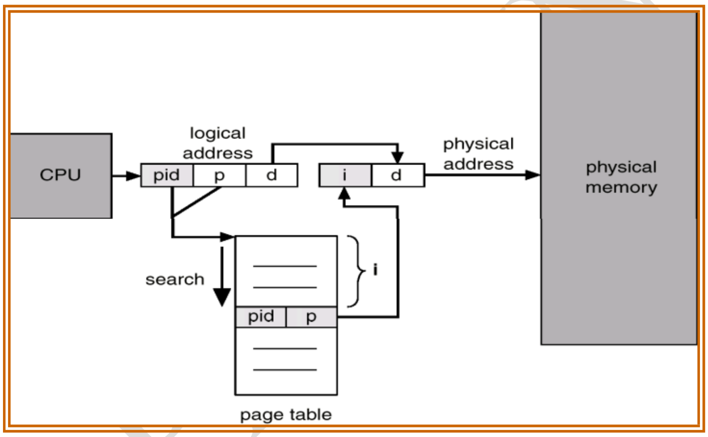
HashedPageTable:

Acommonapproachforhandlingaddressspaceslargerthan32bitsis touseahashedpagetable, withthehashvaluebeingthevirtualpagenumber.Eachentryinthehashtablecontainsalinkedlist ofelementsthathastothesamelocation.Eachelementconsistsofthreefields:(a)thevirtualpage number,(b)thevalueofthemappedpageframe,and(c)apointertothenextelementinthelinked list.Thealgorithmworksasfollows:Thevirtualpagenumberinthevirtualaddressishashedinto thehashtable.Thevirtualpagenumberiscomparedtofield(a)inthefirstelementinthelinkedlist. Ifthereisamatch,thecorrespondingpageframe(field(b))isusedtoformthedesiredphysical address.Ifthereisnomatch,subsequententriesinthelinkedlistaresearchedforamatchingvirtual page number. The scheme is shown in below figure.



InvertedPage Table:

- Oneentryforeachrealpage(frame)ofmemory.
- Entryconsistsofthevirtualaddressofthepagestoredinthatrealmemorylocation,with information about the process that owns that page.
- Thereisonlyonepagetableinthesystem.Notperprocess.
- Decreasesmemoryneededtostoreeachpagetable,butincreasestimeneededtosearchthetable when a page reference occurs.
- Usehashtabletolimitthesearchtoone—oratmostafew—page-tableentries.



Each virtual address in the system consists of a triple <process-id, page-number, offset>. Each invertedpagetableentryisapair<process-id,page-number>wheretheprocess-idassumestherole oftheaddressspaceidentifier.Whenamemoryreferenceoccurs,partofthevirtualaddress, consistingof<process-id,page-number>,ispresentedtothememorysubsystem.Theinvertedpage tableisthensearchedfora match.Ifamatchis foundsayatentryi,thenthephysicaladdress<i, offset> is generated. If nomatchis found, thenanillegaladdress access has beenattempted.

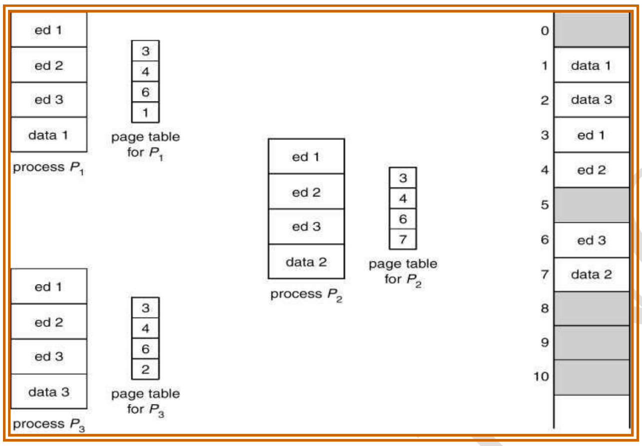
SharedPage:

- Sharedcode

- One copy of read-only (reentrant) code shared among processes (i.e., text editors, compilers, window systems).
- Shared code must appear in same location in the logical address space of all processes.

- Private code and data
  - Each process keeps a separate copy of the code and data.
  - The pages for the private code and data can appear anywhere in the logical address space.

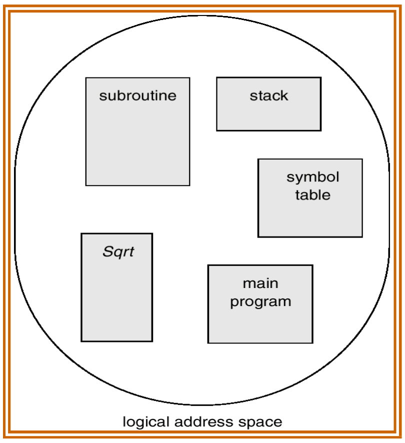
Reentrant code or pure code is non self modifying code. If the code is reentrant, then it never changes during execution. Thus, two or more processes can execute the same code at the same time. Each process has its own copy of registers and data storage to hold the data for the process' execution. The data for two different processes will of course vary for each process.



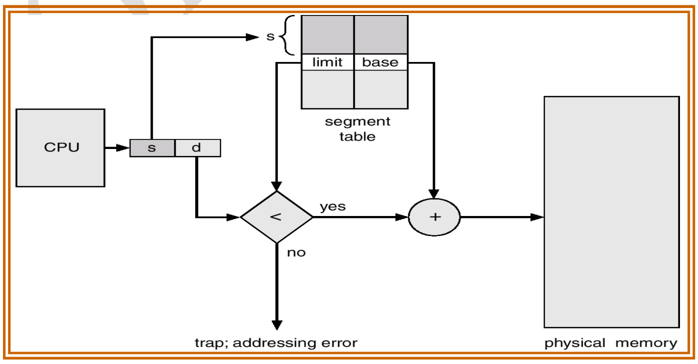
Segmentation

- Memory-management scheme that supports user view of memory.
- A program is a collection of segments. A segment is a logical unit such as:
  - Main program,
  - Procedure,
  - Function,
  - Method,
  - Object,
  - Local variables, global variables,
  - Common block,
  - Stack,

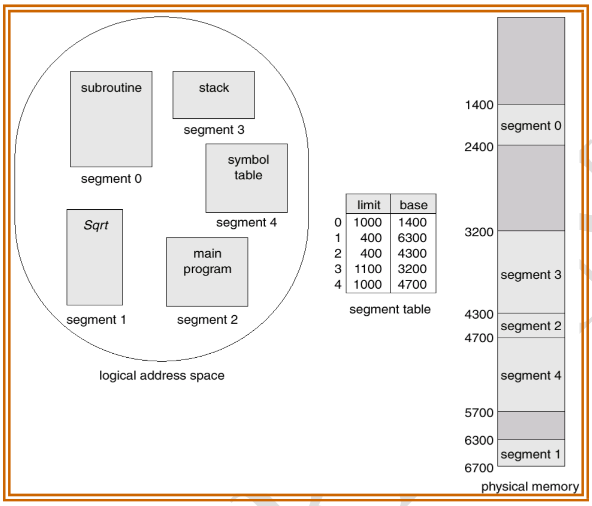
- Symboltable,arrays



- Segmentationisamemorymanagementschemethatsupportsthisuserview ofmemory.
- Alogicaladdressspaceisacollectionofsegments.Eachsegmenthasanameandalength.
- Theaddressesspecifyboththesegmentnameandtheoffsetwithinthesegment.
- Theuserthereforespecifieseachaddressbytwoquantitiessuchassegmentnameandanoffset. Forsimplicityofimplementation,segmentsarenumberedandarereferredtoby asegment number, rather than by a segment name.
- Logicaladdressconsistsofatwotuples:
  - <segment-number,offset>
- Segmenttable–mapstwo-dimensionalphysicaladdresses;eachtableentryhas:
  - Base–containsthestartingphysicaladdresswherethesegmentsresideinmemory.
  - Limit–specifiesthelengthofthesegment.
- Segment-tablebaseregister(STBR)pointstothesegmenttable’slocationinmemory.
- Segment-tablelengthregister(STLR)indicatesnumberofsegmentsusedbyaprogram;
  - Segmentnumbersislegalifs<STLR.



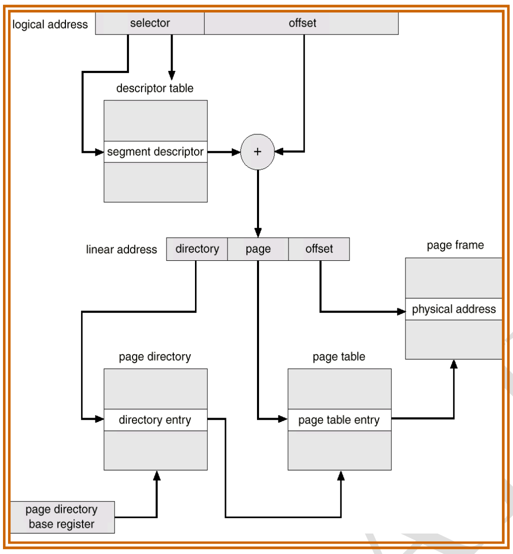
- Whentheuserprogramiscompiledbythecompileritconstructsthesegments.
- Theloadertakesallthesegmentsandassignedthesegmentnumbers.
- Themappingbetweenthelogicalandphysicaladdressusingthesegmentationtechniqueis shown in above figure.
- Eachentryinthesegmenttableaslimitandbaseaddress.
- Thebaseaddresscontainsthestartingphysicaladdressofasegmentwherethelimitaddress specifies the length of the segment.
- Thelogicaladdressconsistsof2partssuchassegmentnumberandoffset.
- Thesegmentnumberisusedasanindexintothesegmenttable.Considerthebelow example is given below.



Segmentationwith Paging

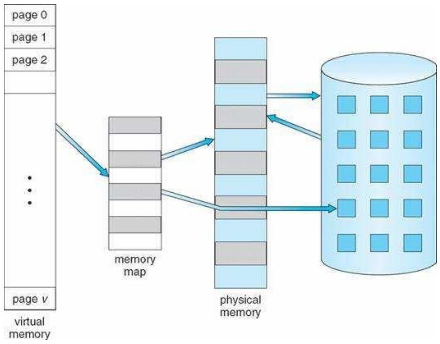
- Bothpagingandsegmentationhaveadvantagesanddisadvantages,that'swhywecancombine these two methods to improve this technique for memory allocation.
- Thesecombinationsare bestillustratedbyarchitectureofIntel-386.
- TheIBMOS/2isanoperatingsystemoftheIntel-386architecture.Inthistechniqueboth segment table and page table is required.
- Theprogramconsistsofvarioussegmentsgivenbythesegmenttablewherethesegmenttable contains different entries one for each segment.
- Theneachsegmentisdividedintoanumberofpagesofequalsizewhoseinformationis maintained in a separate page table.

- Ifaprocesshasfoursegmentsthatis0to3thentherewillbe4pagetablesforthatprocess,one for each segment.
- Thesizefixedinsegmentationtable(SMT)givesthetotalnumberofpagesandtherefore maximum page number in that segment with starting from 0.
- Ifthepagetableorpagemaptableforasegmenthasentriesforpage0to5.
- TheaddressoftheentryinthePMTforthedesiredpagepinagivensegmentscanbeobtained by B + P where B can be obtained from the entry in the segmentation table.
- Usingtheaddress(B+P)asanindexinpagemaptable(pagetable),thepageframe(f)canbe obtainedandphysical address canbe obtainedby adding offsettopage frame.



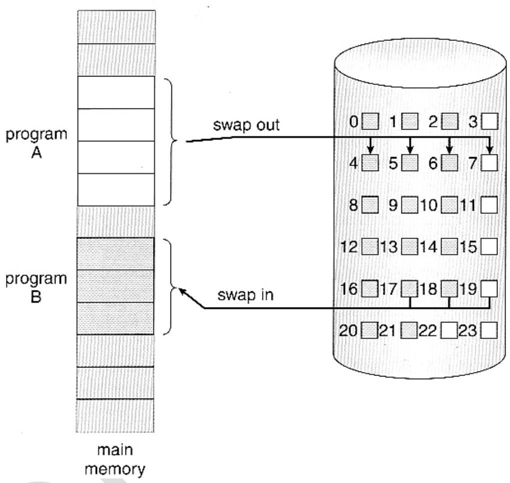
VirtualMemory

- Itisatechniquewhichallowsexecutionofprocessthatmaynotbecompiledwithintheprimary memory.
- Itseparatestheuserlogicalmemoryfromthephysicalmemory.Thisseparationallowsan extremelylargememorytobeprovidedforprogramwhenonlyasmallphysicalmemoryis available.
- Virtualmemorymakesthetaskofprogrammingmucheasierbecausetheprogrammernolonger needs toworking about the amountof the physicalmemory is available or not.
- Thevirtualmemoryallowsfilesandmemorytobesharedbydifferentprocessesbypage sharing.
- Itismostcommonlyimplementedbydemandpaging.

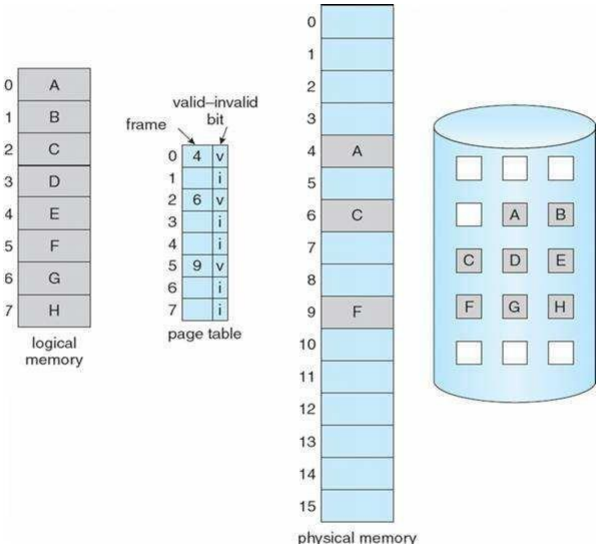


DemandPaging

Ademandpagingsystemissimilartothepagingsystemwithswappingfeature.Whenwewantto executeaprocessweswapitintothememory.Aswappermanipulatesentireprocesswhereasa pagerisconcernedwiththeindividualpagesofaprocess.Thedemandpagingconceptisusingpager ratherthanswapper.Whenaprocessistobeswappedin,thepagerguesseswhichpageswillbeused before theprocessisswappedoutagain.Insteadofswappinginawholeprocess,thepagerbrings onlythosenecessarypagesintomemory.Thetransferofapagedmemorytocontiguousdiskspace is shown in below figure.



Thusitavoidsreadingintomemorypagesthatwillnotusedanywaydecreasingtheswaptimeand the amount of physical memory needed. In this technique we need some hardware support to distinctbetweenthepagesthatareinmemoryandthosethatareonthedisk.Avalidandinvalidbit isusedforthispurpose.Whenthisbitissettovaliditindicatesthattheassociatepageisinmemory. Ifthe bitissettoinvaliditindicatesthatthepageiseithernotvalidoris validbutcurrentlynotin the disk.



Markingapageinvalidwillhavenoeffectiftheprocessneverattemptstoaccessthatpage.Sowhile aprocessexecutesandaccesspagesthatarememoryresident,executionproceedsnormally.Access toapagemarkedinvalidcausesapagefaulttrap.ItistheresultoftheOS'sfailuretobringthe desired page into memory.

Proceduretohandlepagefault

Ifaprocessreferstoapagethatisnotinphysicalmemorythen

- Wecheckaninternaltable(pagetable)forthisprocesstodeterminewhetherthereferencewas valid or invalid.
- Ifthereferencewasinvalid,weterminatetheprocess,ifitwasvalidbutnotyetbroughtin,we have to bring that from main memory.
- Nowwefindafreeframeinmemory.
- Thenwereadthedesiredpageintothenewlyallocatedframe.
- Whenthediskreadiscomplete,wemodifytheinternaltabletoindicatethatthepageisnowin memory.
- Werestarttheinstructionthatwasinterruptedbytheillegaladdresstrap.Nowtheprocesscan accessthepage asifithadalwaysbeeninmemory.

PageReplacement

- Eachprocessisallocatedframes(memory)whichholdtheprocess'spages(data)
- Framesarefilledwithpagesasneeded–thisiscalleddemandpaging

- Over-allocation of memory is prevented by modifying the page-fault service routine to replace pages
- Thejobofthepagereplacementalgorithmistodecidewhichpagegetsvictimizedtomake room for a new page
- Pagereplacementcompletesseparationoflogicalandphysicalmemory

PageReplacementAlgorithm

Optimalalgorithm

- Ideallywewanttoselectanalgorithmwiththelowestpage-faultrate
- Such analgorithmexists, and iscalled(unsurprisingly)theoptimalalgorithm:
- Procedure:replacethepagethatwillnotbeusedforthelongesttime(oratall)–i.e.replacethe page with the greatest forward distance in the reference string
- Exampleusing4frames:

Reference#	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Pagereferenced	**1**	**2**	**3**	**4**	**1**	**2**	**5**	**1**	**2**	**3**	**4**	**5**
Frames	1	1	1	1	1	1	1	1	1	1	4	4
_=faultingpage		2	2	2	2	2	2	2	2	2	2	2
			3	3	3	3	3	3	3	3	3	3
				4	4	4	5	5	5	5	5	5

- Analysis:12 pagereferences, 6 page faults, 2pagereplacements.Pagefaultsper number of frames = 6/4 = 1.5
- Unfortunately,theoptimalalgorithmrequiresspecialhardware(crystalball,magicmirror,etc.) not typically found on today's computers
- Optimalalgorithmisstillusedasametricforjudgingotherpagereplacementalgorithms

FIFOalgorithm

- Replacespagesbasedontheirorderofarrival:oldestpageisreplaced
- Exampleusing4frames:

Reference#	1	2	3	4	5	6	7	8	9	10	11	12
Pagereferenced	**1**	**2**	**3**	**4**	**1**	**2**	**5**	**1**	**2**	**3**	**4**	**5**
Frames	1	1	1	1	1	1	5	5	5	5	4	4
_=faultingpage		2	2	2	2	2	2	1	1	1	1	5
			3	3	3	3	3	3	2	2	2	2
				4	4	4	4	4	4	3	3	3

- Analysis:12 page references, 10 page faults, 6 page replacements.Page faults per number of frames = 10/4 = 2.5

LFUalgorithm(page-based)

- procedure:replacethepagewhichhasbeenreferencedleastoften
- For each page in the reference string, we need to keep a reference count.All reference counts start at0 and are incremented every timea pageis referenced.
- example using 4 frames:

Reference#	1	2	3	4	5	6	7	8	9	10	11	12
Pagereferenced	**1**	**2**	**3**	**4**	**1**	**2**	**5**	**1**	**2**	**3**	**4**	**5**
Frames	$^{1}1$	$^{1}1$	$^{1}1$	$^{1}1$	$^{2}1$	$^{2}1$	$^{2}1$	$^{3}1$	$^{3}1$	$^{3}1$	$^{3}1$	$^{3}1$
_=faultingpage		$^{1}2$	$^{1}2$	$^{1}2$	$^{1}2$	$^{2}2$	$^{2}2$	$^{2}2$	$^{3}2$	$^{3}2$	$^{3}2$	$^{3}2$
^{n}=referencecount			$^{1}3$	$^{1}3$	$^{1}3$	$^{1}3$	$^{1}5$	$^{1}5$	$^{1}5$	$^{2}3$	$^{2}3$	$^{2}5$
				$^{1}4$	$^{1}4$	$^{1}4$	$^{1}4$	$^{1}4$	$^{1}4$	$^{1}4$	$^{2}4$	$^{2}4$

- At the 7th page in the reference string, we need toselect a page to be victimized.Either 3 or 4 willdosincetheyhavethesamereferencecount(1).Let'spick3.
- Likewiseat the10thpagereference;pages4and5havebeenreferencedonceeach.Let'spick page4tovictimize.Page3isbroughtin,anditsreferencecount(whichwas1beforewepaged it out a while ago) is updated to 2.
- Analysis:12 page references, 7 page faults, 3 page replacements.Page faults per number of frames = 7/4 = 1.75

LFUalgorithm(frame-based)

- Procedure:replacethepageintheframewhichhasbeenreferencedleastoften
- Need to keep a reference count for each frame which is initialized to 1 when the page is paged in,incrementedeverytimethepageintheframeisreferenced,andreseteverytimethe pagein the frame is replaced
- Exampleusing4frames:

Reference#	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Pagereferenced	**1**	**2**	**3**	**4**	**1**	**2**	**5**	**1**	**2**	**3**	**4**	**5**
Frames	$^{1}\underline{1}$	$^{1}1$	$^{1}1$	$^{1}1$	$^{2}1$	$^{2}1$	$^{2}1$	$^{3}1$	$^{3}1$	$^{3}1$	$^{3}1$	$^{3}1$
_=faultingpage		$^{1}\underline{2}$	$^{1}2$	$^{1}2$	$^{1}2$	$^{2}2$	$^{2}2$	$^{2}2$	$^{3}2$	$^{3}2$	$^{3}2$	$^{3}2$
^{n}=reference count			$^{1}\underline{3}$	$^{1}3$	$^{1}3$	$^{1}3$	$^{1}\underline{5}$	$^{1}5$	$^{1}5$	$^{1}\underline{3}$	$^{1}3$	$^{1}\underline{5}$
				$^{1}\underline{4}$	$^{1}4$	$^{1}4$	$^{1}4$	$^{1}4$	$^{1}4$	$^{1}4$	$^{2}4$	$^{2}4$

- Atthe7threference,wevictimizethepageintheframewhichhasbeenreferencedleastoften-- in this case, pages 3 and 4 (contained within frames 3 and 4) are candidates, each with a referencecountof1.Let’spickthepageinframe3.Page5ispagedinandframe3’sreference count is reset to 1.
- Atthe10threference,weagainhaveapagefault.Pages5and4(containedwithinframes3and 4)are candidates, eachwitha count of 1.Let’s pick page 4.Page 3is pagedintoframe 3, and frame 3’s reference count is reset to 1.
- Analysis:12 page references, 7 page faults, 3 page replacements.Page faults per number of frames = 7/4 = 1.75

LRUalgorithm

- Replaces pages based on their most recent reference – replace the page with the greatest backward distance in the reference string
- Exampleusing4frames:

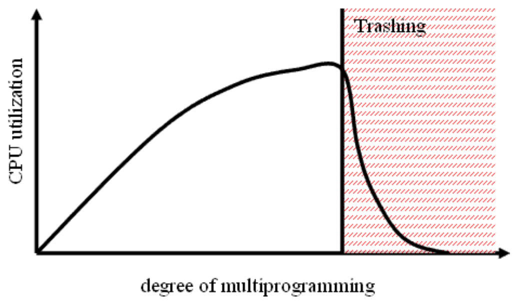
Reference#	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**

Pagereferenced	**1**	**2**	**3**	**4**	**1**	**2**	**5**	**1**	**2**	**3**	**4**	**5**
Frames	1	1	1	1	1	1	1	1	1	1	1	5
_=faultingpage		2	2	2	2	2	2	2	2	2	2	2
			3	3	3	3	5	5	5	5	4	4
				4	4	4	4	4	4	3	3	3

- Analysis:12 pagereferences, 8 page faults, 4pagereplacements.Pagefaultsper number of frames = 8/4 = 2
- Onepossibleimplementation(notnecessarilythebest):
  - Everyframehasatimefield;everytimeapageisreferenced,copythecurrenttimeinto its frame’s time field
  - Whenapageneedstobereplaced,lookatthetimestampstofindtheoldest

Thrashing

- Ifaprocessdoesnothave“enough”pages,thepage-faultrateisveryhigh
  - lowCPUutilization
  - OSthinksitneedsincreasedmultiprogramming
  - addsanotherprocesstosystem
- Thrashingiswhenaprocessisbusyswappingpagesinandout
- Thrashingresultsinsevereperformanceproblems.Considerthefollowingscenario,whichis basedontheactualbehaviourofearlypagingsystems.TheoperatingsystemmonitorsCPU utilization.IfCPUutilizationistoolow,weincreasethedegreeofmultiprogrammingby introducinganewprocesstothesystem.Aglobalpagereplacementalgorithmisused;it replaces pages with no regard to the process to which they belong. Now suppose that a process enters a new phase in its execution and needs more frames.



FILESYSTEM

Fileconcept:

Afileisacollectionofrelatedinformationthatisstoredonsecondarystorage.Informationstoredin filesmustbepersistenti.e.notaffectedbypowerfailures&systemreboots.Filesmaybeoffree fromsuchastextfilesormaybeformattedrigidly.Filesrepresentbothprogramsaswellasdata. PartoftheOSdealingwiththefilesisknownasfilesystem.Theimportantfileconceptsinclude:

1. **Fileattributes:**Afilehascertainattributeswhichvaryfromoneoperatingsystemtoanother.
    - **Name:**Everyfilehasanamebywhichitisreferred.
    - **Identifier:**Itisuniquenumberthatidentifiesthefilewithinthefilesystem.
    - **Type:**Thisinformationisneededforthosesystemsthatsupportdifferenttypesoffiles.
    - **Location:**Itisapointertoadevice&tothelocationofthefileonthatdevice
    - **Size:**Itisthecurrentsizeofafileinbytes,wordsorblocks.
    - **Protection:**Itistheaccesscontrolinformationthatdetermineswhocanread,write& execute a file.
    - **Time,date&useridentification:**Itgivesinformationabouttimeofcreationorlast modification & last use.
2. **Fileoperations:**Theoperatingsystemcanprovidesystemcallstocreate,read,write,reposition, delete and truncate files.
    - **Creatingfiles:**Twostepsarenecessarytocreateafile.First,spacemustbefoundforthe fileinthefilesystem.Secondly,anentrymustbemadeinthedirectoryforthenewfile.
    - **Reading a file:** Data & read from the file at the current position. The systemmust keep a readpointertoknowthelocationinthefilefromwherethenextreadistotakeplace.Once the read has been taken place, the read pointer is updated.

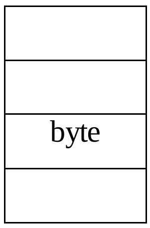
- **Writingafile:**Dataarewrittentothefileatthecurrentposition.Thesystemmustkeepa writepointertoknowthelocationinthefilewherethenextwriteistotakeplace.Thewrite pointer must be updated whenever a write occurs.
- **Repositioningwithina file (seek):**The directory is searchedfor the appropriate entry & thecurrentfilepositionissettoagivenvalue.Afterrepositioningdatacanbereadfromor written into that position.
- **Deletingafile:**Todeleteafile,wesearchthedirectoryfortherequiredfile.Afterdeletion, the space is releasedso that it can be reused by otherfiles.
- **Truncatingafile:**Theusermayerasethecontentsofafilebutallowsallattributesto remainunchangedexpectthefilelengthwhichisrestto‘O’&thespaceisreleased.

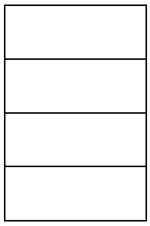
3. **Filetypes:**Thefilenameis spiltinto2parts,Name&extension.Usually these twopartsare separatedbyaperiod.Theuser&theOScanknowthetypeofthefilefromtheextensionitself. Listedbelowaresomefiletypesalongwiththeirextension:

File Type	Extension
Executable File	exe, bin, com
Object File	obj,o(compiled)
Source Code file	C,C++,Java,pas
Batch File	bat,sh(commandstocommandtheinterpreter)
Text File	txt, doc (textual data documents)
ArchieveFile	arc,zip,tar(relatedfilesgroupedtogetherintofilecompressedfor storage)
Multimedia File	mpeg(BinaryfilecontainingaudioorA/Vinformation)

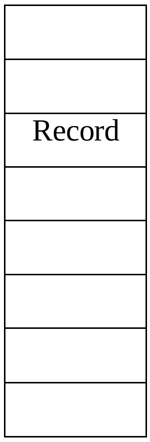
4. **Filestructure:**Filescanbestructuredinseveralways.Threecommonpossibleare:

- **Byte sequence:**The figure shows an unstructured sequence of bytes. The OS doesn’t care aboutthecontentoffile.Itonlyseesthebytes.Thisstructureprovidesmaximumflexibility. Userscanwriteanythingintotheirfiles&namethemaccordingtotheirconvenience.Both UNIX & windows use this approach.





- **Recordsequence:**Inthisstructure,afileisasequenceoffixedlengthrecords.Herethe readoperationreturnsonerecords&thewriteoperationoverwritesorappendorrecord.



- **Tree:**Inthisorganization,afileconsistsofatreeofrecordsofvaryinglengths.Eachrecord consistsofakeyfield.Thetreeisstoredonthekeyfieldtoallowfirstsearchingfora particular key.

Accessmethods: Basically,accessmethodisdividedinto2types:

- **Sequentialaccess:**Itisthesimplestaccessmethod.Informationinthefileisprocessedin orderi.e.onerecordafteranother.Aprocesscanreadallthedatainafileinorderstarting frombeginningbutcan'tskip&readarbitrarilyfromanylocation.Sequentialfilescanbe rewound.It is convenientwhenstorage medium was magnetic tape rather thandisk.
- **Directaccess:**Afileismadeupoffixedlength-logicalrecordsthatallowprogramstoread & write records rapidly inno particularO order. This method can be usedwhen disk are usedforstoringfiles.Thismethodisusedinmanyapplicationse.g.databasesystems.Ifan airlinecustomerwantstoreserveaseatonaparticularflight,thereservationprogrammust beabletoaccesstherecordforthatflightdirectlywithoutreadingtherecordsbeforeit.Ina directaccessfile,thereisnorestrictionintheorderofreadingorwriting.Forexample,we canreadblock14,thenreadblock50&thenwriteblock7etc.Directaccessfilesarevery useful for immediate access to large amount of information.

Directorystructure: Thefilesystemofcomputerscanbeextensive.Somesystemsstorethousands offileondisk.Tomanageallthesedata,weneedtoorganizethem.Theorganizationisdonein2 steps.Thefilesystemisbrokenintopartitions.Eachpartitioncontainsinformationaboutfilewithin it.

Operationona directory:

- **Searchforafile:**Weneedtobeabletosearchadirectoryforaparticularfile.
- **Createafile:**Newfilesarecreated&addedtothedirectory.
- **Deleteafile:**Whenafileisnolongerneeded,wemayremoveitfromthedirectory.
- **Listadirectory:**Weshouldbeabletolistthefilesofthedirectory.
- **Renameafile:**Thenameofafileischangedwhenthecontentsofthefilechanges.
- **Traversethefilesystem:**Itisusefultobeabletoaccesseverydirectory&everyfile within a directory.

Structureofadirectory: Themostcommonschemesfordefiningthestructureofthedirectory are:

1. **Singleleveldirectory:**Itisthesimplestdirectorystructure.Allfilesarepresentinthesame directory. So it is easy to manage & understand.

   Limitation: Asingleleveldirectoryisdifficulttomanagewhentheno.offilesincreasesor whenthereismorethanoneuser.Sinceallfilesareinsamedirectory,theymusthaveunique names. So, there is confusion of file names betweendifferent users.

2. **Twoleveldirectories:**Thesolutiontothenamecollisionprobleminsingleleveldirectoryisto createaseparatedirectoryforeachuser.Inatwoleveldirectorystructure,eachuserhasitsown userfiledirectory.Whenauserlogsin,thenmasterfiledirectoryissearched.Itisindexedby user name & each entry points to the UFD of that user.

   Limitation: Itsolves name collision problem. Butitisolates one user from another. Itis an advantagewhenusersarecompletelyindependent.Butitisadisadvantagewhentheusersneed toaccesseachother'sfiles&co-operateamongthemselvesonaparticulartask.

3. **Treestructureddirectories:**Itisthemostcommondirectorystructure.Atwoleveldirectoryis atwoleveltree.So,thegeneralizationistoextendthedirectorystructuretoatreeofarbitrary height.Itallowsuserstocreatetheirownsubdirectories&organizetheirfiles.Everyfileinthe system has a unique path name. It is the path from the root through all the sub-directories to a specifiedfile.Adirectoryissimplyanotherfilebutitistreatedinaspecialway.Onebitineach

directoryentrydefinestheentryas afile(O)orassub-directories.Eachuserhasacurrent directory.Itcontainsmostofthefilesthatareofcurrentinteresttotheuser.Pathnamescanbe oftwotypes:Anabsolutepathnamebeginsfromtherootdirectory&followsthepathdownto thespecifiedfiles.Arelativepathnamedefines thepathfromthecurrentdirectory.E.g.Ifthe current directory is root/spell/mail, then the relative path name is prt/first & the absolute path name is root/ spell/ mail/ prt/ first. Here users can access the files of other users also by specifying their path names.

4. **Acyclicgraphdirectory:**Itisageneralizationoftreestructureddirectoryscheme.Anacyclic graphallowsdirectoriestohavesharedsub-directories&files.Ashareddirectoryorfileisnot thesameastwocopiesofafile.Hereaprogrammercanviewthecopybutthechangesmadein thefilebyone programmerarenotreflectedintheother'scopy.Butinasharedfile,there is onlyoneactualfile.Somanychangesmadebyapersonwouldbeimmediatelyvisibletoothers. Thisschemeisusefulinasituationwhereseveralpeopleareworkingasateam.So,hereallthe filesthataretobesharedareputtogetherinonedirectory.Sharedfilesandsub-directoriescan beimplementedinseveralways.AcommonwayusedinUNIXsystemsistocreateanew directoryentrycalledlink.Itisapointertoanotherfileorsub-directory.Theotherapproachis toduplicateallinformationinbothsharingdirectories.Acyclicgraphstructureismoreflexible then a tree structure but it is also more complex.

   Limitation: Nowafilemayhavemultipleabsolutepathnames.So,distinctfilenamesmayrefer tothesamefile.Anotherproblemoccursduringdeletionofasharedfile.Whenafileisremoved byanyoneuser.Itmayleavedanglingpointertothenonexistingfile.Oneseriousproblemina cyclicgraphstructureisensuringthattherearenocycles.Toavoidtheseproblems,some systemsdonotallowshareddirectoriesorfiles.E.g.MS-DOSusesatreestructureratherthana cyclic toavoidthe problemsassociatedwithdeletion.Oneapproachfordeletionistopreserve thefileuntilallreferencestoitaredeleted.Toimplementthis approach,wemusthavesome mechanism fordeterminingthelastreference tothefile.Forthis wehave tokeepalistof referencetoafile.Butduetothelargesizeoftheno.ofreferences.Whenthecountiszero,the file can be deleted.

5. **Generalgraphdirectory:**Whenlinksareaddedtoanexistingtreestructureddirectory,the treestructureisdestroyed,resultinginasimplegraphstructure.Linkingisatechniquethat allowsafiletoappearinmorethanonedirectory.Theadvantageisthesimplicityofalgorithmto transversethegraph&determines whentherearenomorereferencestoa file.Butasimilar

problemexistswhenwearetryingtodeterminewhenafilecanbedeleted.Herealsoavalue zerointhereferencecountmeansthattherearenomorereferencestothefileordirectory&the filecanbedeleted.Butwhencycleexists,thereferencecountmaybenon-zeroevenwhenthere arenoreferencestothedirectoryorfile.Thisoccursduetothepossibilityofselfreferencing (cycle)inthestructure.So,herewehavetousegarbagecollectionschemetodeterminewhen thelastreferencestoafilehasbeendeleted&thespacecanbereallocated.Itinvolvestwosteps:

- Transversetheentirefilesystem&markeverythingthatcanbeaccessed.
- Everythingthatisn’tmarkedisaddedtothelistoffreespace.

Butthisprocessisextremelytimeconsuming.Itisonlynecessaryduetopresenceofcyclesin thegraph.So,acyclicgraphstructure iseasiertoworkthanthis.

Protection

Wheninformationiskeptinacomputersystem,amajorconcernisitsprotectionfromphysical damage(reliability)aswellasimproperaccess.

Typesofaccess: Incaseofsystemsthatdon’tpermitaccesstothefilesofotherusers.Protection isnotneeded.So,oneextremeistoprovideprotectionbyprohibitingaccess.Theotherextremeis to provide free accesswith no protection.Both these approachesare too extreme for general use. So,weneedcontrolledaccess.Itisprovidedbylimitingthetypesoffileaccess.Accessispermitted depending onseveralfactors.Onemajorfactoristypeofaccessrequested.Thedifferenttypeof operations that can be controlled are:

- **Read**
- **Write**
- **Execute**
- **Append**
- **Delete**
- **List**

Accesslistsandgroups:

Varioususersmayneeddifferenttypesofaccesstoafileordirectory.So,wecanassociateanaccess listswitheachfileanddirectorytoimplementidentitydependentaccess.Whenauseraccess requestsaccesstoaparticularfile,theOScheckstheaccesslistassociatedwiththatfile.Ifthatuser isgrantedtherequestedaccess,thentheaccessisallowed.Otherwise,aprotectionviolationoccurs &theuserisdeniedaccesstothefile.Butthemainproblemwithaccesslistsistheirlength.Itis

verytedioustoconstructsuchalist.So,weuseacondensedversionoftheaccesslistbyclassifying the users into 3 categories:

- **Owners:**Theuserwhocreatedthefile.
- **Group:**Asetofuserswhoaresharingthefiles.
- **Others:**Allotherusersinthesystem.

Hereonly3fieldsarerequiredtodefineprotection.Eachfieldisacollectionofbitseachofwhich eitherallowsorpreventstheaccess.E.g.TheUNIXfilesystemdefines3fieldsof3bitseach:rwx

- r(readaccess)
- w(writeaccess)
- x(executeaccess)

Separatefieldsarekeptforfileowners,group&otherusers.So,abitisneededtorecordprotection information for each file.

Allocation methods

Thereare3methodsofallocatingdiskspacewidelyused.

1. **Contiguousallocation:**
   a. Itrequireseachfiletooccupyasetofcontiguousblocksonthedisk.
   b. Numberofdiskseeksrequiredforaccessingcontiguouslyallocatedfileisminimum.
   c. TheIBMVM/CMSOSusescontiguousallocation.Contiguousallocationofafileisdefined by the disk address and length (in terms of block units).
   d. Ifthefileis‘n’blockslongandstartsallocation‘b’,thenitoccupiesblocksb,b+1,b+2,---- -------------- --b+n-1.
   e. Thedirectoryforeachfileindicatestheaddressofthestartingblockandthelengthofthe area allocated for each file.
   f. Contiguousallocationsupportsbothsequentialanddirectaccess.Forsequentialaccess,the filesystemremembersthediskaddressofthelastblockreferencedandreadsthenextblock when necessary.
   g. Fordirectaccesstoblockiofafilethatstartsatblockbwecanimmediatelyaccessblockb +i.
   h. **Problems:**Onedifficultywithcontiguousallocationisfindingspaceforanewfile.Italso suffersfromthe problemofexternalfragmentation.Asfilesare deletedandallocated,the freediskspaceisbrokenintosmallpieces.Amajorproblemincontiguousallocationishow

muchspaceisneededforafile.Whenafileiscreated,thetotalamountofspaceitwillneed mustbefoundandallocated.Evenifthetotalamountofspaceneededforafileisknownin advances,pre-allocationisinefficient.Becauseafilethatgrowsveryslowlymustbeallocated enough space for its finalsizeeven though mostofthat space is leftunused for a long periodtime. Therefore,the file hasa large amount of internal fragmentation.

2. **LinkedAllocation:**
   a. Linkedallocationsolvesallproblemsofcontiguousallocation.
   b. Inlinkedallocation,eachfileislinkedlistofdiskblocks,whicharescatteredthroughoutthe disk.
   c. Thedirectorycontainsapointertothefirstandlastblocksofthefile.
   d. Eachblockcontainsapointertothenextblock.
   e. Thesepointersarenotaccessibletotheuser.Tocreateanewfile,wesimplycreateanew entry in the directory.
   f. Forwritingtothefile,afreeblockisfoundbythefreespacemanagementsystemandthis new block is written to & linked to the end of the file.
   g. Toreadafile,wereadblocksbyfollowingthepointersfromblocktoblock.
   h. Thereisnoexternalfragmentationwithlinkedallocation&anyfreeblockcanbeusedto satisfy a request.
   i. Alsothereisnoneedtodeclarethesizeofafilewhenthatfileiscreated.Afilecancontinue to grow as long as there are free blocks.
   j. **Limitations:**Itcanbeusedeffectivelyonlyforsequentialaccessfiles.Tofindthe‘i'th blockofthefile,wemuststartatthebeginningofthatfileandfollowthepointersuntilwe getthe ithblock. Soit is inefficienttosupportdirectaccess files. Due tothe presence of pointerseachfilerequiresslightlymorespacethanbefore.Anotherproblemisreliability. Sincethefilesarelinkedtogetherbypointersscatteredthroughoutthedisk.Whatwould happen if a pointer were lost or damaged.
3. **IndexedAllocation:**
   a. Indexedallocationsolvestheproblemoflinkedallocationbybringingallthepointers together to one location known as the index block.
   b. Eachfilehasitsownindexblockwhichisanarrayofdiskblockaddresses.Theithentryin the index block points to the ith block of the file.

c. Thedirectorycontainstheaddressoftheindexblock.Thereadtheithblock,weusethe pointer in the ith index block entry and read the desired block.
d. Towriteintotheithblock,afreeblockisobtainedfromthefreespacemanagerandits address is put in the ith index block entry.
e. Indexedallocationsupportsdirectaccesswithoutsufferingexternalfragmentation.
f. **Limitations:** The pointer overhead of index block is greater than the pointer overhead of linkedallocation.Soheremorespaceiswastedthanlinkedallocation.Inindexedallocation, an entire index block must be allocated,even if most ofthe pointers are nil.

FreeSpaceManagement

Sincethereisonlyalimitedamountofdiskspace,itisnecessarytoreusethespacefromthedeleted files.Tokeeptrackoffreediskspace,thesystemmaintainsafreespacelist.Itrecordsallthedisk blocksthatarefreei.e.notallocatedtosomefileordictionary.Tocreateafile,wesearchthefree spacelistfortherequiredamountofspaceandallocateittothenewfile.Thisspaceisthenremoved fromthefreespacelist.Whenafileisdeleted,itsdiskspaceisaddedtothefreespacelist.

Implementation:

Thereare4waystoimplementthefreespacelistsuchas:

- **BitVector:**Thefreespacelistis implementedasabitmaporbitvector.Eachblockis representedas 1bit.Iftheblockis free,the bitis1andifitisallocatedthenthebitis0.For example,consideradiskwhereblocks2,3,4,5,8,9,10,11,12,13,17,18,25,26&27arefree and rest of the blocks are allocated. The free space bit map would be
  0011110011111100011000000111……………………..
  Themainadvantageofthisapproachisthatitissimpleandefficienttofindthefirstfreeblock ornconsecutivefreeblocksonthedisk.Butbitvectorsareinefficientunlesstheentirevectoris kept in main memory. Itis possible forsmallerdisks butnotforlarger ones.
- **LinkedList:**Anotherapproachistolinktogetherallthefreediskblocksandkeepapointerto the firstfree block. The firstfree block contains a pointerto the nextfree block and so on.For example,we keepapointertoblock2asthe freeblock.Block2containsapointertoblock whichpointstoblock4whichthenpointstoblock5andsoon.Butthisschemeisnotefficient. To traverse the list,we must read each block whichrequire a lot of I/Otime.

- **Grouping:** In this approach, westore the address ofn free blocks in the firstfree block. The firstn-1oftheseblocksisactuallyfree.Thelastblockcontains the addressofanothernfree blocksandsoon.Heretheaddressesofalargenumberoffreeblockscanbefoundoutquickly.
- **Counting:**Rather than keeping a list of n free disk block addresses,we can keep the address of the first free block and the number offree contiguous blocks. So here each entry in the free space list consists of a disk address and a count.