UNIT-I

COMPUTERSYSTEMOVERVIEW

ComputerSystemOverview-BasicElements,InstructionExecution,Interrupts,MemoryHierarchy,Cache Memory,DirectMemoryAccess,MultiprocessorandMulticoreOrganization.Operatingsystemoverview-objectives and functions, Evolution of Operating System.- Computer System Organization- Operating System Structure and Operations- System Calls, System Programs, OS Generation and System Boot.

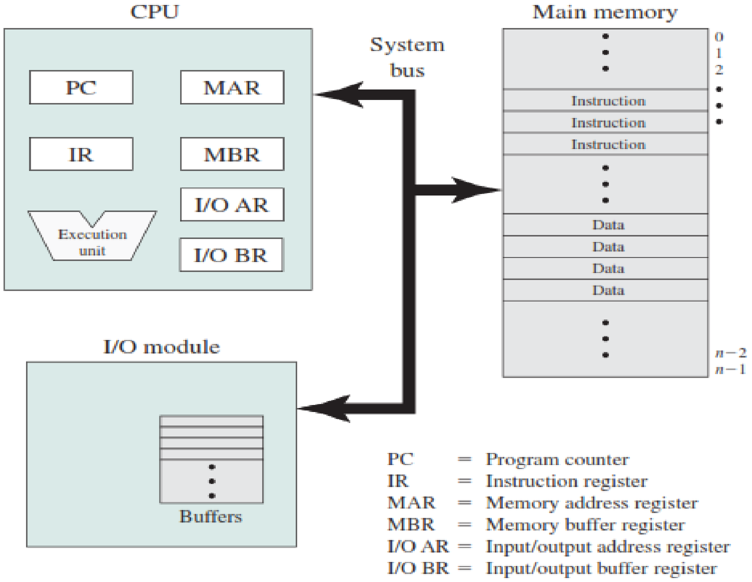
BASICELEMENTSOFACOMPUTER.

➔Atatoplevel,acomputerconsistsofprocessor,memory,andI/Ocomponents,with

oneormoremodulesofeachtype.Thesecomponentsareinterconnectedinsomefashion to achieve the main function of the computer, which is to execute programs.

Thus,therearefourmainstructuralelements:

- **Processor:** Controls the operation of the computer and performs its data processing functions. When there is only one processor, it is often referred to as the central processing unit (CPU).
- **Main memory:** Stores data and programs. This memory is typically volatile; that is, when the computerisshutdown,thecontentsofthememoryarelost.Incontrast,thecontentsofdiskmemory are retained even when thecomputer system is shut down. Mainmemory isalso referredto asreal memory or primary memory.
- **I/O modules:** Move data between the computer and its external environment. The external environment consists of a variety of devices, including secondary memory devices (e.g., disks), communications equipment, and terminals.
- **Systembus:**Providesforcommunicationamongprocessors,mainmemory,andI/Omodules.


CPU
PC
MAR
IR
MBR
I/O AR
Execution unit
I/O BR
System bus
Main memory
0
1
2
Instruction
Instruction
Instruction
Data
Data
Data
Data
n−2
n−1
I/O module
Buffers
PC = Program counter
IR = Instruction register
MAR = Memory address register
MBR = Memory buffer register
I/O AR = Input/output address register
I/O BR = Input/output buffer register

Computer Components: Top-Level View

➔Thefiguredepictsthesetop-levelcomponents.Oneoftheprocessor'sfunctionsistoexchange data with memory. For this purpose, it typically makes use of two internal (to the processor)registers:amemoryaddressregister(MAR),whichspecifiestheaddressinmemoryforthenext read or write; and a memory buffer register (MBR), which contains the data to be written into memoryor which receives the data read from memory.

➔Similarly, an I/O address register (I/OAR) specifies a particular I/O device. An I/O buffer register (I/OBR) is used for the exchange of data between an I/O module and theprocessor.

➔Amemorymoduleconsistsofasetoflocations,definedbysequentiallynumbered addresses.Eachlocationcontainsabitpatternthatcanbeinterpretedaseitheraninstruction or data. An I/O module transfers data from external devices to processor and memory,andviceversa. It contains internal buffers for temporarily holding data until they can be sent on.

INSTRUCTIONEXECUTIONWITHINSTRUCTIONEXECUTIONCYCLE.

➔Aprogramtobeexecutedbyaprocessorconsistsofasetofinstructionsstoredin memory. In its simplest form, instruction processing consists of two steps:

➔The processor reads (fetches) instructions from memory one at a time and executes each instruction. Program execution consists of repeating the process of instruction fetch and instruction execution.

➔The processing required for a single instruction is called an instruction cycle. Using a simplified two-step description, the instruction cycle is depicted in Figure.

Thetwostepsarereferredtoasthe

(i)Fetchstage(ii)Executionstage.

➔Programexecutionhaltsonlyiftheprocessoristurnedoff,somesortofunrecoverableerror occurs, or a program instruction that halts the processor is encountered.

➔The program counter (PC) holds the address of the next instruction to be fetched. Unless instructed otherwise, the processor always increments the PC after eachinstruction fetch so that it will fetch the next instruction in sequence.

➔For example, consider a simplified computer in which each instruction occupies one 16-bit word of memory. Assume that the program counter is set to location 300. The processorwill next fetch the instruction at location 300. On succeeding instructioncycles, it will fetch instructions from locations 301, 302, 303, and so on. This sequence may be altered,as explained subsequently.

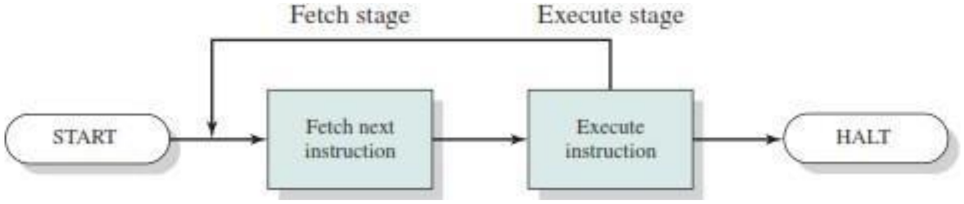
➔Thefetchedinstructionisloadedintotheinstructionregister(IR).Theinstructioncontains bitsthatspecifytheactiontheprocessoristotake.Theprocessorinterprets theinstructionandperformstherequiredaction.

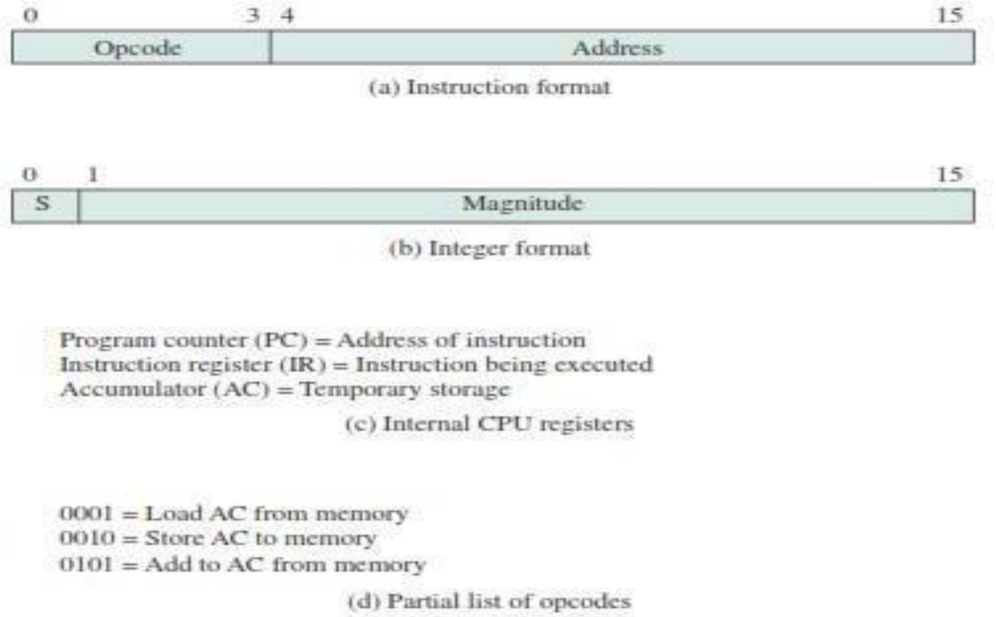
➔Ingeneral,theseactionsfallintofourcategories:

- **Processor-memory**:Datamaybetransferredfromprocessortomemoryorfrommemory to processor.
- **Processor-I/O**: Data may be transferred to or from a peripheral device by transferring between the processor and an I/O module.
- **Dataprocessing**:Theprocessormayperformsomearithmeticorlogicoperationondata.
- **Control:** An instruction may specify that the sequence of execution be altered. For example, the processor may fetch an instruction from location 149, which specifies that the next instruction will befrom location 182. The processor setsthe programcounterto

182.Thus,onthenextfetchstage,theinstructionwillbefetchedfromlocation182 rather than 150.

Basic Instruction Cycle





(a) Instruction format

(b) Integer format

Program counter (PC) = Address of instruction
Instruction register (IR) = Instruction being executed
Accumulator (AC) = Temporary storage

(c) Internal CPU registers

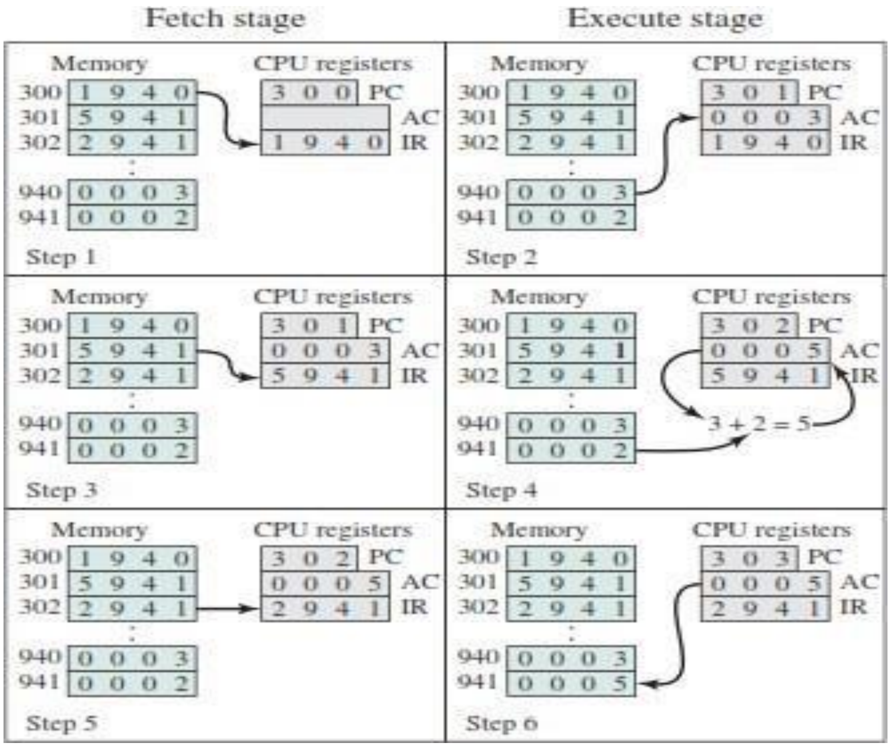
0001 = Load AC from memory
0010 = Store AC to memory
0101 = Add to AC from memory

(d) Partial list of opcodes

➔Figure shows a partial program execution, showing the relevant portions of memory and processor registers. The program fragment shown adds the contents of the memory word at address 940 to the contents of the memory word at address 941 and stores the result in the latter location.

Threeinstructions,whichcanbedescribedasthreefetchandthreeexecutestages,arerequired:

1. The PC contains 300, the address of the first instruction. This instruction (the value 1940 in hexadecimal) is loaded into the IR and the PC is incremented. Note that this process involves the use of a memory address register (MAR) and a memory buffer register (MBR). For simplicity, these intermediate registers are not shown.
2. Thefirst4bits(firsthexadecimaldigit)intheIRindicatethattheACistobeloadedfrommemory. The remaining 12 bits (three hexadecimal digits) specify the address, which is 940.



Example of Program Execution (contents of memory and registers in hexadecimal)

3. Thenextinstruction(5941)isfetchedfromlocation301andthePCisincremented.
4. TheoldcontentsoftheACandthecontentsoflocation941areaddedandtheresultisstoredinthe AC.
5. Thenextinstruction(2941)isfetchedfromlocation302andthePCisincremented.
6. ThecontentsoftheACarestoredinlocation941.

In this example, three instruction cycles, each consisting of a fetch stage and an execute stage, are neededtoaddthecontentsoflocation940tothecontentsof941.Withamorecomplexsetofinstructions, fewer instruction cycles would be needed.

INTERRUPTPROCESSING.

➔Virtually all computers provide a mechanism by which other modules (I/O , memory) may interrupt the normal sequence of the processor. Interrupts are provided primarily as a way to improve processor utilization.

FourClassesofInterruptsare

1. **Program** Generated by some conditionthat occurs asa result of an instruction execution, suchas arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, and reference outside a user’s allowed memory space.
2. **Timer** Generated by a timer within the processor. This allows the operating system to perform certain functions on a regular basis.
3. **I/O** Generated by an I/O controller, to signal normal completion of an operation or to signal a variety of error conditions.
4. **Hardwarefailure**Generatedbyafailure,suchaspowerfailureormemoryparityerror.

➔The user program performs a series of WRITE calls interleaved with processing. The solid verticallinesrepresentsegmentsofcodeinaprogram.Codesegments1,2,and3refertosequences ofinstructionsthatdonotinvolveI/O.TheWRITEcallsaretoanI/Oroutinethatisasystemutility and that will perform the actual I/O operation.

TheI/Oprogramconsistsofthreesections:

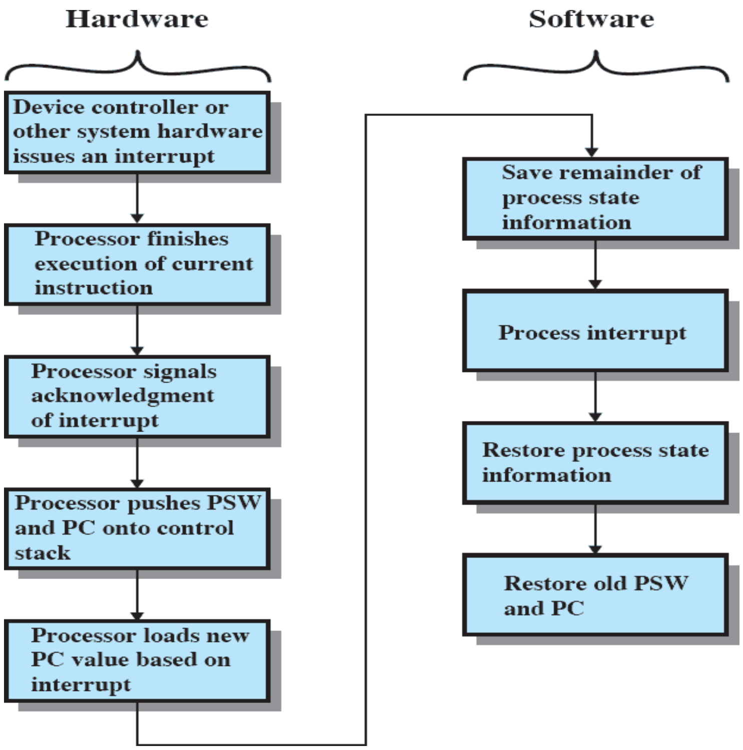
• A sequence of instructions, labeled 4 in the figure, to prepare for the actual I/O operation. This may include copying the data to be output into a special buffer and preparing the parameters for a device command.

• The actual I/O command. Without the use of interrupts, once this command is issued, the program must wait for the I/O device to perform the requested function (or periodicallycheckthe status, or poll, the I/O device). The program might wait by simply repeatedly performing a test operation to determine if the I/O operation is done.

• Asequenceofinstructions,labeled5inthefigure,tocompletetheoperation.Thismay include setting a flag indicating the success or failure of the operation.

INTERRUPTPROCESSING

Thefollowinggivesthedetailedinterruptprocessingprocedure:


Hardware
Software
Device controller or other system hardware issues an interrupt
Processor finishes execution of current instruction
Processor signals acknowledgment of interrupt
Processor pushes PSW and PC onto control stack
Processor loads new PC value based on interrupt
Save remainder of process state information
Process interrupt
Restore process state information
Restore old PSW and PC

➔Aninterrupttriggersanumberofevents,bothintheprocessorhardwareandinsoftware.

This figure shows a typical sequence. When an I/O device completes an I/O operation, the following sequence of hardware events occurs:

1. Thedeviceissuesaninterruptsignaltotheprocessor.
2. Theprocessorfinishesexecutionofthecurrentinstructionbeforerespondingto the interrupt.
3. Theprocessortestsforapendinginterruptrequest,determinesthatthereisone, and sends an acknowledgment signal to the device that issued the interrupt. The acknowledgment allows the device to remove its interrupt signal.
4. Theprocessornextneedstopreparetotransfercontroltotheinterruptroutine.
5. Theprocessorthenloadstheprogramcounterwiththeentrylocationofthe interrupt- handling routine that will respond to this interrupt.
6. Atthispoint,theprogramcounterandPSWrelatingtotheinterruptedprogram have been saved on the control stack.
7. Theinterrupthandlermaynowproceedtoprocesstheinterrupt.
8. Thesavedregistervaluesareretrievedfromthestackandrestoredtothe registers
9. ThefinalactistorestorethePSWandprogramcountervaluesfromthestack. It is important to save all of the state information about the interrupted program for later resumption.

Becausetheinterruptisnotaroutinecalledfromtheprogram.

Rather,theinterruptcanoccuratanytimeandthereforeatany pointin the execution of a user program.

Itsoccurrenceisunpredictable.

MULTIPLE INTERRUPTS

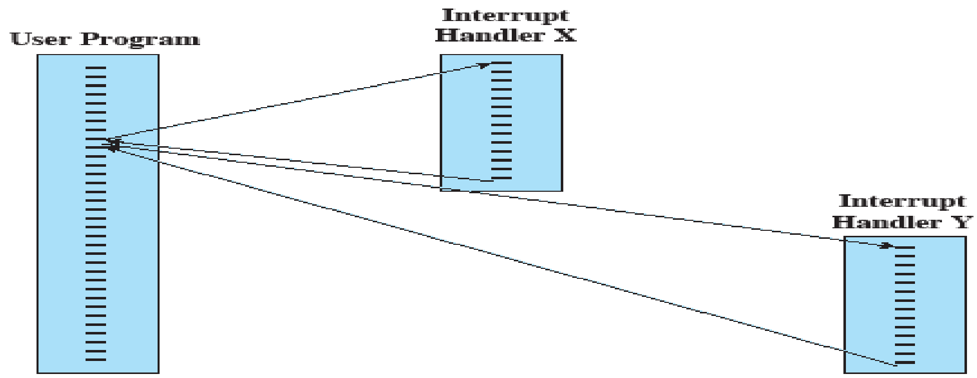
➔Theaboveonly discussedthecase in which a single interrupt happens.Actually,inacomputersystem,therearemultipleinterruptsignalsources,somorethanoneinterruptrequests may happenatthesame timeor duringa same period.

➔The typical two approaches are: sequential interrupt processing-by disabling interruptrequestwhileaninterruptisbeingprocessed,allinterruptswillbeprocessedsequentially(usually PSWcontainsabitforthispurpose);nestedinterruptprocessing-allthe interruptsmaybeassigneddifferentpriorities,sothatwheneveraninterruptoccurswhilean interrupthandlerisrunning,theirprioritieswillbecomparedfirst,andthefurtheraction willbedeterminedaccordingtotheresult. Thesetwoapproaches are illustrated bythefollowingfigures:

a) SequentialInterruptProcessing

➔Twoapproachescanbetakentodealingwithmultipleinterrupts.Thefirstisto disable interrupts while an interrupt is being processed. A *disabled interrupt* simply means that the processor ignores any new interrupt request signal. If an interrupt occursduringthistime,itgenerallyremainspendingandwillbe checked by the processor after the processor has re-enabled interrupts.

Thus if an interrupt occurs when a user program is executing, then interrupts are disabled immediately.Aftertheinterrupt-handlerroutinecompletes,interruptsarere-enabledbefore resumingtheuser programand the processor checkstoseeifadditional interrupts have occurred. This approach is simple, as interrupts are handled in strict sequential order

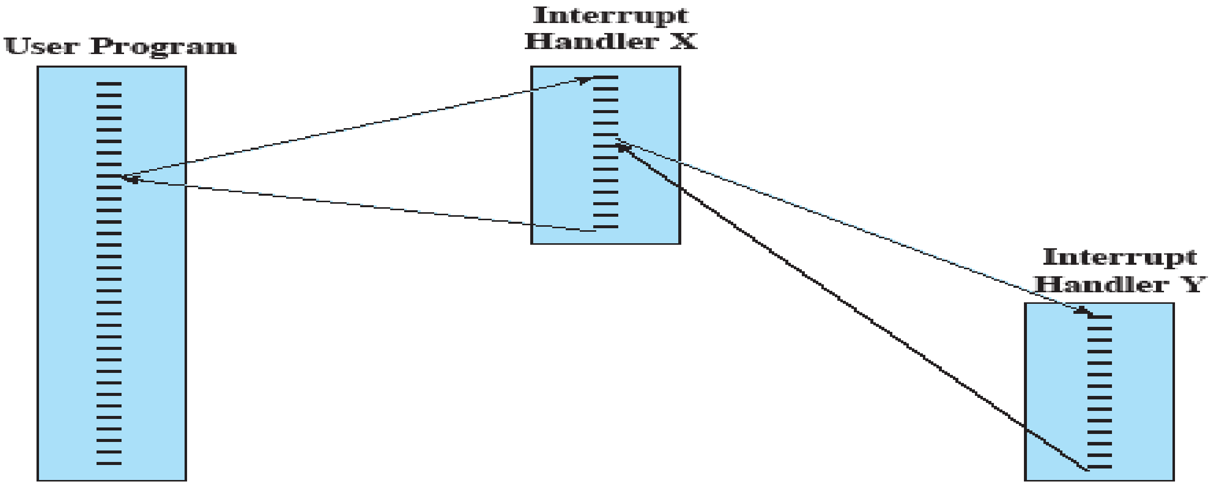


(a) Sequential interrupt processing

Thedrawbackofsequentialapproachisthatitdoesnottakeintoaccountrelativepriorityor time- critical needs.

b) NestedInterruptProcessing

Asecondapproachistodefineprioritiesforinterruptsandtoallowaninterruptof higher priority to cause a lower-priority interrupt handler to be interrupted.



(b) Nested interrupt processing

Asanexampleofthissecondapproach,considerasystemwiththreeI/Odevices:

- aprinter(priority 2),
- adisk(priority4),and
- acommunicationsline(priority5).

This figure illustrates a possible sequence.

1. Auserprogrambeginsat*t*=0.
2. At*t*=10,aprinterinterruptoccurs;

   userinformation is placedonthe control stackand executioncontinues atthe printer interrupt service routine (ISR).

3. Whilethisroutineisstillexecuting,at*t*=15acommunicationsinterruptoccurs.

   Because the communications line has higher priority than the printer, the interrupt request is honored.

4. TheprinterISRisinterrupted,itsstateispushedontothestack,andexecution continues at the communications ISR.

5. Whilethisroutineisexecuting,adiskinterruptoccurs(*t*=20).

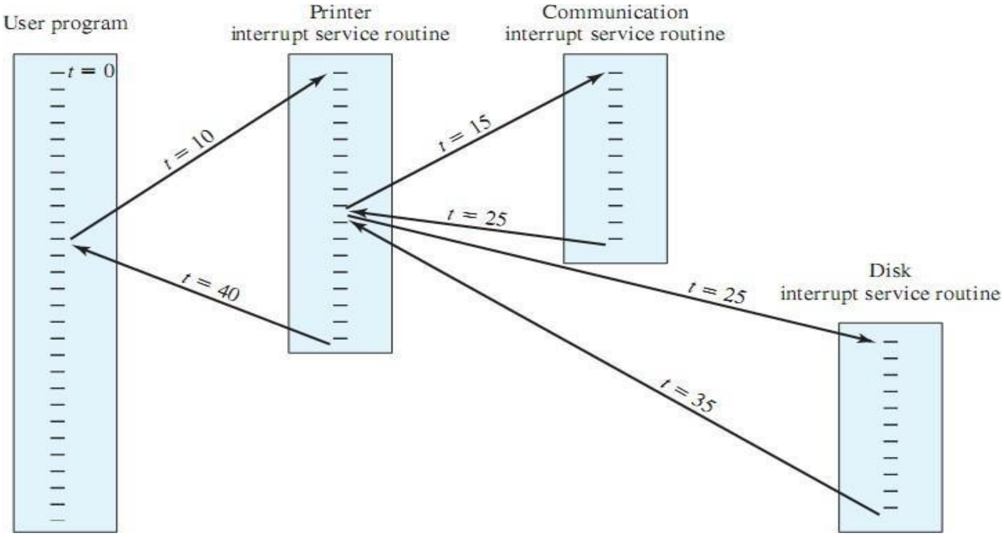
   Because this interrupt is of lower priority, it is simply held, and the communications ISR runs to completion.

6. WhenthecommunicationsISRiscomplete(*t*=25),thepreviousprocessorstateis restored, which is the execution of the printer ISR.

7. However,beforeevenasingleinstructioninthatroutinecanbeexecuted,theprocessor honors the higher-priority disk interrupt and transfers control to the disk ISR.

8. Onlywhenthatroutineiscomplete(*t*=35)istheprinterISRresumed.

9. Whenthatroutinecompletes(*t*=40),controlfinallyreturnstotheuserprogram.



MEMORYHIERARCHY.

MemoryHierarchy

➔Thememoryunitisanessential componentinanydigitalcomputersinceitisneeded for storing programs and data.

➔Not all accumulated information is needed by the CPU at the same time. Therefore, it is more economical to use low-cost storage devices to serve as a backup for storing the information that is not currently used by CPU.

➔Computer Memory Hierarchy is a pyramid structure that is commonly used to illustrate the significant differences among memory types.

➔ThememoryunitthatdirectlycommunicateswithCPUiscalledthemainmemory. Devices that provide backup storage is called auxiliary memory.

➔Thememoryhierarchysystemconsistsofallstoragedevicesemployedinacomputer system from the slow by high-capacity auxiliary memory to a relatively faster main memory, to an even smaller and faster cache memory

Performance

Access time —Time between presenting the address and getting the valid data

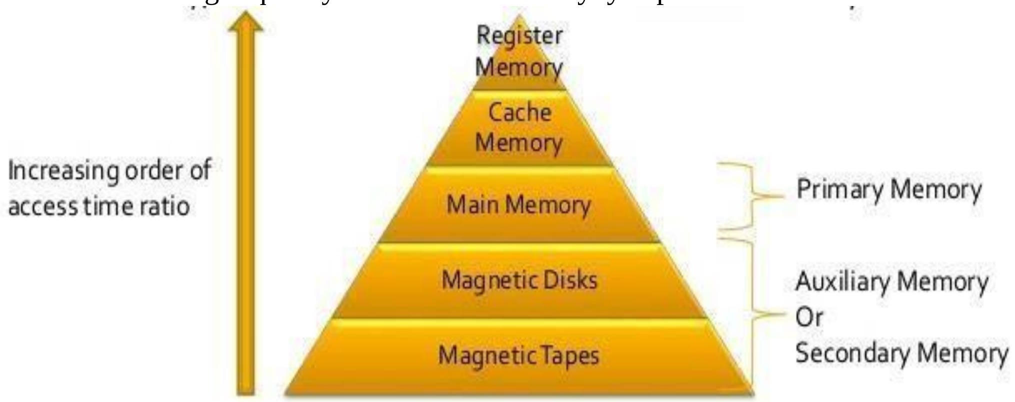
MemoryCycletime —Timemayberequiredforthememoryto“recover”beforenext access

—Cycletimeisaccess+recovery

Transfer Rate —Rate at which data can be moved

Goingdownthehierarchy

– Decreasingcostper bit
– Increasingcapacity
– Increasingaccess time
– Decreasingfrequencyofaccesstothememorybytheprocessor



<u>MainMemory</u>

➔MostofthemainmemoryinageneralpurposecomputerismadeupofRAM integrated circuits chips, but a portion of the memory may be constructed with ROM chips

1. RAM–RandomAccessmemory
2. ROM–ReadOnlymemory

<u>RAM</u>

ARAMchipisbettersuitedforcommunicationwiththeCPUifithasoneor more controlinputsthatselectthechipwhenneeded.

Keyfeatures

RAMispackagedasachip.
Basicstorageunitisacell(onebitpercell). Multiple RAM chips form a memory.

StaticRAM (SRAM)

Eachcellstoresbitwithasix-transistorcircuit.
Retains value indefinitely, as long as it is kept powered.
Relativelyinsensitivetodisturbancessuchaselectricalnoise.
Faster and more expensive than DRAM.

DynamicRAM(DRAM)

Eachcellstoresbitwithacapacitorandtransistor. Value must be refreshed every 10-100 ms.

Sensitive todisturbances.
SlowerandcheaperthanSRAM.

ROM

➔ROMisused for storingprograms that are**PERMENTLY** resident in the computer and for tables of constants that do not change in value once the production of the computeris completed.

➔The ROM portionof main memory is needed for storing an initial program called*bootstraploader,* which is to start the computer software operating when power is turned off.

➔DataisprogrammedintothechipusinganexternalROMprogrammer The programmed chip is used as a component into the circuit ThecircuitdoesnʼtchangethecontentoftheROM

AuxiliaryMemory

➔Auxiliarymemory,alsoknownasauxiliarystorage,secondarystorage,secondarymemory or external memory, is a non-volatile memory (does not lose stored data when thedevice is powered down) that is not directly accessible by the CPU, because it is not accessedvia the input/output channels (it is an external device).

➔Someexamplesofauxiliarymemorywouldbedisks,externalharddrives,USBdrives, etc.

CacheMemory

➔Cachememory,alsocalledCPUmemory,israndomaccessmemory(RAM)thatacomputer microprocessorcan access more quickly than it can access regular RAM. Thismemoryistypically integrated directly with the CPU chip or placed on a separate chip that hasa separatebusinterconnectwiththeCPU.

➔Thebasicpurposeof cache memory is to store program instructions that are frequentlyre-referenced by software during operation. Fast access to these instructions increases theoverall speedofthesoftwareprogram.

➔As the microprocessor processes data, it looks first in the cache memory; if it finds the instructionsthere(fromapreviousreadingofdata),itdoesnothavetodoamoretime-consuming reading of data from larger memory or other data storage devices.

TertiaryStorage

➔Tertiary Storage, also known as tertiary memory, consists of anywhere from one to several storage drives. It is a comprehensive computer storage system that is usually very slow, so it is usually used to archive data that is not accessed frequently. A computer can accesstertiarystorage without being told to do so, which is unlike off-line storage.

➔Thistypeofcomputerstoragedeviceisnotaspopularastheothertwostoragedevicetypes. Its main use is for storing data at a very large-scale. This includes optical jukeboxes and tape libraries. Tertiary storage devices require a database to organize the datathatarestoredinthem,and the computer needs to go through the database to access those data.

Memoryhierarchyisjustliketherealworldsituationwhere-

1. Atrainfareischeaper anditcancarryalotpeopleatatimebutittakeslongtime
2. Theairfareofprofessionalflightsismorethanthetrain,itcancarrylessernumberofpeople but it is much faster than the train
3. Theairfareforpersonaljetisfurtherhigh,itcancarryfurtherlessernumberofpeoplebutitis fastest of the three.

So, depending upon the price and the urgency to reach destination, you will use combinationof these in different situations.

Thememoryhierarchyisexactlythesame.Here,thesituationis-

1. Weneedalot ofmemorywhichischeapandcouldbeslow(secondarymemory,HardDisk)
2. We also need some memory which could be smaller than secondary memory but should be faster than it (primary memory, RAM)
3. Wealsoneedanotherkindofmemorywhichcouldbesmallerthantheprimary memory but it should be much faster than it (cache memory).

That'swhyweneedmemory hierarchy.

CACHEMEMORY

Concept

Smallamountoffastestmemory.

SitsbetweennormalmainmemoryandCPU. May be located on CPU chip or module.

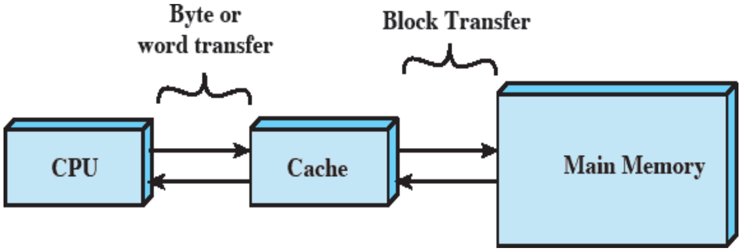
CachePrinciples

Containscopyofaportionofmainmemory

Processor first checks cache

If desired data item not found, relevant block of memory read into cache

Becauseoflocalityofreference,itislikelythatfuturememoryreferencesarein thatblock.



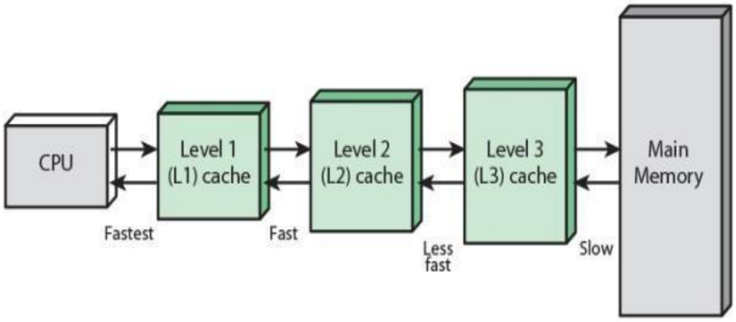
CacheOperation

CPUrequestscontentsofmemorylocation. Check cache for this data.

Ifpresent,getfromcache(fast).

Ifnotpresent,readrequiredblockfrommainmemorytocache. Then deliver from cache to CPU.

Cacheincludestagstoidentifywhichblockofmainmemoryisineachcacheslot.



ThreeLevelCacheMemoryHierarchy

The L3 cache is usually built onto the motherboard between the main memory (RAM) and the L1 and L2 caches of the processor module.

Thisservesasanotherbridgetoparkinformationlikeprocessorcommandsandfrequently useddatainordertopreventbottlenecksresultingfromthefetchingofthesedata fromthemain memory.

In short, the L3cache of today is what the L2cache was before it got built-in within the processor module itself.

The CPU checks for information it needs from L1 to the L3 cache. If it does not findthis info in L1 it looks to L2 then to L3, the biggest yet slowest in the group.

The purpose oftheL3differs dependingonthedesignoftheCPU. In some casesthe L3 holds copies of instructions frequently used by multiple cores that share it.

MostmodernCPUshavebuilt-inL1andL2cachesper coreandshareasingleL3cache on the motherboard, while other designs have the L3 on the CPU die itself.

CacheMemoryStructure

➔Naddresslines=>2nwordsofmemory

➔CachestoresfixedlengthblocksofKwords

➔CacheviewsmemoryasanarrayofMblockswhereM=2n/K

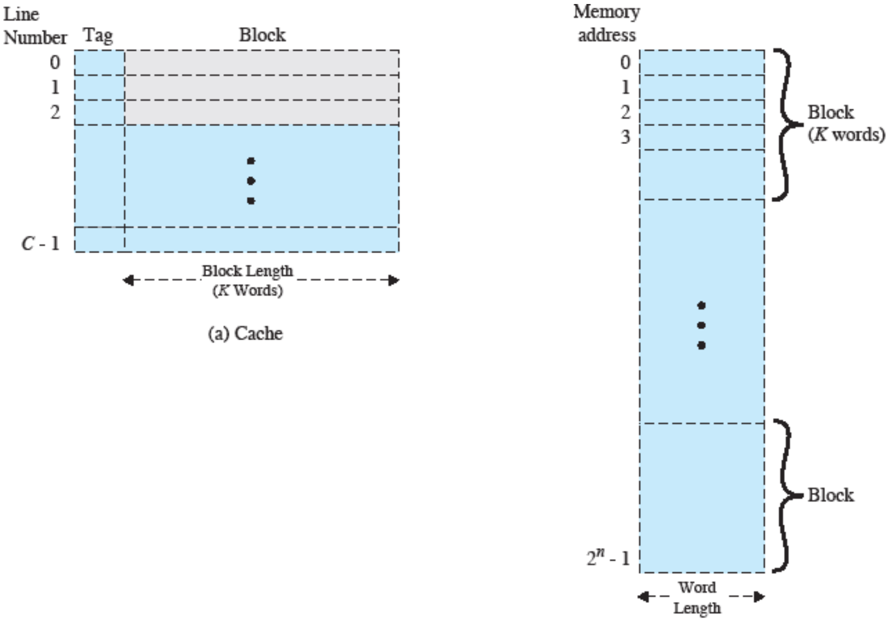
➔Ablockofmemoryincacheisreferredtoasaline.Kisthelinesize

➔CachesizeofCblockswhereC<M
(considerably)

➔Eachlineincludesatagthatidentifiestheblockbeing stored

➔Tagisusuallyupperportionofmemoryaddress

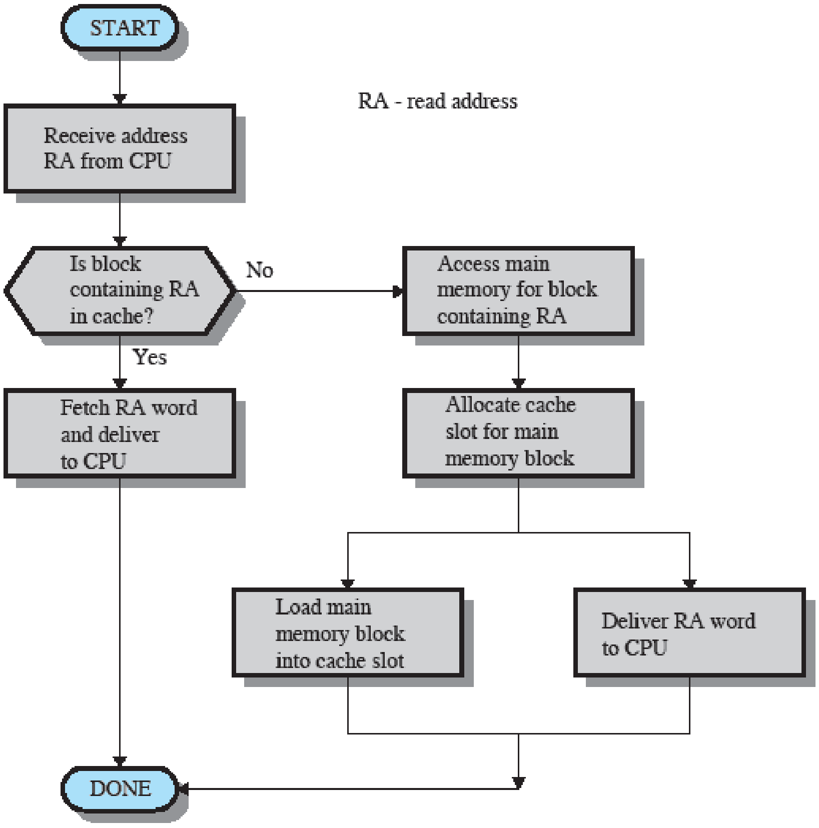
Asasimpleexample,supposethatwehavea6-bitaddressanda2-bittag.
The tag 01 refers to the block of locations with the following addresses: 010000, 010001, 010010,010011,010100,010101,010110,010111,011000,011001,011010,011011,011100, 011101,011110,and011111.



(a) Cache

(b) Main memory

CacheReadOperation


START
RA - read address
Receive address RA from CPU
Is block containing RA in cache?
No
Yes
Access main memory for block containing RA
Fetch RA word and deliver to CPU
Allocate cache slot for main memory block
Load main memory block into cache slot
Deliver RA word to CPU
DONE

The processor generates the address, RA, of a word tobe read. If the word is contained inthe cache,itisdeliveredtotheprocessor.Otherwise,theblockcontainingthatwordisloadedinto the cache and the word is delivered to the processor.

CacheDesign

ElementsofCacheDesign

- Addresses(logicalorphysical)
- Size
- MappingFunction(direct,associative,setassociative)
- ReplacementAlgorithm(LRU,LFU,FIFO,random)
- WritePolicy(writethrough,writeback,writeonce)
- LineSize
- NumberofCaches(howmanylevels,unifiedorsplit)

Cache size

Evensmallcacheshavesignificantimpactonperformance Block size

Theunitofdataexchangedbetweencacheandmainmemory

Larger block size yields more hits until probability of using newly fetched databecomes less than the probability of reusing data that have to be moved out of cache.

Mappingfunction

Determineswhichcachelocationtheblockwilloccupy

Replacement algorithm

Chooseswhichblocktoreplace

Least-recently-used(LRU)algorithm

Write policy

- Dictateswhenthememorywriteoperationtakesplace
- Canoccureverytimetheblockisupdated
- Canoccurwhentheblockisreplaced
  - Minimize write operations
  - Leavemainmemoryinanobsoletestate

DIRECTMEMORYACCESS(DMA)

➔Three techniques are possible for I/O operations: programmed I/O, interrupt-driven I/O, and direct memoryaccess(DMA).BeforediscussingDMA,webrieflydefinetheothertwotechniques;seeAppendix Cformoredetail.WhentheprocessorisexecutingaprogramandencountersaninstructionrelatingtoI/O, it executes that instruction by issuing a command to the appropriate I/O module.

➔In the case of **programmed I/O** , the I/O module performs the requested action and then sets the appropriate bits in the I/O status register but takes no further action to alert the processor. In particular, it does not interrupt the processor. Thus, after the I/O instruction is invoked, the processor must take some active role in determining when the I/O instruction is completed. For this purpose, the processor periodically checks the status of the I/O module until it finds that the
Operationiscomplete.

➔With programmed I/O, the processor has to wait a long time for the I/O module of concern to be ready foreitherreceptionortransmissionofmoredata.Theprocessor,whilewaiting,mustrepeatedlyinterrogate the status of the I/O module.

➔As a result, the performance level of the entire system is severely degraded. An alternative, known as **interrupt-drivenI/O**,isfortheprocessortoissueanI/Ocommandtoamoduleandthengoontodosome other useful work.

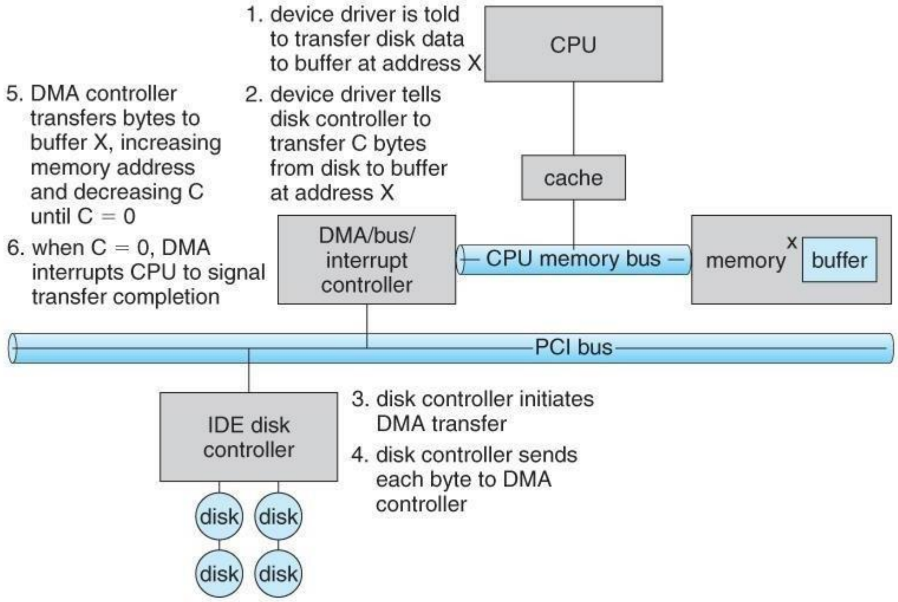
➔The I/O module will then interrupt the processor to request service when it is ready to exchange data with the processor. The processor then executes the data transfer, as before, and then resumes its former processing.

➔When large volumes of data are to be moved, a more efficient technique is required: **direct memory access (DMA).** The DMA function can be performed by a separate module on the system bus or it can be incorporatedintoanI/Omodule.Ineithercase,thetechniqueworksasfollows.Whentheprocessorwishes toreadorwriteablock ofdata,it issuesacommandto the DMAmodule, bysendingtotheDMA module the following information:

- Whetherareadorwriteisrequested
- TheaddressoftheI/Odeviceinvolved
- Thestartinglocationinmemorytoreaddatafromorwritedatato
- Thenumberofwordstobereadorwritten

➔The processor then continues with other work. It has delegated this I/O operation to the DMA module, andthatmodulewilltakecareofit.TheDMAmoduletransferstheentireblockofdata,onewordatatime, directlytoorfrommemorywithoutgoingthroughtheprocessor.Whenthetransferiscomplete,the DMA modulesendsaninterruptsignaltotheprocessor.Thus,theprocessorisinvolvedonlyatthebeginningand end of the transfer.

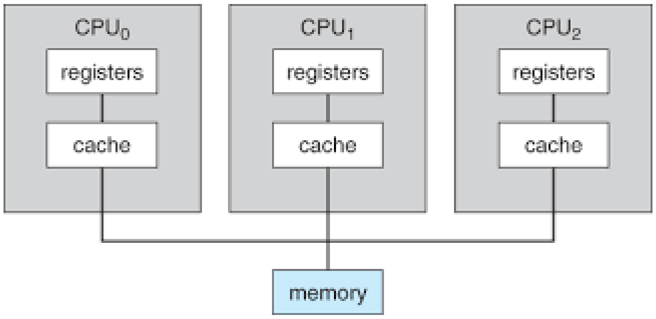
TheFigurebelowillustratestheDMAprocess.


1. device driver is told to transfer disk data to buffer at address X
2. device driver tells disk controller to transfer C bytes from disk to buffer at address X
3. disk controller initiates DMA transfer
4. disk controller sends each byte to DMA controller
5. DMA controller transfers bytes to buffer X, increasing memory address and decreasing C until C = 0
6. when C = 0, DMA interrupts CPU to signal transfer completion
CPU
cache
DMA/bus/ interrupt controller
CPU memory bus
memory
x
buffer
PCI bus
IDE disk controller
disk
disk
disk
disk

MULTIPROCESSORANDMULTICOREORGANIZATION

MULTIPROCESSING

➔Multiprocessing is the use of two or more central processing units (CPUs) within a single computer system. The term also refers to the ability of a system to support more than one processorand/or the ability to allocate tasks between them.



➔Therearemultipleprocessors,eachofwhichcontainsitsowncontrolunit,arithmeticlogicunit, and registers. Each processor hasaccesstoa sharedmainmemory andthe I/Odevices through some form of interconnection mechanism; a shared bus is a common facility. The processors can communicate with eachotherthroughmemory(messagesandstatusinformationleftinsharedaddressspaces).Itmay alsobe possible for processors to exchange signals directly. The memory is often organized so that multiple simultaneous accesses to separate blocks of memory are possible.

➔Multiprocessorsystemshavethreemainadvantages.

1. Increasedthroughput.
2. Economyofscale.
3. Increasedreliability.

The mostcommonmultiple-processor systemsnow use **symmetric multiprocessing** (SMP), inwhich each processor runs an identical copy of the operating system, and these copiescommunicate with one another as needed.

Some systems use **asymmetric multiprocessing**, in which each processor is assignedaspecifictask.Amasterprocessorcontrolsthesystem;theotherprocessors either look to the master forinstruction orhave predefined tasks.This schemedefinesa master-slaverelationship.Themasterprocessorschedulesandallocatesworktotheslaveprocessors.

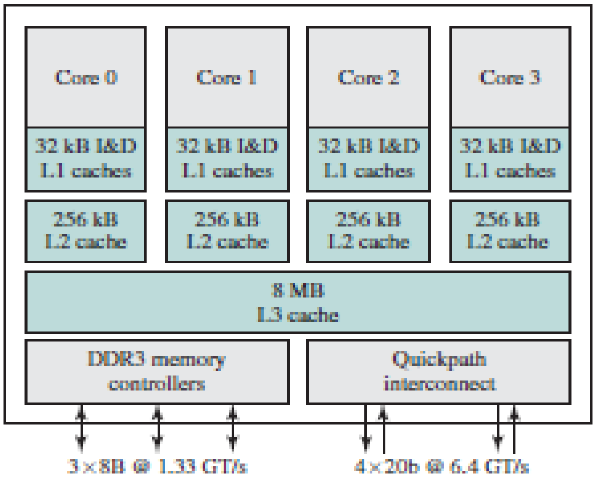
An SMP organization has a number of potential advantages over a uni-processor organization, including the following:

- **Performance:**Iftheworktobedonebyacomputercanbeorganized so thatsome portions of the work can be done in parallel, then a system withmultipleprocessorswillyieldgreaterperformancethanonewith a single processor of the same type.
- **Availability:**Inasymmetricmultiprocessor,becauseallprocessorscan perform the same functions, the failure of a single processor does nothaltthemachine.Instead,thesystemcancontinuetofunction at reduced performance.
- **Incremental growth:** A user canenhance the performance of a system by adding an additional processor.
- **Scaling:** Vendors can offer a range of products with different price and performancecharacteristicsbasedonthenumberofprocessorsconfigured in the system.

➔MULTICORECOMPUTERS

➔A **multicore** computer, also known as a **chip multiprocessor,** combines two or more processors (called cores) on a single piece of silicon (called a die). Typically, each core consists of all of the components of an independent processor, such as registers, ALU, pipeline hardware, and control unit, plus L1 instruction and data caches. In addition to the multiple cores, contemporary multicore chips also include L2 cache and, in some cases, L3 cache. The motivation for the development of multicore computers can be summed up as follows.

➔For decades, microprocessor systems have experienced a steady, usually exponential, increase inperformance.Thisispartlyduetohardwaretrends,suchasanincreaseinclockfrequencyandtheability to put cache memory closer to the processor because of the increasing miniaturization of microcomputer components. Performance has also been improved by the increased complexity of processor design to exploit parallelism in instruction execution and memory access.



➔Inbrief,designershavecomeupagainstpracticallimitsintheabilitytoachievegreaterperformanceby meansofmorecomplexprocessors.Designershavefoundthatthebestwaytoimproveperformancetotake advantageofadvancesinhardwareistoputmultipleprocessorsandasubstantialamountofcachememory on a single chip

➔An example of a multicore system is the Intel Core i7, which includes four x86processors, each witha dedicated L2 cache, and with a shared L3 cache. One mechanism Intel uses to make its caches more effective is prefetching, in which the hardware examines memory access patterns and attempts to fill the caches speculatively with data that' s likely to be requested soon.

COMPONENTSOFOPERATINGSYSTEM.

Thereareeightmajoroperatingsystemcomponents.

They are :
□Process management
□Main-memorymanagement □
File management
□ I/O-systemmanagement
□ Secondary-storagemanagement
□ Networking
□ Protectionsystem
□ Command-interpretersystem

(i) ProcessManagement

□Aprocesscanbethoughtofasaprograminexecution.Abatchjobisaprocess. A time shared user program is a process.

□Aprocessneedscertainresources-includingCPUtime,memory,files, and I/O devices-to accomplish its task.

□A program by itself is not a process; a program is a passive entity, such as the contents of a file stored on disk, whereas a process is an activeentity,with a program counter specifying the next instruction to execute.

□Aprocessistheunit ofworkinasystem.

□Theoperatingsystemisresponsibleforthe following activities in connectionwith process management:

□Creatinganddeletingbothuserandsystemprocesses □
Suspending and resuming processes
□Providingmechanismsforprocesssynchronization □
Providingmechanismsfor process communication □
Providing mechanisms for deadlock handling

(ii) Main–MemoryManagement

□Main memory isa large array of wordsor bytes, ranging in size from hundreds of thousands to billions. Each word or byte has its own address.

□MainmemoryisarepositoryofquicklyaccessibledatasharedbytheCPUandI/O devices.

□ToimprovebooththeutilizationoftheCPUandthespeedofthecomputer'sresponseto its users, we must keep several programs in memory.

□The operating system is responsible for the followingactivitiesinconnection with memory management:

□Keeping track of which parts of memory are currently being usedandby whom.

□Decidingwhich processesaretobeloadedintomemorywhen memory space becomes available

□Allocating anddeallocating memoryspace asneeded.

(iii) FileManagement

□Filemanagementisoneofthemostvisiblecomponentsofanoperating system.

□The operating systemis responsible for the following activities in connection with file management:

□Creatinganddeletingfiles□
Creatinganddeletingdirectories
□Supportingprimitivesformanipulatingfilesanddirectories □
Mapping files onto secondary storage
□Backingupfilesonstable(nonvolatile)storagemedia

(iv) I/OSystemmanagement

□Oneofthepurposesofanoperatingsystemistohidethe peculiaritiesofspecifichardwaredevicesfromtheuser. Thisisdoneusing theI/O subsystem.

□TheI/Osubsystemconsistsof
□A memory-management componentthat includes buffering, caching, and spooling
□Ageneraldevice-driverinterface□
Driversforspecifichardwaredevices

(v) Secondarystoragemanagement

□Because main memory is too small to accommodate all data and programs,andbecausethedatathatitholdsarelostwhenpowerislst thecomputersystemmustprovidesecondarystoragetobackup main memory.

□Theoperatingsystemisresponsibleforthefollowingactivitiesin connection with disk management:

□Free-spacemanagement
□Storageallocation□Disk
scheduling

(vi) Networking

□Adistributedsystemisacollectionofprocessorsthatdonot share memory,peripheraldevices,oraclock.

□Instead,eachprocessorhasitsownlocalmemoryandclock,and the processors communicate with one another through various communicationlines,suchashigh-speedbusesornetworks.

□Theprocessorsinthesystem areconnectedthroughacommunication network, which can be configured in a number of different ways.

(vii) ProtectionSystem

□Variousprocessesmustbeprotectedfromoneanother'sactivities.For that purpose, mechanisms ensure that the files, memory segments, CPU, and other resources can be operated on by only those processes that have gainedproperauthorizationfromtheoperating system.

□Protectionisanymechanismforcontrollingtheaccessof programs,processes,oruserstotheresourcesdefinedbyacomputer system.

□Protectioncanimprovereliabilitybydetectinglatenterrorsatthe interfaces between component subsystems.

(viii) Command-InterpreterSystem

□One of the most importantsystemsprogramsforanoperatingsystemis the command interpreter.

□Itistheinterfacebetweentheuserandtheoperatingsystem.

□Someoperatingsystemsincludethecommandinterpreterinthe kernel. Other operating systems, such as MS-DOS and UNIX, treat the commandinterpreterasaspecialprogramthatisrunningwhenajobis initiated, or when a user first logs on (on time-sharing systems).

Manycommandsaregiventotheoperatingsystem bycontrol statements.

Whena newjob is startedin a batchsystem, or when auserlogs on to a time-shared system, a program that reads and interprets control statements is executed automatically.

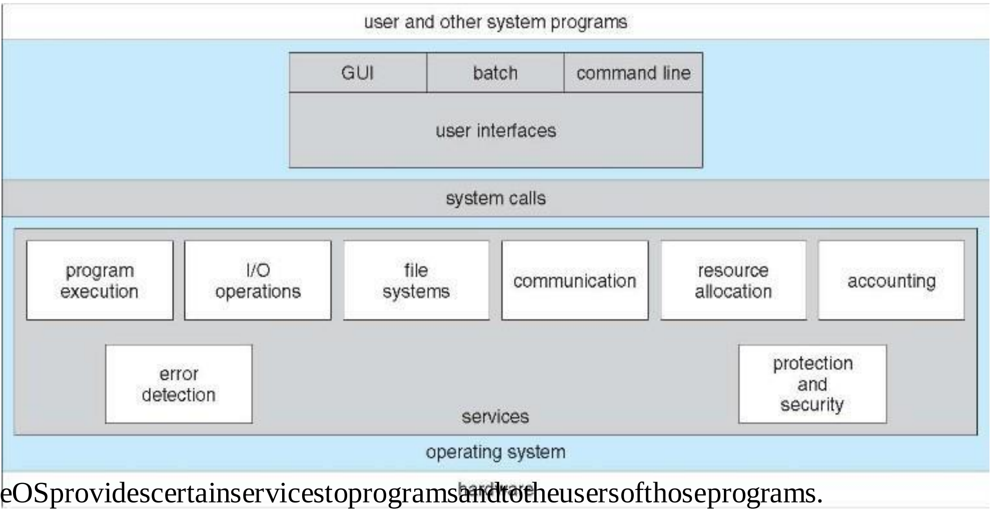
This program issometimescalled thecontrol-cardinterpreter or the Command-line interpreter, and is often known as the shell.

SERVICESOFOPERATING SYSTEM

➔Anoperatingsystemprovidesservicestoprogramsandtotheusersofthoseprograms. Itprovidedbyoneenvironmentfortheexecutionofprograms.

➔Theservicesprovidedbyoneoperatingsystemisdifficultthanotheroperatingsystem. Operating system makes the programming task easier. The common service providedbythe operating system is listed below.

1. Programexecution
2. I/Ooperation
3. Filesystemmanipulation
4. Communications
5. Errordetection



TheOSprovidescertainservicestoprogramsandtotheusersofthoseprograms.

1. **Programexecution:**Thesystemmustbeabletoloadaprogramintomemoryandtorun that program. The program must be abletoendits execution, either normally or abnormally (indicating error).
2. **I/Ooperations:**ArunningprogrammayrequireI/O.ThisI/OmayinvolveafileoranI/Odevice.
3. **File-system manipulation:**Theprogramneedstoread,write,create,deletefiles.
4. **Communications:**Inmanycircumstances,oneprocessneedstoexchange Informationwithanotherprocess.Suchcommunicationcanoccurintwomajorways.Thefirst takes place between processes that are executing onthesamecomputer;thesecondtakesplacebetweenprocessesthatareexecutingondifferent computer systems that are tied together by a computer network.
5. **Errordetection:**Theoperatingsystemconstantlyneedstobeawareofpossibleerrors.Errors may occur in the CPU and memory hardware (such as amemoryerrororapowerfailure),

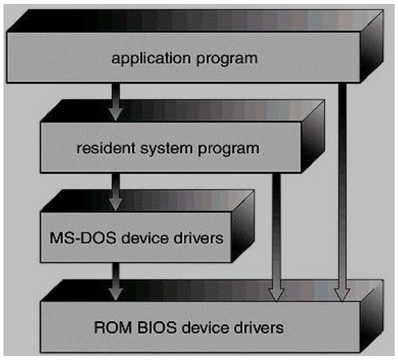
inI/O devices (suchasa parity error on tape,aconnectionfailureonanetwork,orlackofpaper intheprinter),andintheuserprogram(suchasanarithmeticoverflow,anattempttoaccessanillegal memorylocation,oratoo-greatuseofCPUtime).Foreachtypeoferror,theoperatingsystemshould take the appropriate actionto ensure correct and consistent computing.

6. **Resourceallocation:**DifferenttypesofresourcesaremanagedbytheOs.Whentherearemultiple users or multiple jobs running at the same time, resources must be allocated to each of them.
7. **Accounting:**Wewanttokeeptrackofwhichusersusehowmanyandwhichkindsofcomputer resources. This record keeping may be used foraccountingorsimplyfor accumulating usage statistics.
8. **Protection:**The ownersofinformationstoredinamultiusercomputersystem may want to control use of that information. Security of the system is also important.

OPERATINGSYSTEMSTRUCTURES

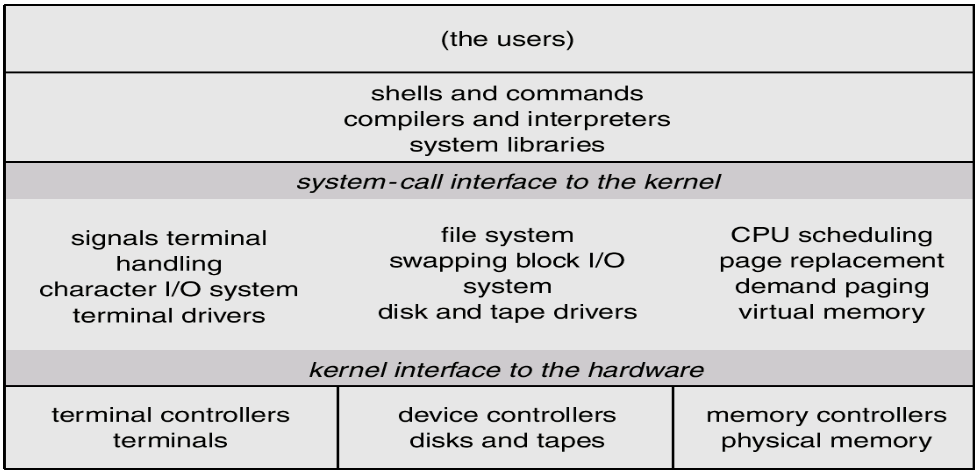
SIMPLE STRUCTURE:

➔In MS-DOS, application programs are able to access the basic I/O routines to write directly to the display and disk drives. Such freedom leaves MS-DOS vulnerable to errant (or malicious) programs, causing entire system to crash when user programs fail.



MS-DOSLAYERSTRUCTURE:

➔ UNIX operating system. It consists of two separable parts, the kernel and the system programs. The kernel is further separated into a series of interfaces and device drivers.Wecanview the traditional UNIX operating system as being layered. Everything belowthesystemcallinterface and above the physical hardware is the kernel.



The kernel provides the file system, CPU scheduling, memory management, and other operating system functions through system calls. There is number of functionality tobecombined into one level. This monolithic structure was difficult to implement andmaintain.

LAYEREDAPPROACH:

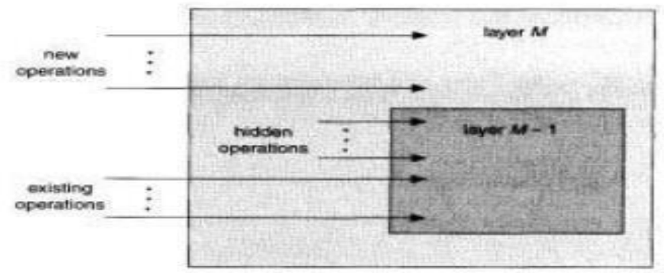
➔The operating system is broken into a number of layers (levels). The bottom layer (layer 0) is the hardware; the highest (layer M) is the user interface.

➔The main advantage of the layered approach is simplicity of construction anddebugging.Thelayersareselectedsothateachusesfunctions(operations)andservices ofonly lower-level layers. This approach simplifies debugging and system verification. The first layer can be debugged without any concern for the rest of the system, because, by definition, it uses only the basic hardware to implement its functions.

➔Once the first layer is debugged, its correct functioning can be assumed while the second layer is debugged, and so on. If an error is found during the debugging of a particularlayer, the error must be on that layer, because the layers below it are already debugged.Eachlayerhidesthe existence of certain data structures, operations, and hardware from higher-level layers.

➔Themajordifficultywiththelayeredapproachinvolvesappropriatelydefining thevariouslayersasalayercanuseonlylower-levellayers.Anotherproblemwith layered implementations is they tend to be less efficient than other types. Each layer adds overheadtothe system call; the net result is a system call that takes longer than a non- layered system.

ExampleofLayeredApproach

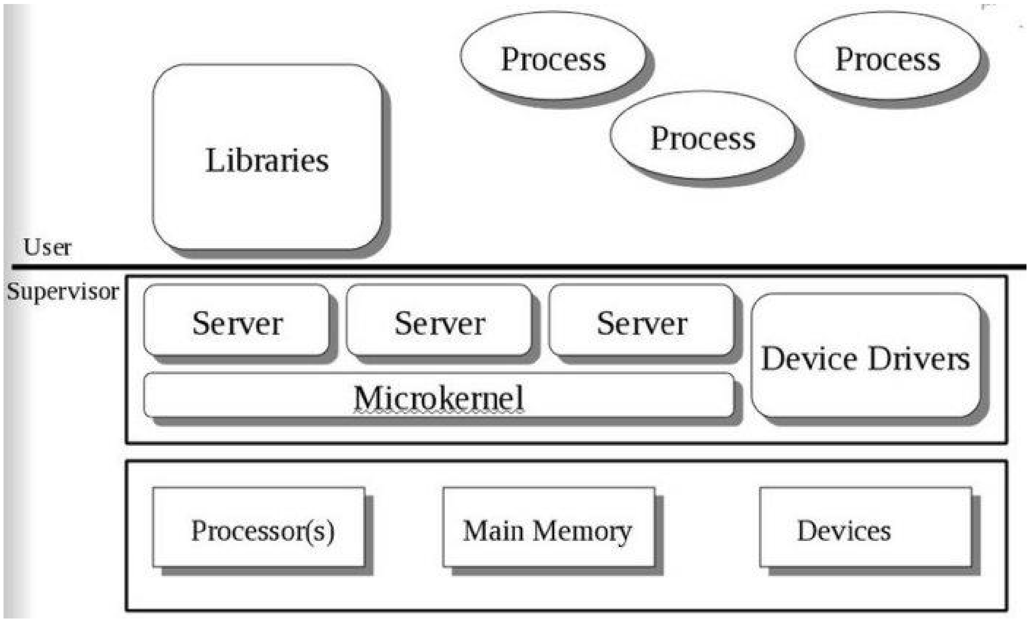


MICROKERNELAPPROACH:

➔In the mid-1980s, researchers at Carnegie Mellon University developed an operating system called Mach that modularized the kernel using the microkernel approach. Microkernel’s provide minimal process and memory management, in addition to a communication facility.

➔The main function of the micro kernel is to provide a communication facility betweenthe clientprogramandthevarious servicesrunninginuserspace. Onebenefit ofthemicrokernel approach is ease of extending the operating system. All new services are addedtouserspaceandconsequentlydo not require modification of the kernel. The microkernel also provides more security and reliability, since most services are running as user, rather than kernel-processes.

➔Microkernel’s can suffer from decreased performance due toincreased system function overhead.



MODULES:

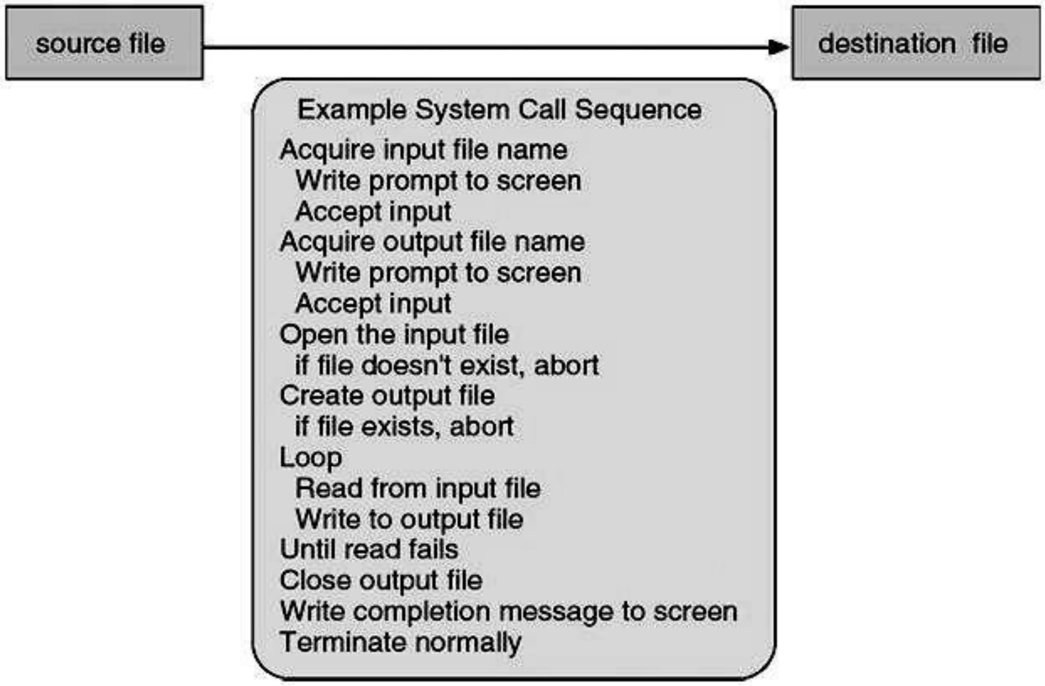
➔The current methodology for operating-system design involves using object-oriented programmingtechniquestocreateamodularkernel.Here,thekernelhasasetofcorecomponents and links in additional services either during boot time or during run time. Such a strategyusesdynamicallyloadablemodules.

➔Such a design allows the kernel to provide core services yet also allows certain features to be implemented dynamically.

SYSTEMCALLS

➔Asystemcallisarequestthataprogrammakestothekernelthroughasoftware interrupt.System calls provide theinterface between aprocess and the operating system.

➔These calls are generally available as assembly-language instructions. Certain systems allow system calls to be made directly from a high-level language program, in which case the calls normally resemble predefined function or subroutine calls.



```
Example System Call Sequence
Acquire input file name
  Write prompt to screen
  Accept input
Acquire output file name
  Write prompt to screen
  Accept input
Open the input file
  if file doesn't exist, abort
Create output file
  if file exists, abort
Loop
  Read from input file
  Write to output file
Until read fails
Close output file
Write completion message to screen
Terminate normally
```

TYPESOFSYSTEMCALLS:

Traditionally,SystemCallscanbecategorizedinsixgroups,whichare:ProcessControl, File Management, DeviceManagement,InformationMaintenance, Communications and Protection.

PROCESSCONTROL

- Arunningprogramneedstobeabletostopexecutioneithernormallyorabnormally.
- When execution is stopped abnormally, often a dump of memory is taken and can be examinedwith a debugger.

Followingarefunctionsofprocesscontrol:

End, abort
Load,execute
Createprocess,terminateprocess
Getprocessattributes,setprocessattributes
Wait for time
Wait event, signal event
Allocateandfreememory

FILEMANAGEMENT

- We first need to be able to create and delete files. Either system call requires the name of the file and perhaps some of the file's attributes.
- Once the file is created, we need to open it and to use it. We may also read, write, or reposition. Finally, we need to close the file, indicating that we are no longer using it.
- We may need these same sets of operations for directories if we have a directory structure for organizing files in the file system.
- In addition, for either files or directories, we need to be able to determine the values of various attributes and perhaps to reset them if necessary. File attributes include the file name, a file type, protection codes, accounting information, and so on

Functions:

Create,deletefile
Open, close
Read,write,reposition
Getfileattributes,setfileattributes

DEVICEMANAGEMENT

- A process may need several resources to execute - main memory, disk drives, access to files, and so on. If the resources are available, they can be granted, and control can be returned to the user process. Otherwise, the process will have to wait until sufficient resources are available.
- ThevariousresourcescontrolledbytheOScanbethoughtofasdevices.Someofthesedevicesare physicaldevices(for example,tapes),whileotherscanbethought ofas abstractor virtual devices (for example, files).
- Once the device has been requested (and allocated to us), we can read, write, and (possibly) reposition the device, just as we can with files.
- In fact, the similarity between I/O devices and files is so great that many OSs, including UNIX, merge the two into a combined file-device structure.
- Asetofsystemcallsisusedonfilesanddevices.Sometimes,1/0devicesareidentifiedbyspecial file names, directory placement, or file attributes.

Functions:

Requestdevice,releasedevice
Read, write, reposition
Getdeviceattributes,setdeviceattributes Logically
attach or detach devices

INFORMATIONMAINTENANCE

- Many system calls exist simply for the purpose of transferring information between the user program and the OS. For example, most systems have a system call to return the current time and date.
- Other system calls may return information about the system, such as the number of current users, the version number of the OS, the amount of free memory or disk space, and so on.
- In addition, the OS keeps information about all its processes, and system calls are used to access this information. Generally, calls are also used to reset the process information.

Functions:

Gettimeordate,settimeordate
Getsystemdata,setsystemdata
Getprocess,file,ordeviceattributes Set
process,file, or deviceattributes

COMMUNICATIONS

- Therearetwocommonmodelsofinterprocesscommunication:themessage-passingmodelandthe shared-memory model. In the message-passing model, the communicatingprocesses exchange messages with one another to transfer information.
- In the shared-memory model, processes use shared memory creates and shared memory attaches system calls to create and gain access to regions of memory owned by other processes.
- Recallthat,normally,theOStriestopreventoneprocessfromaccessinganotherprocess'smemory. Sharedmemoryrequiresthattwoormoreprocessesagreetoremovethisrestriction. Theycanthen exchange information by reading and writing data in the shared areas.
- Message passing is useful for exchanging smaller amounts of data, because no conflicts need be avoided. It is also easier to implement than is shared memory for intercomputer communication.
- Shared memory allows maximum speed and convenience of communication, since it can be done atmemoryspeedswhenit takesplacewithina computer.Problemsexist,however,intheareasof protection and synchronization between the processes sharing memory.

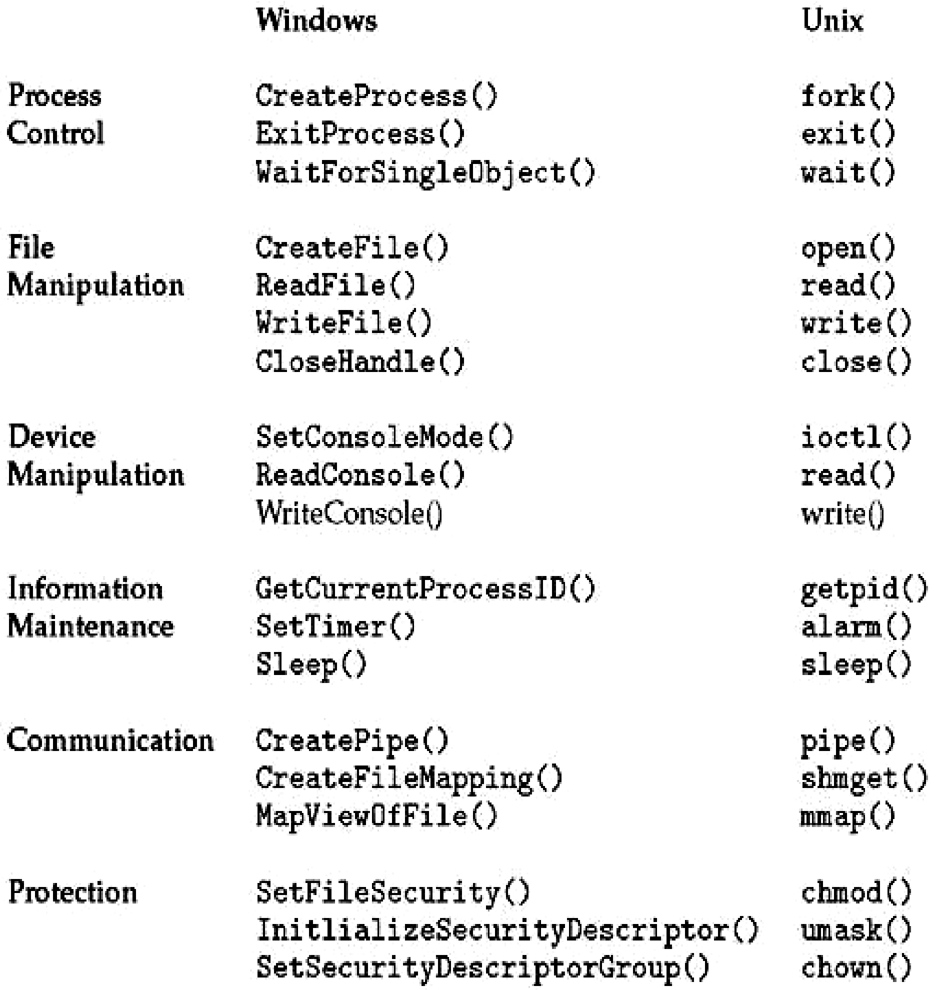
  Functions:

Create,deletecommunicationconnection
Send, receive messages
Transfer status information
Attachordetachremotedevices

PROTECTION

GetFileSecurity,SetFileSecurity

GetSecurityGroup,SetSecurityGroup



	Windows	Unix

Process Control	CreateProcess()
	ExitProcess()
	WaitForSingleObject()
		fork()
		exit()
		wait()

File Manipulation	CreateFile()
	ReadFile()
	WriteFile()
	CloseHandle()
		open()
		read()
		write()
		close()

Device Manipulation	SetConsoleMode()
	ReadConsole()
	WriteConsole()
		ioctl()
		read()
		write()

Information Maintenance	GetCurrentProcessID()
	SetTimer()
	Sleep()
		getpid()
		alarm()
		sleep()

Communication	CreatePipe()
	CreateFileMapping()
	MapViewOfFile()
		pipe()
		shmget()
		mmap()

Protection	SetFileSecurity()
	InitlializeSecurityDescriptor()
	SetSecurityDescriptorGroup()
		chmod()
		umask()
		chown()

SYSTEMPROGRAMS

➔Systemprogramsprovideaconvenientenvironmentforprogramdevelopmentandexecution. They can be divided into several categories:

1. **Filemanagement:**Theseprogramscreate,delete,copy,rename,print, dump, list, and generally manipulate files and directories.
2. **Statusinformation:**Thestatussuchasdate,time,amountofavailablememoryor diskspace, number of users or similar status information.
3. **File modification:** Several text editors may be available to create and modify the content of files stored on disk or tape.
4. **Programming-languagesupport:**Compilers,assemblers,andinterpretersforcommon programming languages are often provided to the user with the operating system.
5. **Programloadingandexecution:**Thesystemmayprovideabsoluteloaders,relocatableloaders,

linkageeditors,andoverlayloaders.

6. **Communications:** These programs provide the mechanism for creatingvirtual connections among processes, users, and different computer systems. (email, FTP, Remote log in)
7. **Application programs:** Programs that are useful to solve common problems, or to perform common operations.

Eg.Webbrowsers,databasesystems.

GENERATIONANDSYSTEMBOOT.

Operating-SystemGeneration

➔ It is possible to design, code, and implement an operating system specifically for one machine atonesite.Morecommonly,however,operatingsystemsaredesignedtorunonanyofaclassofmachines at a variety of sites with a variety of peripheral configurations. The system must then be configured or generatedforeachspecificcomputersite,aprocesssometimesknownassystemgeneration(SYSGEN).

➔TheoperatingsystemisnormallydistributedondiskorCD-ROM.Togenerateasystem,weuse aspecial program.TheSYSGENprogram readsfrom agivenfile, or askstheoperatorofthesystem for informationconcerningthespecificconfigurationofthehardwaresystem,orprobesthehardwaredirectly to determine what components are there. The following kinds of information must be determined.

➔What CPUistobeused?

What options(extended instructionsets,floating-pointarithmetic,andsoon)are installed? FormultipleCPUsystems,eachCPUmustbedescribed.

➔Howmuchmemoryisavailable?

Some systems will determine this value themselves by referencing memory location after memory location until an "illegal address" fault is generated. This procedure defines the finallegal address and hence the amount of available memory.

➔What devicesareavailable?

The system will need to know how to address each device (the device number), the device interrupt number, the device's type and model, and any special device characteristics.

➔What operating-system options are desired, or what parameter values are to be used? These options or valuesmightincludehowmanybuffers ofwhichsizesshould be used,whattype of CPU-scheduling algorithm is desired, what the maximum number of processes to be supportedis,andsoon. Oncethisinformationisdetermined,it can be usedinseveral ways.

SystemBoot

➔Afteranoperatingsystemisgenerated,itmustbemadeavailableforusebythe hardware. But how does the hardware know where the kernel is or how to load that kernel? Theprocedureofstartingacomputerbyloadingthekernelisknownas booting thesystem.

➔Onmostcomputersystems,asmallpieceofcodeknownasthe**bootstrapprogram**or **bootstrap loader** locates the kernel, loads it into main memory, and starts its execution. Some computer systems, such as PCs, use a two-step process in which a simple bootstrap loader fetches a more complex boot program from disk, which in turn loads the kernel.

➔When a CPU receives a reset event—for instance, when it is powered up or rebooted—the instruction register is loaded with a predefined memory location, and execution starts there. At that location is the initial bootstrap program.

➔This program is in the form of read-only memory (ROM), because the RAM is inan unknown state at system startup. ROM is convenient because it needs no initialization and cannotbeinfectedbyacomputer virus.

➔Thebootstrapprogramcanperformavarietyoftasks.Usually,onetaskistorundiagnostics to determine the state of the machine. If the diagnostics pass, the program can continue with the booting steps. It can also initialize all aspects of the system,fromCPU registers to device controllers and the contents of main memory.

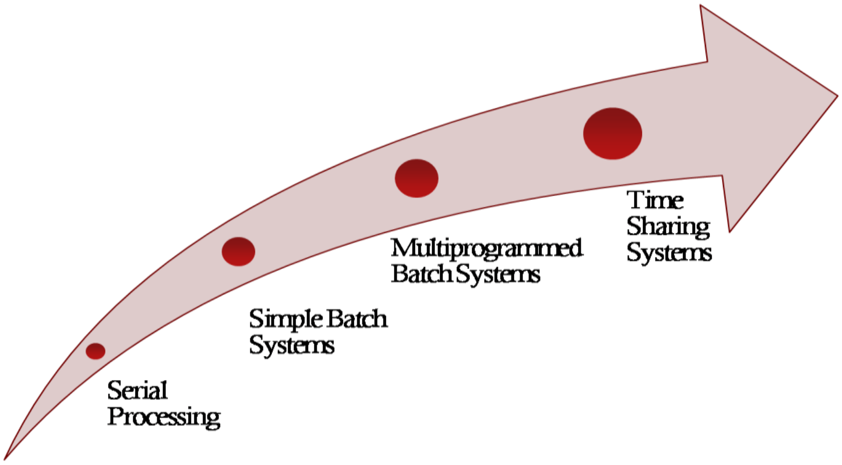
➔Sooner or later, it starts the operating system. Some systems—such as cellular phones, PDAs, and game consoles—store the entire operating system in ROM. Storing the operating system in ROM is suitable for small operating systems, simple supportinghardware, and rugged operation.

➔A problem with this approach is that changing the bootstrap code requires changing the ROM hardware chips. Some systems resolve this problem by using erasable programmable read-only memory (EPROM), which is read-only except when explicitly givenacommandtobecome writable.

➔All forms of ROM are also known as firmware, since their characteristics fall somewhere between those of hardware and those of software. A problem with firmware in general is that executing code there is slower than executing code in RAM.Somesystemsstorethe operating system in firmware and copy it to RAM for fast execution. A final issue with firmware is that it is relatively expensive, so usually only small amounts are available.

EVALUATIONOFOPERATINGSYSTEMS.

StagesofEvaluation



SerialProcessing

➔Usersaccessthecomputerinseries.Fromthelate1940'stomid1950's,theprogrammer interacted directly with computer hardware i.e., no operating system.

➔Thesemachineswererunwithaconsoleconsistingofdisplaylights,toggle switches, some form of input device and a printer. Programs in machine code are loaded withtheinputdevicelikecardreader.

➔If an error occur the program was halted and the error condition was indicated by lights. Programmers examine the registers and main memory to determine error.If the program is success, then output will appear on the printer.

➔Mainproblemhereisthesetuptime.Thatissingleprogramneedstoloadsource program into memory, saving the compiled (object) program and then loading and linkingtogether.

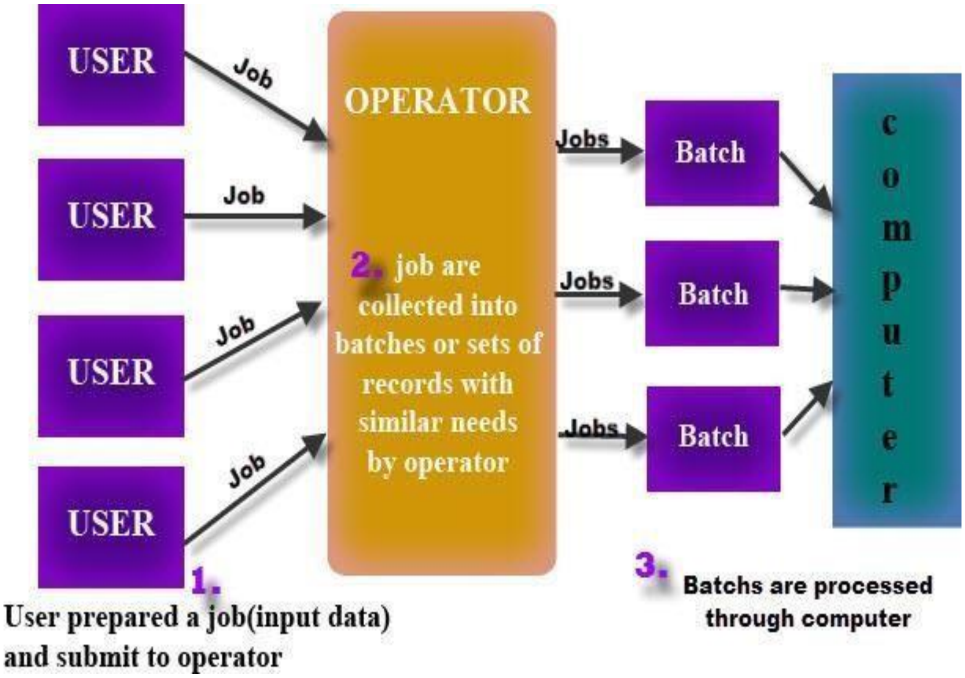
SimpleBatchSystems

➔Tospeedupprocessing,jobswithsimilarneedsarebatchedtogetherandrunas a group. Thus, the programmers will leave their programs with the operator. The operator will sort programs into batches with similar requirements.

TheproblemswithBatchSystemsare:

➔Lack of interaction between the user and job.CPU is often idle, because the speeds of the mechanical I/O devices are slower than CPU. For overcoming this problem use the Spooling

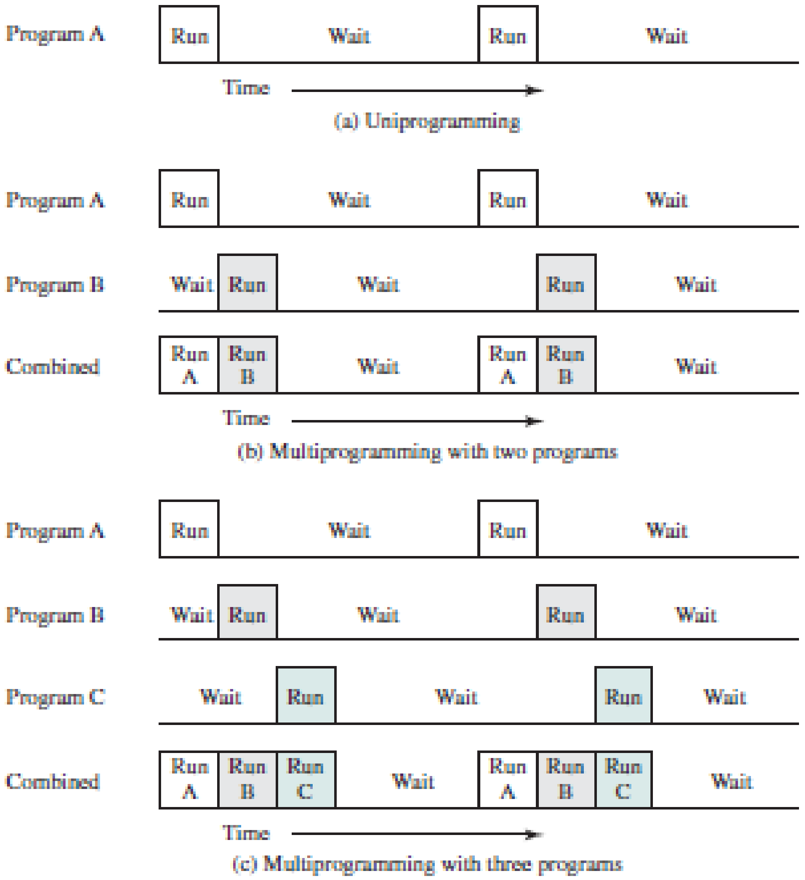
➔Technique.Spoolisabufferthatholdsoutputforadevice,suchasprinter,that can not accept interleaved data streams. That is when the job requests the printer tooutputaline.Thatlineiscopiedintoasystembufferandiswrittentothe disk. When the job is completed, the output is printed. Spooling technique can keep both the CPU and the I/O devices working at much higher rates.


USER
Job
USER
Job
USER
Job
USER
Job
OPERATOR
2. job are collected into batches or sets of records with similar needs by operator
Jobs
Batch
Jobs
Batch
Jobs
Batch
c o m p u t e r
1.
User prepared a job(input data) and submit to operator
3. Batchs are processed through computer

MultiprogrammedBatchSystems

➔Jobs must be run sequentially, on a first-come, first-served basis. However when several jobs are on a direct-access device like disk, job scheduling is possible. The mainaspectofjobschedulingismultiprogramming. Single user cannot keep the CPU or I/O devices busy at all times. ThusmultiprogrammingincreasesCPUutilization.

➔Inwhenonejobneedstowait,theCPUisswitchedtoanotherjob,andsoon. Eventually, the first job finishes waiting and gets the CPU back.

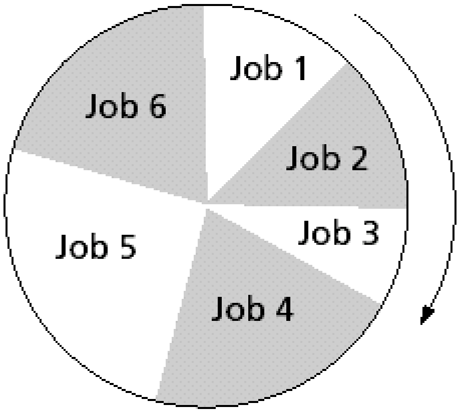


Time-SharingSystems

➔Time-sharingsystemsarenotavailablein 1960s.Time-sharing or multitaskingis alogical extension of multiprogramming. That is processors time is shared among multiple users simultaneously is called time-sharing. ThemaindifferencebetweenMultiprogrammedBatch Systems and Time-Sharing Systems is in multiprogrammedbatch systems its objective is maximize processor use, whereas in Time-Sharing Systems its objective is minimize response time.

➔MultiplejobsareexecutedbytheCPUbyswitchingbetweenthem,buttheswitches occur so frequently. Thus, the user can receives an immediate response. For example,in atransaction processing, processor execute each user program in a short burst or quantum of computation.Thatisifnusersarepresent,eachusercanget time quantum. When the user submitsthecommand,theresponsetimeissecondsatmost.

➔Operating system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time. Computer systems that were designed primarily as batchsystems have been modified to time-sharing systems.
ForexampleIBM's OS/360.
Time-sharing operating systems are even more complex than multi-programmed operating systems. As in multiprogramming, several jobs must be kept simultaneously in memory.



OBJECTIVESANDFUNCTIONSOFANOPERATINGSYSTEMS

AnOSisaprogramthatcontrolstheexecutionofapplicationprogramsandactsas an interface between applications and the computer hardware. It can be thought of as having three objectives:

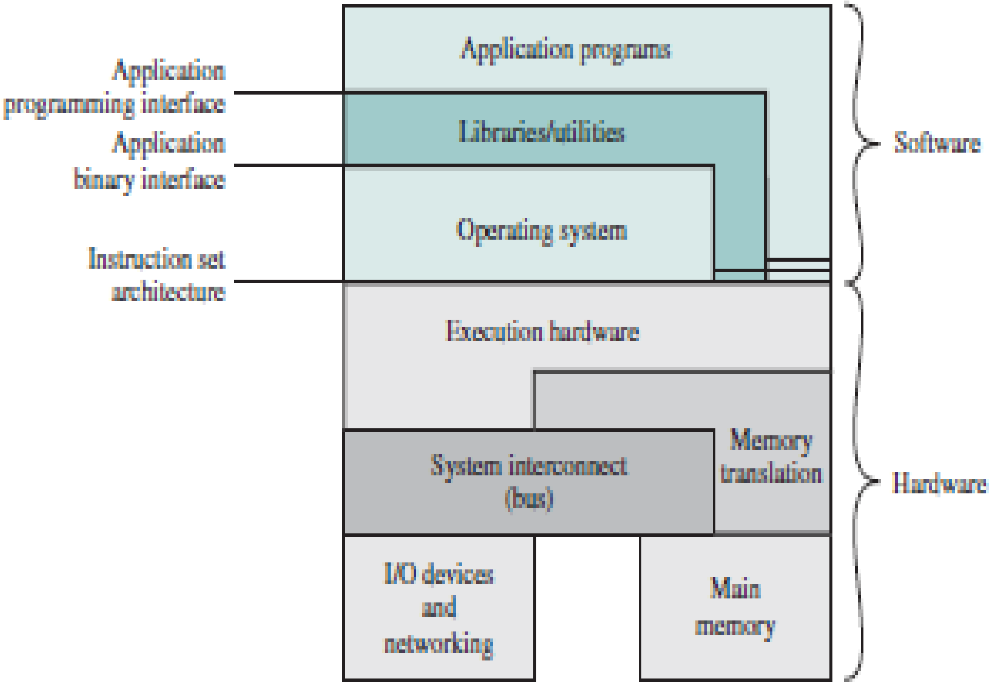
- **Convenience**:AnOSmakesacomputermoreconvenienttouse.
- **Efficiency**:AnOSallowsthecomputersystemresourcestobeusedinanefficient manner.
- **Ability to evolve**: An OS should be constructed in such a way as to permit the effectivedevelopment,testing,andintroductionofnewsystemfunctionswithout interfering with service.

TheOperatingSystemasaUser/ComputerInterface

➔The hardware and software used in providing applications to a user can be viewed in a layered or hierarchicalfashion.Theuserofthoseapplications,theenduser,generallyisnotconcernedwiththedetails of computer hardware. Thus, the end user views a computer system in terms of a set of applications.

➔An application can be expressed in a programming language and is developed by an application programmer. If one were to develop an application program as a set of machine instructions that is completelyresponsibleforcontrollingthecomputerhardware,onewouldbefacedwithanoverwhelmingly complex undertaking.

➔To ease this chore, a set of system programs is provided. Some of these programs are referred to as utilities, or library programs. These implement frequently used functions that assist in program creation, themanagement offiles, and thecontrol of I/O devices. A programmer will makeuse of thesefacilitiesin developinganapplication,andtheapplication,whileitisrunning,willinvoketheutilitiestoperformcertain functions.


Application programs
Libraries/utilities
Operating system
Execution hardware
Memory translation
System interconnect (bus)
I/O devices and networking
Main memory
Application programming interface
Application binary interface
Instruction set architecture
Software
Hardware

Three key interfacesinatypicalcomputersystem:

- **Instruction set architecture (ISA)** : The ISA defines the repertoire of machine language instructions that a computer can follow.This interface is the boundary between hardware andsoftware. Note that both application programs and utilities may access the ISA directly. For these programs, a subset of the instructionrepertoireisavailable(userISA).TheOShasaccesstoadditionalmachinelanguageinstructions that deal with managing system resources (system ISA).
- **Applicationbinaryinterface(ABI)**:TheABIdefinesastandardforbinaryportabilityacrossprograms. TheABIdefinesthesystemcallinterfacetotheoperatingsystemandthehardwareresources andservices available in a system through the user ISA.
- **Applicationprogramminginterface(API)**:TheAPIgivesaprogramaccesstothehardwareresources and services available in a system through the user ISA supplemented with high-level language (HLL) library calls. Any system calls are usually performed through libraries. Using an API enables application software to be ported easily, through recompilation, to other systems that support the same API.

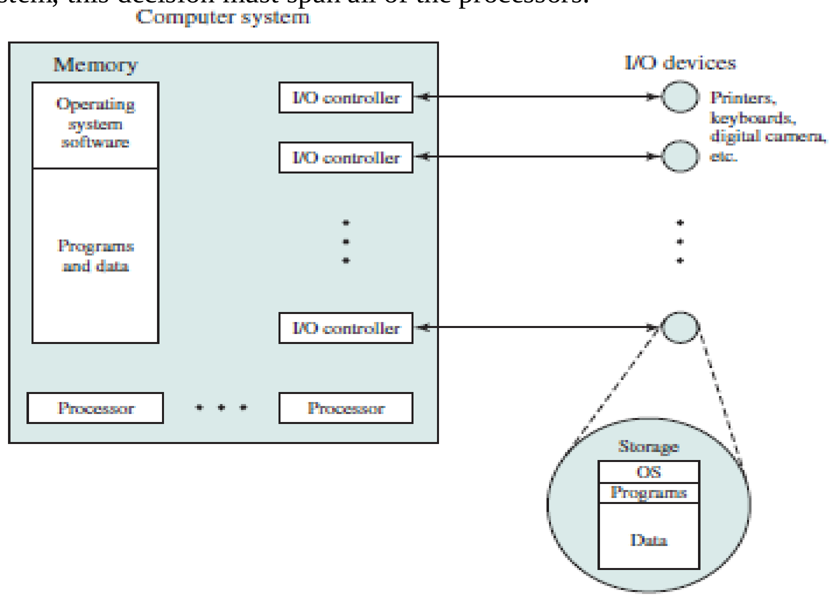
TheOperatingSystemasResourceManager

A computer is a set of resources for the movement, storage, and processing of data and for the control of these functions. The OS is responsible for managing these resources.

Bymanagingthecomputer’sresources,theOSisincontrolofthecomputer’ sbasic functions

ThemainresourcesthataremanagedbytheOS.AportionoftheOSisinmainmemory.Thisincludesthe **kernel**,or**nucleus**,whichcontainsthemostfrequentlyusedfunctionsintheOSand,atagiventime,other portionsoftheOScurrentlyinuse.Theremainderofmainmemorycontainsuserprogramsanddata.

The memory management hardware in the processor and the OS jointly control the allocation of main memory, as we shall see. The OS decides when an I/Odevice can be used by a program in executionand controlsaccesstoanduseoffiles.Theprocessoritselfisaresource,andtheOSmustdeterminehow much processor time is to be devoted to the execution of a particular user program. In the case of a multiple- processor system, this decision must span all of the processors.



EaseofEvolutionofanOperating System

AmajorOSwillevolveovertimeforanumberofreasons:

- **Hardwareupgradesplusnewtypesof hardware**
- **Newservices**
- **Fixes**

OPERATINGSYSTEMOPERATIONS

Modern operating systems are interrupt driven. If there are no processes to execute, OS will sit idle and wait for some event to happen. Interrupts could be hardware interrupts or software interrupts. The OS is designedtohandleboth.Atrap(oranexception)isasoftwaregeneratedinterruptcausedeitherbyanerror (e.g.dividebyzero)orbyaspecificrequestfromauserprogram.Aseparatecodesegmentiswritten in the OS to handle different types of interrupts. These codes are known as interrupt handlers/ interrupt service routine. A properly designed OS ensures that an illegal program should not harm the execution of other programs. To ensure this, the OS operates in dual mode.

Dualmodeofoperation

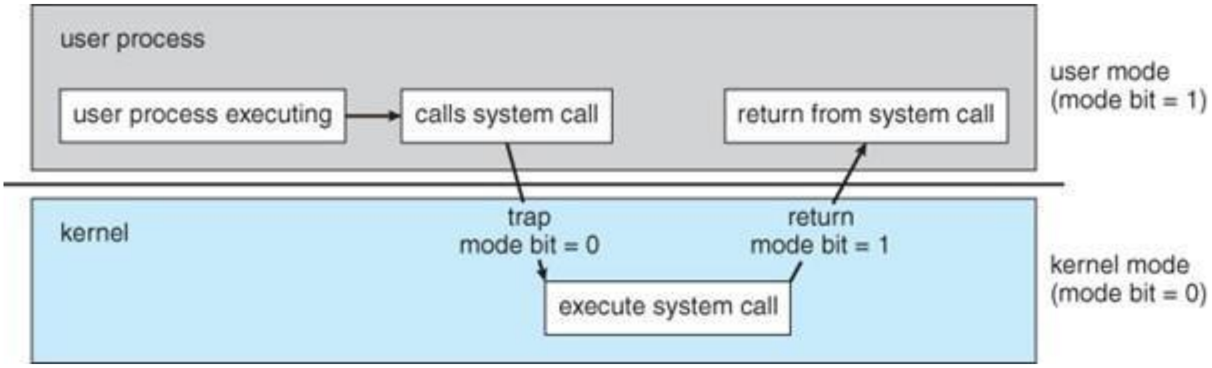
The OS is design in such a way that it is capable of differentiating between the execution of OS code and userdefinedcode.ToachievethisOSneedtwodifferentmodesofoperationsthisistherebycontrolledby mode bit added to hardware of computer system as shown in Table 4.

ModeType	**Definition**	**Mode Bit**	**Examples**
User Mode	UserDefinedcodesareexecuted	ModeBit=1	Creation of word document or in generaluserusinganyapplication program
KernelMode	OSsystemcodesareexecuted (also known as supervisor, system, or privileged mode)	ModeBit=0	Handling interrupts-Transferring controlofaprocessfromCPUtoI/O on request

UserandKernelModeofOperatingSystem

Transition from User to Kernel mode

When a user application is executing on the computer system OS is working in user mode. On signal of system call via user application, the OS transits from user mode to kernel mode to service that request as shown in Fig. 11.



Transitionfromusertokernelmode

Whentheuserstartsthesystemthehardwarestartsinmonitor/kernelmodeandloadstheoperatingsystem. OS has the initial control over the entire system, when instructions are executed in kernel mode. OS then starts the user processes in user mode and on occurrence of trap, interrupt or system call again switch to kernelmodeandgainscontrolofthesystem.Systemcallsaredesignedfortheuserprogramsthroughwhich user can ask OS to perform tasks reserved for operating system. System calls usually take the form of the trap.OncetheOSservicetheinterruptittransferscontrolbacktouserprogramhenceusermodebysetting mode bit=1.

BenefitsofDualMode

The dual mode of operation protects the operating system from errant users, and errant users from one anotherbydesignatingsomeofthemachineinstructionsthatmaycauseharmasprivilegedinstructions. Theseinstructionscanexecuteonlyinkernelmode.Ifanattemptismadetoexecuteaprivilegedinstruction in user mode, the hardware does not execute the instruction, but rather treats the instruction as illegal and traps to the operating system. Examples of privileged instructions:

1. Switchingtokernel mode
2. ManagingI/Ocontrol
3. TimerManagement
4. InterruptManagement

Timer

SinceOS operatesindualmodeit shouldmaintain control over CPU. Thesystem shouldnot allowa user application:

1. Tobestuckinaninfinite loop
2. Tofailtocallsystem services
3. NeverreturncontroltotheOS

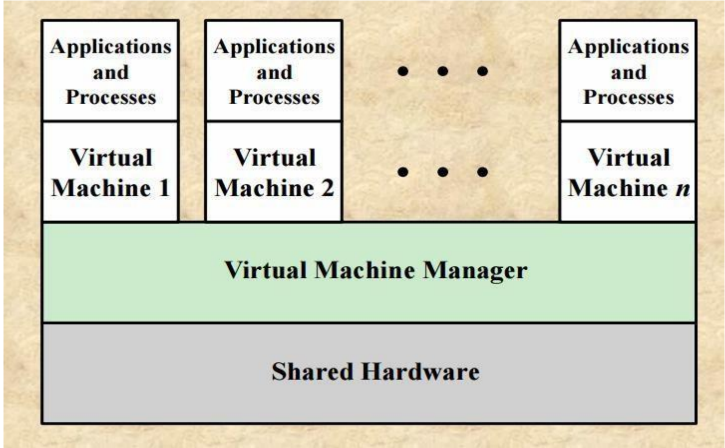
Toachievethisgoal,wecanusetimer.Thistimercontrolmechanismwillinterruptthesystemataspecified period; thereby preventing user program from running too long. This can be implemented either as fixed timer or variable timer

AdditionalTopics

VirtualMachines(VM)

Virtualization technology enables a single PCor server to simultaneously run multiple operating systems or multiple sessions of a single OS

- A machinewithvirtualizationsoftwarecanhostnumerous applications,includingthosethatrunon different operating systems, on a single platform
- The host operating system can support a number of virtual machines, each of which has the characteristics of a particular OS
- Thesolutionthat enablesvirtualizationisavirtualmachinemonitor(VMM),orhypervisor



Avirtual machinetakesthelayeredapproachtoitslogicalconclusion.Ittreatshardwareandtheopera with its

own (virtual) memory.

Theresourcesofthephysicalcomputerarésharedtocreatethevirtualmachines.

1. CPUschedulingcancreatetheappearancethatusershavetheirownprocessor.
2. Spoolingandafilesystemcanprovidevirtualcardreadersandvirtualline printers.
3. Anormalusertime-sharingterminalservesasthevirtualmachineoperator’sconsole.

Advantages/DisadvantagesofVirtualMachines

Thevirtual-machineconceptprovidescompleteprotectionofsystemresourcessinceeachvirtual machine is isolated from all other virtual machines.

Thisisolation,however,permitsnodirectsharingofresources.

Avirtual-machinesystemisaperfectvehicleforoperating-systemsresearchanddevelopment.System development is done on the virtual machine, instead of on a physical machine and so does not disrupt normal system operation.

Thevirtualmachineconceptisdifficulttoimplementduetotheeffortrequiredtoprovideanexact duplicate to the underlying machine.

CS6401-OPERATINGSYSTEMS

UNITII PROCESSMANAGEMENT

Processes-ProcessConcept,ProcessScheduling,OperationsonProcesses, Interprocess Communication;Threads-Overview,MulticoreProgramming,MultithreadingModels; Windows7-ThreadandSMPManagement.ProcessSynchronization -CriticalSection Problem, Mutex Locks, Semophores, Monitors; CPU Scheduling and Deadlocks.

PROCESSCONCEPTS

ProcessConcept

➔Aprocesscanbethoughtofasaprogramin execution.

➔Aprocessistheunitoftheunitofworkin amoderntime-sharingsystem.

Aprocessismorethantheprogramcode,whichissometimesknownasthe**textsection**. It also includes the current activity, as represented by the value of the **programcounter** and the contents of the processor's registers.

Aprocessgenerallyalsoincludestheprocess**stack**,whichcontainstemporarydata(such asfunctionparameters,returnaddresses,andlocalvariables),anda **datasection**,which contains global variables. A process may also include a **heap**, which is memory that is dynamically allocated during process run time.

Differencebetweenprogramand process

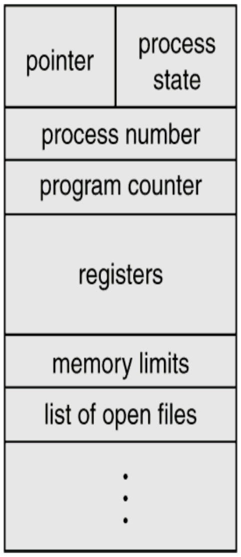
➔A program is a passive entity, such as the contents of a file stored on disk, whereas a process is an active entity, with a program counter specifying the next instruction to execute and a set of associated resources.

ProcessControlBlock(PCB)

➔Eachprocessis representedintheoperatingsystembyaprocess control block (PCB)- also called a task control block.

➔APCBdefinesaprocesstotheoperatingsystem.

➔Itcontainstheentireinformationaboutaprocess. Some of the information a PCB.



Process state: The state may be new, ready, running, and waiting, halted, and SO on.

Program counter: The counter indicates the address of the next instruction to be executed for this process.

CPU registers: The registers varyin number and type, depending on the computer architecture.

CPU-schedulinginformation:Thisinformationincludesaprocess priority, pointers to scheduling queues, and any other schedulingparameters.

Memory-managementinformation:Thisinformationmayincludesuch informationasthevalueofthebaseandlimitregisters,thepagetables,or the segment tables, depending on the memory system used by the operating system.

Accountinginformation:Thisinformationincludestheamount of CPU and real time used, time limits, account numbers, job or process numbers, and so on.

Status information: The information includes the list of I/Odevices allocated to this process, a list of open files, and so on.

<u>ProcessStates:</u>

□ Asaprocessexecutes,itchangesstate.

□ The stateofa process is defined inpartbythecurrentactivityofthatprocess. □ Each process may be in one of the following states:

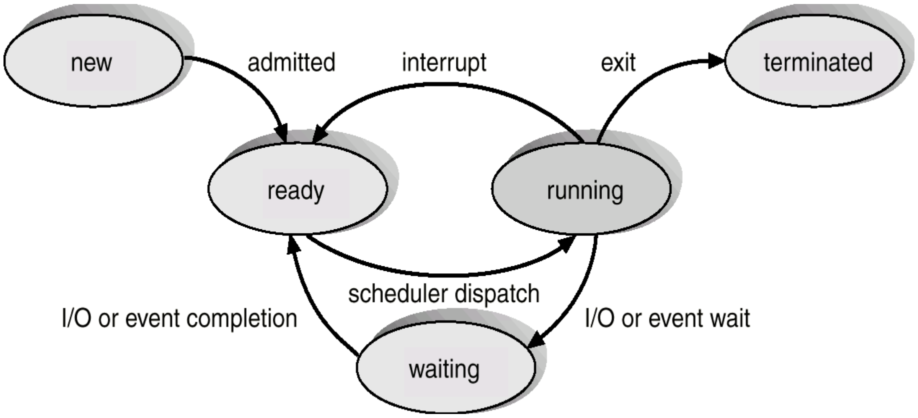
□ **New**:Theprocessisbeingcreated.

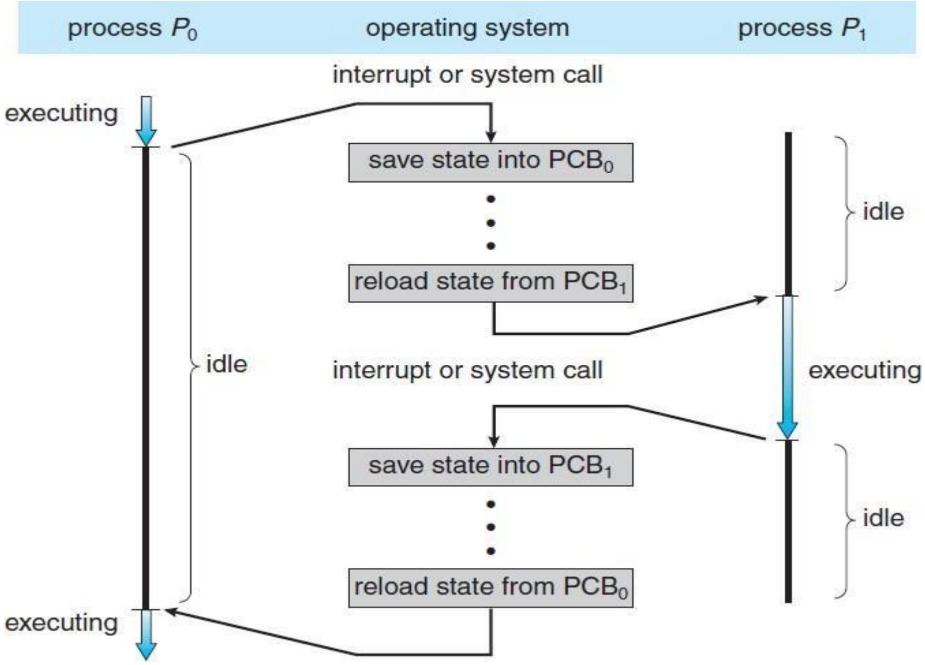
□ **Running**:Instructionsarebeingexecuted.

□**Waiting**:Theprocessiswaitingforsomeeventtooccur(suchasan I/O completion or reception of a signal).

□ **Ready**:Theprocessiswaitingtobeassigned toaprocessor.

□ **Terminated**:Theprocesshasfinishedexecution.





DiagramshowsCPUswitchfromprocesstoprocess.

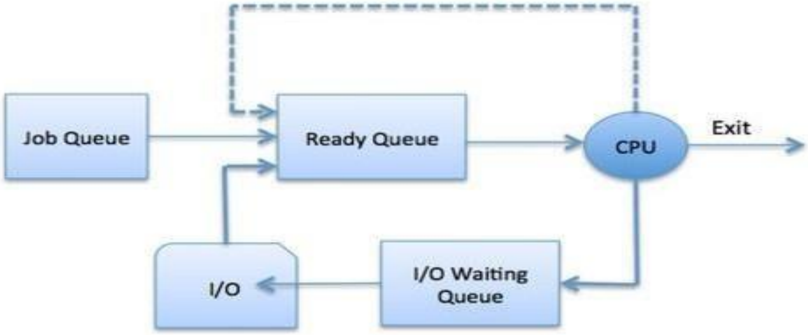
PROCESSSCHEDULING

- Theobjectiveofmultiprogrammingistohavesomeprocessrunningatall times, so as to maximize CPU utilization.

SchedulingQueues

Thereare3typesofschedulingqueues.Theyare:

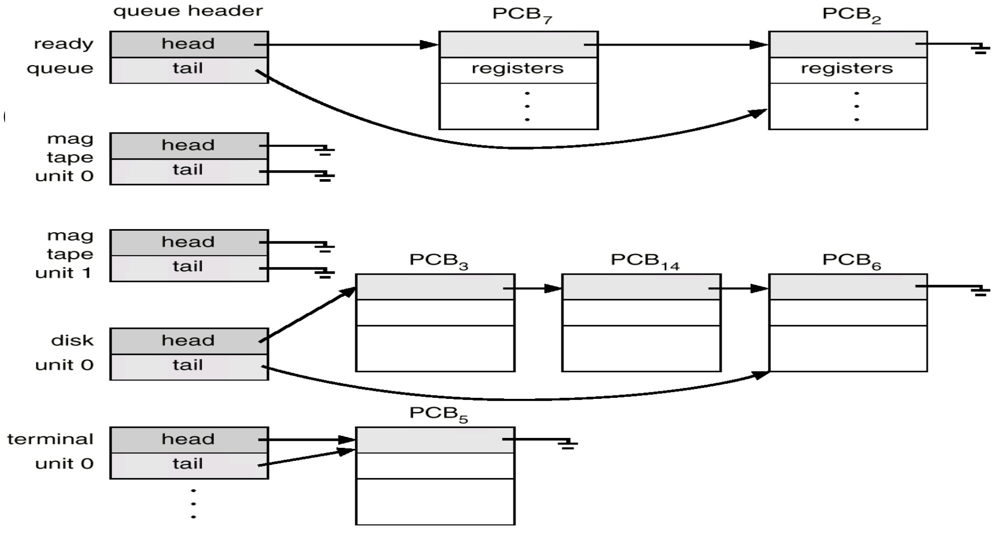
1. Job Queue
2. ReadyQueue
3. DeviceQueue



Asprocessesenterthesystem,theyareputinto a**job queue.**

The processes that are residing in main memory and are ready and waiting to executeare kept on a list called the **ready queue**.

Thelistofprocesseswaitingforan I/Odeviceiskeptina**devicequeue**forthat particular device.



ueueuntilitisselected or ts

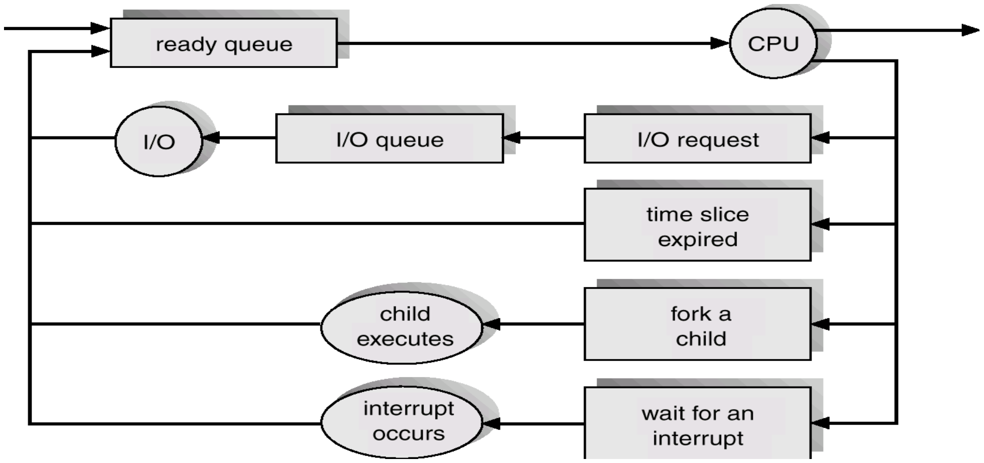
(

could occur:

- TheprocesscouldissueanI/Orequest,andthenbeplacedinan I/O queue.
- Theprocesscouldcreateanewsubprocessandwaitforits

termination.

- Theprocesscouldberemovedforciblyfromthe CPU,asaresultof aninterrupt, and be put back in the ready Queue.
- Acommonrepresentationofprocessschedulingisaqueueingdiagram.



Schedulers

- Theoperatingsystemmustselect,forschedulingpurposes,processesfrom thesequeues in some order
- Theselectionprocessiscarriedoutbytheappropriatescheduler.

Theyare:

1. Long-termSchedulerorJobScheduler
2. Short-termSchedulerorCPUScheduler
3. MediumtermScheduler

Long-TermScheduler

- The**long-termscheduler**,or**jobscheduler**,selectsprocessesfromthispoolandloads themintomemoryforexecution.Itisinvokedveryinfrequently.Itcontrolsthedegree of multiprogramming.

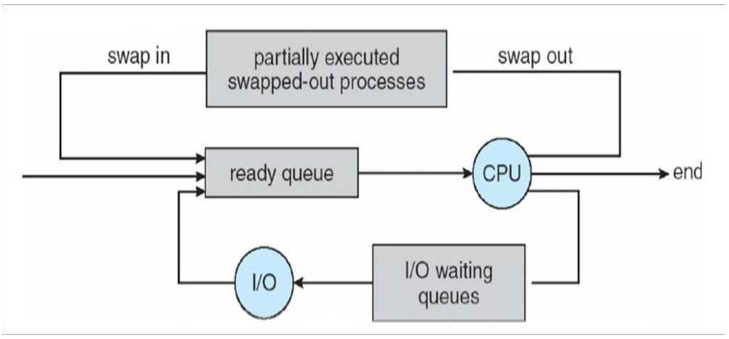
Short-TermScheduler

- The**short-termscheduler**,or**CPUscheduler**,selectsfromamongthe processesthatarereadytoexecute,andallocatestheCPUtooneofthem.Itisinvoked very frequently.
- Processescanbedescribedaseither**I/Obound**or**CPUbound**.
- An **I\O-bound process** spends more of its time doing I/O than it spends doing computations.
- A**CPU-boundprocess**,ontheotherhand,generatesI/Orequestsinfrequently, using more of its time doing computation than an I/O-bound processuses.
- The system with the best performance will have a combination of CPU- bound and I/O-bound processes.

MediumTermScheduler

- Someoperatingsystems,suchastime-sharingsystems,mayintroduceanadditional, intermediate level of scheduling.
- The key idea is medium-term scheduler, removes processes from memory and thus reduces the degree of multiprogramming.

- Atsomelatertime,theprocesscanbereintroducedintomemoryanditsexecution can be continued where it left off. This scheme is called swapping.



ContextSwitching

- SwitchingtheCPU toanotherprocessrequiressaving thestateof theold process andloadingthesavedstateforthenewprocess.Thistaskisknownasacontext switch.
- Context-switchtimeispure overhead,becausethe systemdoesnousefulwork while switching.
- Its speed varies from machine to machine, depending on the memory speed,the numberofregistersthatmustbecopied,andtheexistenceofspecial instructions.

1. ProcessCreation

OPERATIONSONPROCESS

Aprocessmaycreateseveralnewprocesses,duringexecution.

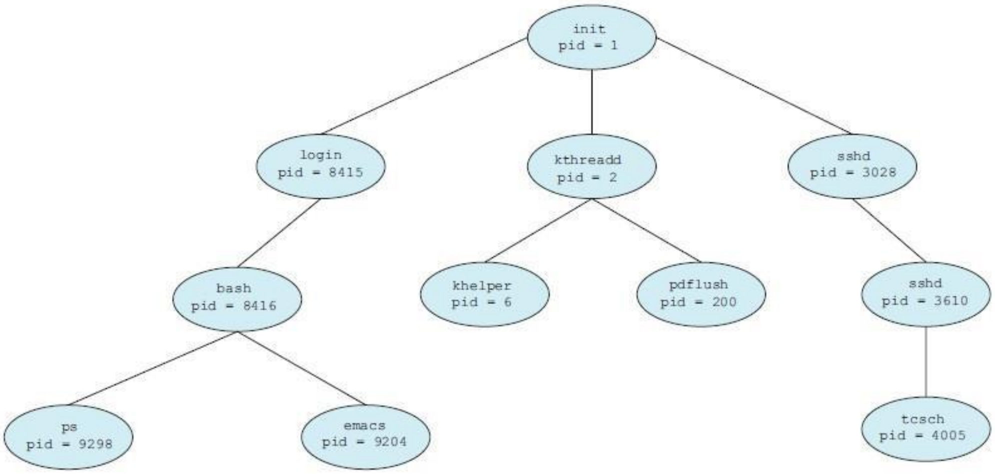
The creating process is called a **parent** process, whereas the new processes are calledthe **children** of that process.

Whenaprocesscreatesanewprocess,twopossibilitiesexist**intermsof execution:**

1. Theparentcontinuestoexecuteconcurrentlywithitschildren.
2. Theparentwaitsuntilsomeorallofitschildrenhaveterminated.

Therearealsotwopossibilities**intermsoftheaddressspace**ofthenewprocess:

1. Thechildprocessisaduplicateoftheparent process.



2. Thechildprocesshasaprogramloadedintoit.

In UNIX, each process is identified by its process identifier, which isa uniqueinteger. A new process is created by the **fork** system call.

AtreeofprocessesonatypicalLinuxsystem.

wesee twochildrenof**init**—kthreaddand**sshd**.

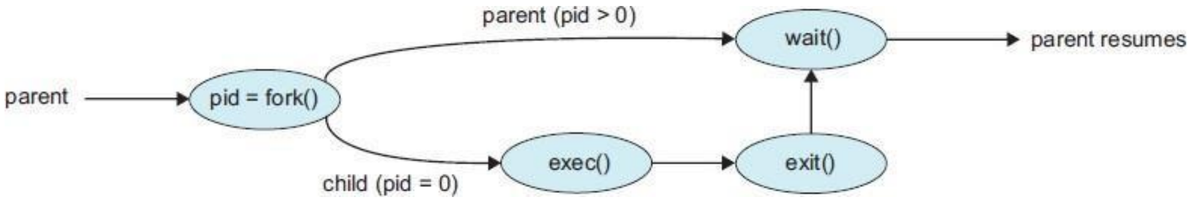
The kthreadd process is responsible for creating additional processes that perform tasks on behalf of the kernel (in this situation, khelper and pdflush).

Thesshdprocessisresponsibleformanagingclientsthatconnecttothesystembyusingssh (which is short for *secure shell*). The login process is responsible for managing clients that directly log onto the system.

In general, when a process creates a child process, that child process will need certain resources (CPU time, memory, files, I/O devices) to accomplish its task.

A child process may be able to obtain its resources directly from the operating system, or it may be constrained to a subset of the resources of the parent process.

The parent may have to partition its resources among its children, or it may be able to share some resources (such as memory or files) among several of its children. Restricting a child processtoasubsetoftheparent'sresourcespreventsanyprocessfromoverloadingthesystem by creating too many child processes.



2. ProcessTermination

A process terminates when it finishes executing its final statement and asksthe operating system to delete it by using the **exit** system call.

Atthatpoint,theprocessmayreturndata(output)toitsparentprocess(viathe**wait** systemcall).

Aprocesscan causetheterminationofanotherprocessvia an appropriatesystem call.

A parent may terminate the execution of one of its children for a variety of reasons, such as these:

1. The child has exceeded its usage of some of the resources that it hasBeen allocated.

2. Thetaskassigned tothechildisnolongerrequired.
3. Theparentisexiting,andtheoperatingsystemdoesnotallowachildtocontinue ifitsparentterminates.Onsuchsystems,ifaprocessterminates(eithernormally orabnormally),thenallits children must also be terminated. Thisphenomenon,referredtoascascading**termination**,isnormallyinitiated by the operating system.

Whenaprocessterminates,itsresourcesarede-allocatedbytheoperatingsystem.

Aprocessthathasterminated,butwhoseparenthasnotyetcalledwait(),isknownasazombie process.

Nowconsiderwhatwouldhappenifaparentdidnotinvokewait()andinsteadterminated, thereby leaving its child processes as orphans.

CO-OPERATINGPROCESS

Processesexecutingconcurrentlyintheoperatingsystem maybeeither**independentprocesses** or**cooperating processes**.

Aprocessisindependentifitcannotaffectorbeaffectedbytheotherprocessesexecutinginthe system. Any process that does not share data with any other process is independent.

Aprocessiscooperatingifitcanaffectorbeaffectedbytheotherprocessesexecutinginthe system. Clearly, any process that shares data with other processes is a cooperating process.

Thereareseveralreasonsforprovidinganenvironmentthatallowsprocesscooperation:

• **Information sharing.** Since several users may be interested in the same piece of information (forinstance,asharedfile),wemustprovideanenvironmenttoallowconcurrent accesstosuch information.

• **Computation speedup**. If we want a particular task to run faster, we must break it into subtasks,eachofwhichwillbeexecutinginparallelwiththeothers.Noticethatsuchaspeedup can be achieved only if the computer has multiple processing cores.

• **Modularity.**Wemaywanttoconstructthesysteminamodularfashion,dividingthesystem functions into separate processes or threads.

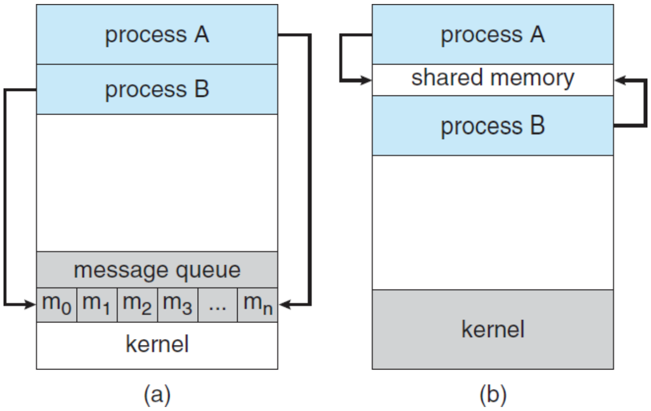
• **Convenience.**Evenanindividualusermayworkonmanytasksatthesametime.Forinstance, a user may be editing, listening to music, and compiling in parallel.

INTERPROCESSCOMMUNICATION

Cooperating processes require an **Inter Process Communication (IPC)** mechanism that will allow them to exchange data and information. There are two fundamental models of interprocess communication: **shared memory** and **message passing**.

In the shared-memory model, a region of memory that is shared by cooperating processes is established. Processes can then exchange information by reading and writing data to the shared region.

In the message-passing model, communication takes place by means ofmessagesexchangedbetween the cooperating processes.



(a)Messagepassing. (b)Sharedmemory.

Shared-MemorySystems

Interprocesscommunicationusingsharedmemoryrequirescommunicating processes to establish a region of shared memory.

Other processes that wish to communicate using this shared-memory segment must attach it totheir address space.

Messagepassing

Messagepassingprovidesamechanismtoallowprocessestocommunicateandtosynchronize their actions without sharing the same address space.

1. BasicStructure:

IfprocessesPandQwanttocommunicate,theymustsendmessagestoandreceive messages from each other; a communication link must exist between them.

Physicalimplementationofthelinkisdonethroughahardwarebus, network etc,

There are several methods for logically implementing a link and the operations:

1. **Directorindirectcommunication**
2. **Symmetricorasymmetriccommunication**
3. **Automaticorexplicitbuffering**
4. **Sendbycopyorsend byreference**
5. **Fixed-sizedorvariable-sized messages**

2. Naming

Processesthatwanttocommunicatemusthaveawaytorefertoeachother.

Theycanuseeitherdirect orindirectcommunication.

1. **DirectCommunication**

   Eachprocessthat wants to communicate must explicitlyname therecipientor sender of the communication.

   Acommunicationlinkinthisschemehasthefollowingproperties:

   i. Alinkisestablishedautomaticallybetweeneverypairofprocesses that wanttocommunicate.Theprocessesneedto know only each other's identity to communicate.
   ii. Alinkisassociatedwithexactlytwo processes.
   iii. Exactlyonelinkexistsbetweeneachpairofprocesses.

   There are two ways of addressing namely

   Symmetry in addressing
   Asymmetryinaddressing

   In symmetry in addressing, the sendandreceive primitivesare definedas: send(P, message)□ Send a message to process P
   receive(Q,message)□Receiveamessagefrom Q

   Inasymmetryinaddressing,thesend&receiveprimitivesare defined as:
   send(p,message)senda□messagetoprocesspreceive(id, message)
   receive message from□any process

2. **IndirectCommunication**

   Withindirectcommunication,themessagesaresenttoandreceivedfrom mailboxes, or ports.

   Thesendandreceiveprimitivesaredefinedasfollows:
   send (A, message) □Sendamessagetomailbox A.receive
   (A, message) Receive □a message from mailbox A.

Acommunicationlinkhasthefollowingproperties:

i. Alinkisestablishedbetweenapairofprocessesonlyifbothmembers of the pair have a shared mailbox.
ii. Alinkmaybeassociatedwithmorethantwoprocesses.
iii. Anumberofdifferentlinksmayexistbetweeneachpairofcommunicatingprocesses,witheachlinkcorrespondingtoonemailbox.

3. **Buffering**

   Alinkhassomecapacitythatdeterminesthenumberofmessagethat canreside in it temporarily. This propertycan be viewed as a queue of messages attached to the link.

   Therearethreewaysthat such aqueuecanbeimplemented.

   Zero capacity : Queue length of maximum is 0. No message is waitinginaqueue.Thesendermustwaituntiltherecipientreceivesthemessage.

   Bounded capacity: The queue has finite length n. Thus at most n messages can reside in it.

   Unboundedcapacity:Thequeuehaspotentiallyinfinitelength.Thus any number of messages can wait in it. The sender is never delayed

4. **Synchronization**

   Messagepassingmaybeeitherblockingornon-blocking.

1. **BlockingSend** -Thesenderblocksitselftillthemessagesentbyitis received by the receiver.
2. **Non-blocking Send** - The sender does not block itself after sendingthe message but continues with its normal operation.
3. **BlockingReceive**-Thereceiverblocksitselfuntilitreceivesthe message.
4. **Non-blockingReceive**–Thereceiverdoesnotblockitself.

THREADS

Thread

AthreadisabasicunitofCPUutilization;itcomprisesathread ID,aprogramcounter,aregister set, and a stack.

It shares with other threads belonging to the same process its code section, data section, and otheroperating-systemresources,suchasopenfilesandsignals.Traditional(orheavyweight) process has a single thread of control.

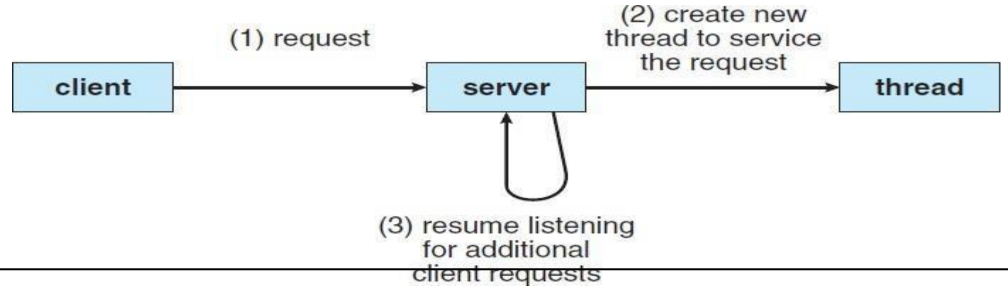
Ifaprocess hasmultiplethreads ofcontrol,it canperformmorethanonetask at atime.

Motivation

Mostsoftwareapplicationsthatrunonmoderncomputersaremultithreaded.Anapplication typically is implemented as a separate process with several threads of control.

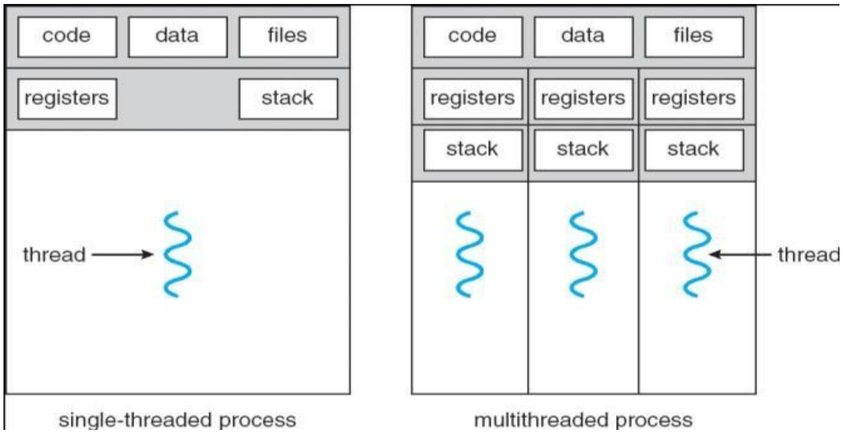
Aweb browser might haveonethreaddisplayimages ortext whileanotherthread retrieves data from the network.

A word processor may have a thread for displaying graphics, another thread for responding to keystrokesfromtheuser, andathirdthreadforperformingspellingandgrammarcheckinginthe background.


(1) request
(2) create new thread to service the request
client
server
thread
(3) resume listening for additional client requests

MULTITHREADING

Multithreadingistheabilityofaprogramoranoperatingsystemprocesstomanageitsuseby more than one user at a time and to even manage multiple requests by the same user without having to have multiple copies of the programming running in the computer.



Benefits

Therearefourmajorcategoriesofbenefitstomulti-threading:

1. **Responsiveness**-Onethreadmayproviderapidresponsewhileotherthreadsareblocked or slowed down doing intensive calculations.
2. **Resourcesharing**-Bydefaultthreadssharecommoncode,data,andotherresources, whichallowsmultipletaskstobeperformedsimultaneouslyinasingleaddressspace.
3. **Economy-**Creatingandmanagingthreads(andcontextswitchesbetweenthem)is much faster than performing the same tasks for processes.
4. **Scalability**, i.e. Utilization of multiprocessor architectures - A single threaded processcanonlyrunononeCPU,nomatterhowmanymaybeavailable,whereastheexecutionof a multi-threaded application may be split amongst available processors

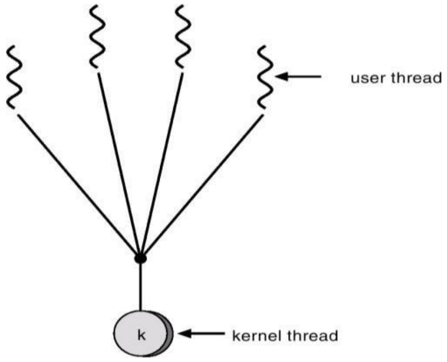
MultithreadingModels

1. Many-to-One
2. One-to-One
3. Many-to-Many

1. Many-to-One:

Many to one model maps many user level threads tooneKernelleveltthread.Threadmanagement is done inuser space. When thread makes a blocking system call, the entire process will be blocks. Only one thread can access the Kernel at a time,so multiple threads are unable to run in parallel on multiprocessors.

If the user level thread libraries are implemented in the operatingsysteminsuchaway that systemdoesnotsupport them then Kernel threads use the many to one relationship modes.

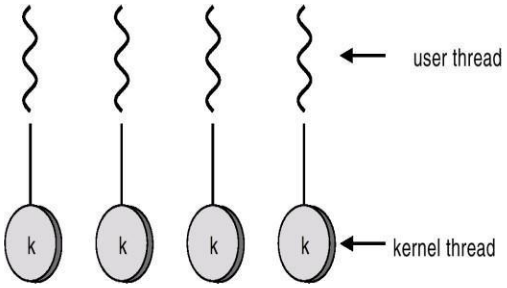


2. One-to-One:

Thereisonetoonerelationshipofuserlevel thread to the kernel level thread.

This modelprovidesmoreconcurrencythan the many to one model.

Italsoanotherthreadtorunwhenathread



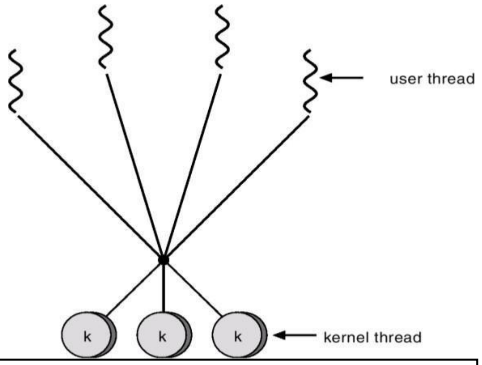
makes a blocking system call. It supports multiplethreadtoexecuteinparallel on microprocessors.

3. Many-to-ManyModel:

In this model, many user level threads multiplexes to theKernelthreadofsmalleror equal numbers.

ThenumberofKernelthreadsmaybespecifictoeither a particularapplicationoraparticular machine.

Inthismodel,developerscancreateasmanyuserthreads as necessary and the corresponding Kernel threads can run inparallels on a multiprocessor.



THREADINGISSUES:

1. **fork()and exec()systemcalls.**

   Afork ()system call mayduplicateall threads orduplicateonlythethread that invoked fork().

   If a thread invoke exec() system call ,the program specified in the parameter to exec will replace the entire process.

2. **Threadcancellation.**

   Itis thetask ofterminatingathread beforeit has completed .Athread that is to be cancelled is called a target thread.
   Therearetwotypesofcancellationnamely

   1. **Asynchronous Cancellation**– One thread immediately terminatesthe target thread.
   2. **Deferred Cancellation** – The target thread can periodically check if it should terminate , and does so in an orderly fashion.

3. **Signalhandling**

   1. Asignalis aused tonotifyaprocessthataparticulareventhasoccurred.
   2. Ageneratedsignalisdeliveredtotheprocess.

      a. Deliverthesignaltothethreadtowhichthesignalapplies.b. Deliver the signal to every thread in the process.

      c. Deliverthesignaltocertainthreadsintheprocess.

      d. Assignaspecificthreadtoreceiveallsignals fortheprocess.

   3. Oncedeliveredthesignalmustbehandled.a. Signalishandledby

      i. Adefaultsignalhandler

      ii. Auserdefinedsignal handler

4. **Threadpools**

   - CreationofunlimitedthreadsexhaustsystemresourcessuchasCPUtimeor memory. Hencewe use a thread pool.
   - In a thread pool , a number of threads are created at process startup and placed in the pool.
   - Whenthereisa needforathreadtheprocesswillpick athreadfrom thepooland assign it a task.

- Aftercompletionofthetask,thethreadisreturnedtothepool.

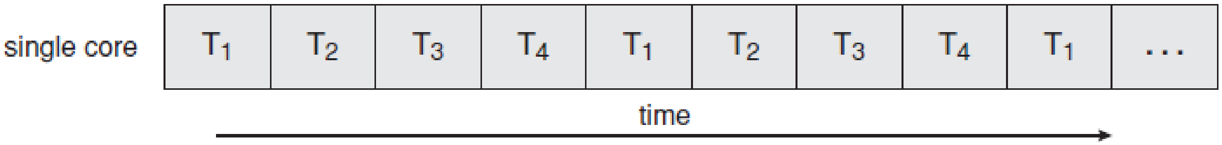
5. **Threadspecificdata**
   Threads belonging to aprocess share the data of the process. However each thread mightneed its own copy of certain data known as thread-specific data.

MULTICOREPRORGAMMING

Single-CPUsystemsevolvedintomulti-CPUsystems.Amorerecent,similartrendinsystem design is to place multiple computing cores on a single chip.

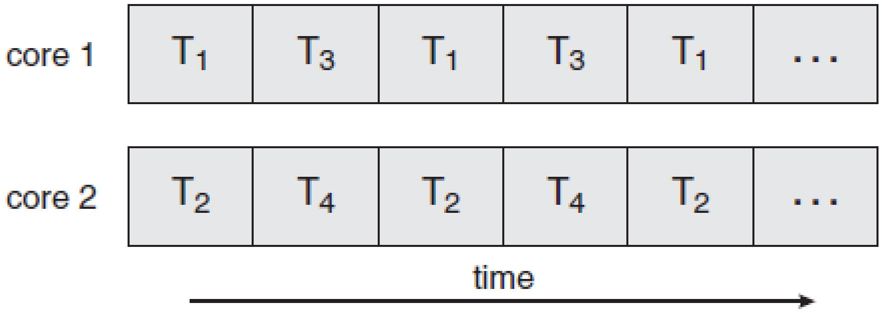
Eachcoreappearsasaseparateprocessortotheoperatingsystem.Whetherthecoresappearacross CPU chips or within CPU chips, we call these systems multicore or multiprocessor systems.

Multithreadedprogrammingprovidesamechanismformoreefficientuseofthesemultiple computing cores and improved concurrency.



Asystemisparallelifitcanperformmorethanonetasksimultaneously.

Aconcurrentsystemsupportsmorethanonetaskbyallowingallthetaskstomakeprogress.Thus, it is possible to have concurrency without parallelism.



In general,fiveareaspresent challengesinprogrammingformulticoresystems:

1. **Identifying tasks**. This involves examining applications to find areas that can be divided into separate, concurrent tasks.

2. **Balance.**Whileidentifyingtasksthatcanruninparallel,programmersmustalsoensurethat the tasks perform equal work of equal value.

3. **Datasplitting**.Justasapplicationsaredividedintoseparatetasks,thedataaccessedand manipulated by the tasks must be divided to run on separate cores.

4. **Datadependency**.Thedataaccessedbythetasksmustbeexaminedfordependenciesbetween two or more tasks. When one task depends on data from another, programmers must ensure that the execution of the tasks is synchronized to accommodate the data dependency.

5. **Testing and debugging**. When a program is running in parallel on multiple cores, many different execution paths are possible. Testing and debugging such concurrent programs is inherently more difficult than testing and debugging single-threaded applications.

TypesofParallelism
Ingeneral,therearetwotypesofparallelism:**dataparallelismandtaskparallelism**.
Dataparallelismfocusesondistributingsubsetsofthesamedataacrossmultiplecomputing cores and performing the same operation on each core.
Taskparallelisminvolvesdistributingnotdatabuttasks(threads)acrossmultiplecomputing cores.

PROCESSSYNCHRONIZATION

□ Concurrentaccesstoshareddatamayresult indatainconsistency.

□ Maintaining data consistency requires mechanisms to ensure the orderly execution ofcooperating processes.

□ Shared-memorysolution to bounded-butterproblem allows at most *n*–1items in bufferat the same time. A solution, where all *N* buffers are used is not simple.

□ Suppose that we modify the producer-consumer code by adding a variable *counter*, initialized to 0 and increment it each time a new item is added to the buffer

□ Racecondition:Thesituationwhereseveralprocessesaccess –andmanipulateshareddata concurrently. The final value of the shared data depends upon which process finishes last.

□ Topreventraceconditions, concurrentprocesses must besynchronized.

THECRITICAL-SECTIONPROBLEM

Definition:Eachprocesshasasegmentofcode,calledacriticalsection(CS),inwhichthe process may be changing common variables, updating a table, writing a file, and so on.

 The important feature of the system is that, when one process is executing in its CS, no other process is to be allowed to execute in its CS.

 That is, no two processes are executing in their CSs at the same time.
 Each process must request permission to enter its CS. The section of code implementingthis request is the entry section.

 The CS may be followed by an exit section.

□ The remaining code is the remainder section.

RequirementstobesatisfiedforaSolutiontotheCritical-Section Problem:

1. **Mutual Exclusion -** If process Pi is executing in its critical section, then no other processes can be executing in their critical sections.
2. **Progress-**Ifnoprocessisexecutinginitscriticalsectionandthereexistsomeprocesses thatwishtoentertheircriticalsection,thentheselectionoftheprocessesthatwillenter thecriticalsectionnextcannotbe postponedindefinitely.
3. **Bounded Waiting -** A bound must exist on the number of times that other processesareallowedtoentertheircriticalsectionsafteraprocesshas made a request to enter its critical section and before that requestis granted.

GeneralstructureofprocessPi

```
{
        entry section
        criticalsection

        exitsection

        remaindersection

} while(1);
```

Twogeneralapproachesareusedtohandlecriticalsectionsinoperatingsystems:preemptive kernels and nonpreemptive kernels.

□ Apreemptivekernelallowsaprocesstobepreempted whileitisrunninginkernelmode.

□ A non-preemptive kernel does not allow a process running in kernel mode to be preempted; a kernelmodeprocesswillrununtilitexitskernelmode,blocks,orvoluntarilyyieldscontrolofthe CPU.

MUTEXLOCKS

□ Operating-systems designers build software tools to solve the critical-section problem. Thesimplest of these tools is the mutex lock.

□ We use the mutex lock to protect critical regions and thus preventrace conditions.

whenitexitsthecritical section.

□ That is, a process must acquire the lock before entering a critical section; it releases the lock

Solutiontothecritical-sectionproblemusingmutexlocks.

□ The acquire()function acquires the lock, and the release() function releases the lock.

```
do {
        acquire lock
        critical section
        release lock
        remaindersection
}while(true);
```

 A mutex lock has a boolean variable available whose value indicates if the lock is available or not.

 If the lock isavailable,acalltoacquire()succeeds,andthelockisthenconsidered unavailable.

 A process that attempts to acquire an unavailable lock is blocked until the lock is released.

 The definition of acquire() is as follows:

```
        acquire()
        {
                while(!available);/*busywait*/
                available = false;;
        }
```

 The definition of release() is as follows:

```
                release()
                {
                        available=true;
                }
```

 Calls to either acquire() or release() must be performed atomically. Thus, mutex locks are oftenimplementedusingone ofthehardwaremechanisms.

Disadvantageoftheimplementationgivenhereisthatitrequiresbusywaiting. must

 While a process is in its critical section, any other process that tries to enter its critical section loop continuously in the call to acquire().

 In fact, this type of mutex lock is also called a spinlock because the process “spins” while waitingforthelocktobecome available.

 This continual looping is clearly a problem in a real multiprogramming system, where a single CPU is shared among many processes. Busy waiting wastes CPU cycles that some other process might be able to use productively.

SEMAPHORES

 A semaphore S is an integer variable that, apart from initialization, is accessed only through twostandardatomic operations:

wait()and
signal().

 The wait() operation was originally termed P (from the Dutch proberen, “to test”); signal() wasoriginallycalledV(fromverhogen,“toincrement”).

 The definition of wait() is as follows:

```
wait(S)
{
while(S<=0);//busywait S--
;
}
```

 The definition of signal() is as follows:

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

SemaphoreUsage

 The value of a counting semaphore can range over an unrestricted domain. The value of a binarysemaphorecanrangeonlybetween0and1.

 Binary semaphores behavesimilarlytomutexlocks.

 On systems that do not provide mutex locks, binary semaphores can be used instead for providing mutual exclusion.

 Counting semaphores can be used to control access to a given resource consisting of a finite numberofinstances.

 The semaphore is initialized to the number of resources available.

 Each process that wishes to use a resource performs a**wait()** operationon thesemaphore (thereby decrementing the count).

 When a process releases a resource, it performs a**signal()**operation (incrementing the count).

When the count for the semaphore goes to 0, all resources are being used. After that, processesthatwishto usearesourcewillblockuntil thecountbecomesgreaterthan0

 We can also use semaphores to solvevarioussynchronization problems.

 For example, consider two concurrently running processes: P1 with a statement S1 and P2 with a statement S2. Suppose we require that S2 be executed only after S1 has completed. We can implement this scheme readily by letting P1 and P2 share a common semaphore synch, initialized to 0. In process P1, we insert the statements

```
S1;
signal(synch);
```

InprocessP2,weinsertthestatements

```
wait(synch);
S2;
```

 Because synch is initialized to 0, P2 will execute S2 only after P1 hasinvokedsignal(synch), which is after statement S1 has been executed.

SemaphoreImplementation

 To overcome the need for busy waiting, we can modify the definition of the wait() and signal() operations as follows: When a process executes the wait() operation and finds that the semaphore value is not positive, it must wait.

□ Rather than engaging in busy waiting, the process can block itself.

□ The block operation places a process into a waiting queue associated with the semaphore, and thestateoftheprocessisswitched tothewaiting state.

□ Then control is transferred to the CPU scheduler, which selects another process to execute.
□ A process that is blocked, waiting on a semaphore S, should be restarted when some other processexecutesasignal()operation.

□ The process is restarted by a wakeup() operation, which changes the process from the waiting state to the ready state.
□ The process is then placed in the ready queue. (The CPU may or may not be switched from therunningprocesstothenewlyreadyprocess,dependingontheCPU-scheduling algorithm.)

□ To implement semaphores under this definition, we define a semaphore as follows:

```
typedefstruct
{
intvalue;
structprocess *list;
}semaphore;
```

□ Each semaphore has an integer value and a listofprocesseslist.

□ When a process must wait on a semaphore, it is added to the list of processes.
A signal() operation removesone process from the list of waiting processes and awakens thatprocess.

□ The wait() semaphore operation can be defined as

```
wait(semaphore*S)
{
S->value--;
if(S->value< 0)
{
addthisprocesstoS->list;
block();
}
}
```

□ The signal() semaphore operation can be defined as

```
signal(semaphore*S)
{
S->value++;
if(S->value<=0)
{
removeaprocessPfromS->list; wakeup(P);
}
```

}

 Theblock()operationsuspendstheprocess thatinvokesit.

 The wakeup(P) operation resumes the execution of a blocked process P.

Deadlocksand Starvation

 The implementation of a semaphore with a waiting queue may result in a situation where two ormoreprocessesarewaitingindefinitelyforaneventthatcanbecausedonlybyoneofthe waiting processes
.

 When such a state is reached, these processes are said to be deadlocked

 To illustrate this, consider a system consisting of two processes, P0 andP1,eachaccessing twosemaphores,SandQ,settothevalue 1:

```
P0                  P1
wait(S);            wait(Q);
wait(Q);            wait(S);
. .                  ..
. .                  ..
. .                 ..
signal(S);          signal(Q);
signal(Q);          signal(S);
```

 Suppose that P0 executes wait(S) and then P1 executes wait(Q).When P0executeswait(Q), it mustwaituntilP1executessignal(Q).

 Similarly, when P1 executes wait(S), it must wait until P0 executes signal(S).

 Since these signal() operations cannot be executed, P0 and P1 are deadlocked.

 We say that a set ofprocessesisinadeadlockedstatewheneveryprocessinthesetiswaiting for an event that can be caused only by another process in the set.

 Another problem related to deadlocks is indefinite blocking or starvation, a situation in which processeswaitindefinitelywithinthesemaphore. semaphore

 Indefinite blocking may occur if we remove processes from the list associated with a in LIFO (last-in, first-out) order.

PriorityInversion

 A scheduling challenge arises when a higher-priorityprocessneedstoreadormodifykernel data that are currently being accessed by a lower-priority process—or a chain of lower-priority processes.

The kerneldataaretypically protectedwitha lock,the higher-priority processwillhave to wait for a lower-priority one to finish with the resource.

□ The situation becomes more complicated if the lower-priorityprocessispreemptedinfavor ofanotherprocess with ahigher priority.

□ This problem is known as priority inversion. It occurs only in systems with more than two priorities, so one solution is to have only two priorities.

□ Typically these systems solve the problem by implementing a priority-inheritanceprotocol. Accordingtothisprotocol,allprocessesthatareaccessingresourcesneededbyahigher-priority process inherit the higher priority until they are finished with the resources in question.

□ When they are finished, their priorities revert to their original values. In the example above, a priority-inheritance protocol would allow process L to temporarily inherit the priority of process.

CLASSICPROBLEMSOFSYNCHRONIZATION

1. BoundedBufferProblem
2. ReaderWriterProblem
3. DiningPhilosopher'sProblem

TheBounded-BufferProblem

□ We assume that the pool consists of n buffers, each capableofholdingoneitem.Themutex semaphoreprovidesmutualexclusionforaccessestothebufferpoolandisinitializedtothevalue 1.

□ The empty and full semaphores count the number of empty and full buffers.
□ The semaphore empty is initialized to thevalue n.

□ The semaphore full is initialized to the value 0.

Theproducerandconsumerprocessessharethefollowingdatastructures: int

```
n;
semaphoremutex=1;
semaphoreempty=n;
semaphore full = 0
```

Thestructureoftheproducerprocess.

```
do {
. . .
/*produceanitemin nextproduced*/
. . .
wait(empty);
wait(mutex);
. . .
/*addnextproducedto thebuffer*/
. . .
signal(mutex);
signal(full);
```

```
        }while(true);
```

Thestructureoftheconsumerprocess.

```
        do {
        wait(full);
        wait(mutex);
        . . .
        /*removeanitemfrombuffertonextconsumed*/
        . . .
        signal(mutex);
        signal(empty);
        . . .
    /*consumetheitemin nextconsumed */
    . . .
    }while(true);
```

 We can interpret this code as the producer producing full buffers for the consumer or as the consumerproducingemptybuffersfortheproducer.

ReaderWriterProblem

TheR-Wproblemisanotherclassicproblemforwhichdesignofsynchronizationandconcurrency mechanisms can be tested. The producer/consumer is another such problem; the dining philosophers is another.

Definition

- Thereisadataareathat issharedamonganumberof processes.
- Anynumberofreadersmaysimultaneouslywriteto the data area.
- Onlyone writerat atimemaywriteto thedataarea.
- Ifawriteriswritingtothedataarea,noreadermayread it.
- Ifthereisatleastonereaderreadingthedata area,nowritermaywriteto it.
- Readersonlyread andwritersonlywrite
- Aprocessthatreadsandwritestoadataareamustbeconsideredawriter(consider producer or consumer)

In the solution to the first readers–writers problem, the reader processes share the following data structures:

```
        semaphorerwmutex=1;
        semaphore mutex = 1;
        intread count=0;
```

 The semaphores mutex and rw mutex are initialized to 1; read count is initialized to 0.
 The semaphore rw mutex is common to both reader and writer processes.
 The mutex semaphore is used to ensure mutual exclusion when the variable read count is updated.
 The read count variable keeps track of how many processes are currently reading the object.
 The semaphore rw mutex functions as a mutual exclusion semaphore for the writers.

Thestructureofawriterprocess. do

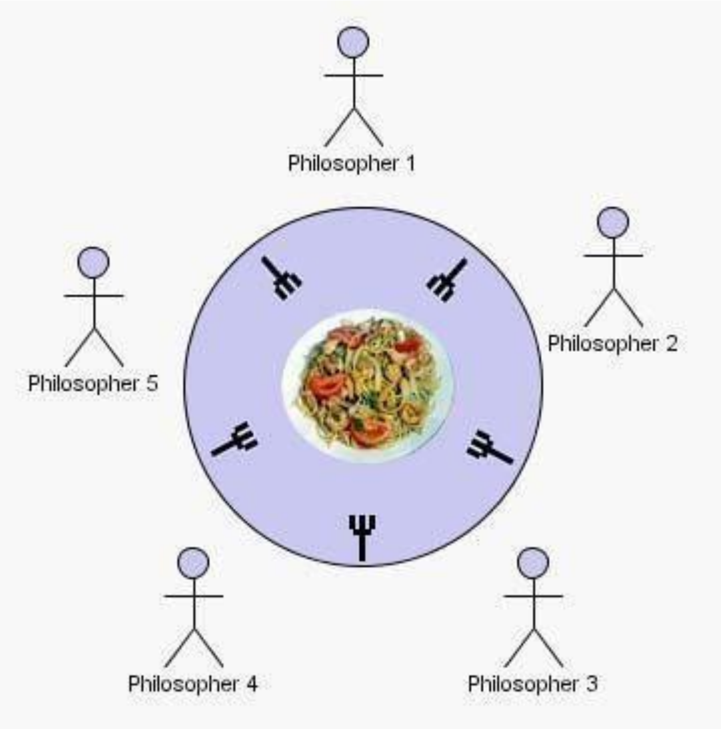
```
    {
    wait(rwmutex);
    . . .
    /*writingisperformed*/
    . . .
    signal(rwmutex);
    }while(true);
```

Thestructureofareaderprocess. do

```
        {
        wait(mutex);
        readcount++;
        if(readcount==1)
        wait(rw mutex);
        signal(mutex);
        . . .
        /*readingisperformed*/
wait(mutex);
readcount--;
if(readcount==0) signal(rw
mutex); signal(mutex);
}while(true);
```

<u>DiningPhilosophersProblem</u>

Consider there are five philosophers sitting around a circular dining table. The dining table hasfive chopsticks and a bowl of rice in the middle.



Atanyinstant,aphilosopheriseithereatingorthinking.Whenaphilosopherwantstoeat,he uses two chopsticks - one from their left and one from their right.

Whenaphilosopherwants to think,hekeepsdown both chopsticks at theiroriginalplace.

- Whenaphilosopher thinks, hedoesnotinteractwithhis others.
- From time to time, a philosopher gets hungry and tries to pick up the two forks that areclosest to him (the forks that are between him and his left and right neighbors).
- A philosopher may pick up only one fork at a time. Obviously, he cannot pick up a fork that is already in the hand of a neighbor.
- When a hungryphilosopher has both his forks at the same time, he eats without releasing his forks.
- Whenheis finishedeating, heputs downboth ofhis forksand startsthinking again.

Solution:

From the problem statement, it is clear that a philosopher can think for an indefinite amount of time. But when a philosopher starts eating, he has to stop at some point of time. The philosopher is in an endless cycle of thinking and eating.

Anarrayoffivesemaphores,**stick[5]**,foreachofthefivechopsticks. The

code for each philosopher looks like:

```
while(TRUE){
wait(stick[i]);
wait(stick[(i+1)%5]);//mod isusedbecauseifi=5, next
                //chopstickis 1(diningtableis circular)
/* eat */
signal(stick[i]);
signal(stick[(i+1)%5]);
/* think */
}
```

Whenaphilosopherwantstoeattherice,hewillwaitforthechopstickathisleftandpicksupthat chopstick.Thenhewaitsfortherightchopsticktobeavailable,andthenpicksittoo.Aftereating, he puts both the chopsticks down.

But if all five philosophers are hungry simultaneously, and each of them pickup one chopstick, then a deadlock situation occurs because they will be waiting for another chopstick forever.

Thepossiblesolutions forthis are:

1) A philosopher must be allowed to pick up the chopsticks only if both the left and right chopsticks are available.

2) Allowonlyfourphilosopherstosit atthetable.Thatway,ifallthefourphilosopherspick up four chopsticks, there will be one chopstick left on the table. So, one philosopher can start eating and eventually, two chopsticks will be available. In this way, deadlocks canbe avoided.

MONITORS

Definition: Monitoris ahigh-level languageconstruct with acollectionof procedures,variables, and data structures that are all grouped together in a special kind of module or package.

 Processes may call the procedures in a monitor whenever they want to, but they cannot directly access the monitor's internal data structures from procedures declared outside the monitor.

 Monitors have an important property that makes them useful for a chievingmutualexclusion: onlyoneprocesscan beactivein a monitorat anyinstant.

MonitorUsage

 A monitor type presents a set of programmer-definedoperationsthatareprovidedmutual exclusion within the monitor.

 The monitor type also contains the declaration of variables whose values define the state of an instance of that type, along with the bodies of procedures or functions that operate on those variables.

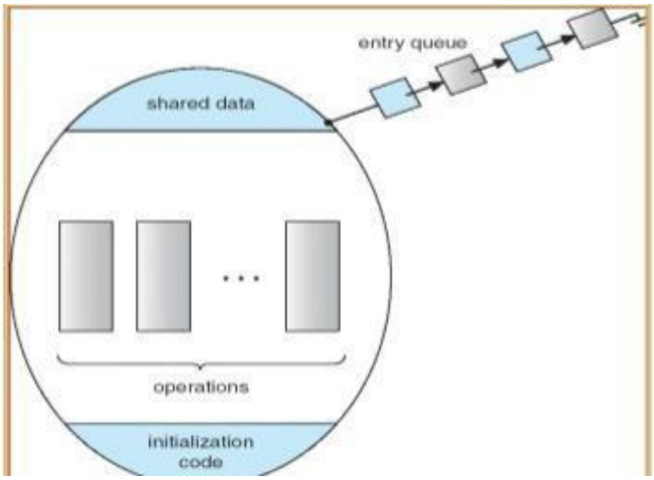
```
monitormonitor name
{
/*sharedvariabledeclarations*/
function P1 ( . . . ) {
. . .
}
functionP2 ( ... ) {
. . .
}
.
.
functionPn ( ... ) {
. . .
}
initializationcode(.. .) {
. . .
}
}
```

 The representation of a monitor type cannot be used directly by the various processes. Thus, a procedure defined within a monitor can accessonly those variables declared locally within themonitor and its formal parameters.

 Similarly, the local variablesofamonitorcan beaccessed byonlythe local procedures.

- The monitor construct ensures that only one process at a time can be active within the monitor.

Schematicviewof aMonitor



Themonitorconstructisnotsufficientlypowerfulformodelingsomesynchronizationschemes.
 For this purpose, we need to define additional synchronization mechanisms. These mechanisms are provided by the condition construct condition x, y;

Theonlyoperationsthatcanbeinvokedonaconditionvariablearewait()andsignal().The operation

```
        x.wait();
```

means that the process invokingthis operation is suspended until anotherprocess invokes

```
        x.signal();
```

Thex.signal()operationresumesexactlyonesuspendedprocess.

Amonitorsolutiontothedining-philosopherproblem.

```
monitorDiningPhilosophers
{
enum{THINKING,HUNGRY,EATING}state[5];
conditionself[5];
voidpickup(inti)
{
state[i]=HUNGRY;
test(i);
if(state[i]!=EATING)
self[i].wait();
}
voidputdown(int i)
{
state[i]=THINKING;
test((i + 4) % 5);
test((i+1)%5);
}
void test(int i)
{
if((state[(i+4)%5]!=EATING)&&(state[i]==HUNGRY)&&(state[(i+1)%5]!= EATING))
{
state[i]=EATING;
self[i].signal();
}
}
initializationcode()
{
for(inti=0;i<5;i++) state[i]
= THINKING;
}
```

CPUSCHEDULING

CPUschedulingisthebasisofmulti-programmedoperatingsystems.

ByswitchingtheCPUamongprocesses,theoperatingsystemcanmakethecomputermore productive.

BasicConcepts

- Theobjectiveofmulti-programmingistohavesomeprocessrunningatalltimes,to maximize CPU utilization.
- ForaUni-processorsystem,therewillneverbemorethanonerunningprocess.
- Schedulingisafundamentaloperatingsystemfunction.
- Theideaof multi-programmingis to execute aprocess until it must wait, typicallyfor the completion of some I/O request.
- TheCPUisoneoftheprimarycomputerresources.
- TheCPUschedulingiscentraltooperatingsystemdesign.

CpuScheduler

- When the CPU becomes idle, the operating system must select on of the processes in the ready queue to be executed.
- Theselectionprocessiscarriedoutbytheshort-termscheduler(CPUscheduler)
- Theschedulerselects from amongthe processes in memorythat arereadyto execute, and allocates the CPU to one of them.
- AreadyqueuemaybeimplementedasaFIFOqueue,apriorityqueue, atreeorsimplyan unordered link list.
- AlltheprocessesinthereadyqueuearelinedupwaitingforachancetorunontheCPU.

CPUschedulingdecisionsmaytakeplacewhenaprocess.

1. Switchesfromrunningtowaiting state
2. Switchesfromrunningtoreadystate
3. Switchesfromwaitingto ready

4. Terminates

Schedulingunder1 and4is **nonpreemptive.**
Allotherschedulingis **preemptive.**

NonpreemptiveScheduling➔Aschedulingdisciplineisnonpreemptiveif,onceaprocesshas been given the CPU, the CPU cannot be taken away from that process.

PreemptiveScheduling➔ Aschedulingdisciplineispreemptiveif,onceaprocesshasbeen given the CPU can taken away.

Dispatcher

DispatcherisamodulethatgivescontroloftheCPUtotheprocessselected bytheshort-term scheduler. This function involves the following:

□ switching context.

□ switching to user mode.

□ jumping to the proper location in the user program to restartthat program.

Dispatchlatency–Thetimetakenforthedispatchertostoponeprocessandstartanother running.

Schedulingcriteria

1. **CPUutilization**–keeptheCPUasbusyaspossibleThroughput–#ofprocessesthat complete their execution per time unit .
2. **Turnaroundtime**–amountoftimetoexecuteaparticularprocess
3. **Waitingtime**–amount oftimeaprocess hasbeenwaitingin thereadyqueue
4. **Responsetime**–amountoftimeittakesfromwhenarequestwassubmitteduntilthe first response is produced, not output (for time-sharing environment)
5. **Throughput**–Thenumberofprocessesthatcompletetheirexecutionpertimeunit.

BestAlgorithmconsider following:

□ Max CPU utilization
□ Max throughput
□ Min turnaround time
□ Min waiting time **Minresponsetime**
□

Formulas to calculate Turn-around time & waiting time is:
Waitingtime=FinishingTime–(CPUBursttime+ArrivalTime)
Turnaround time = Waiting Time + Burst Time

SchedulingAlgorithms

AProcessSchedulerschedulesdifferentprocessestobeassignedtotheCPUbasedonparticular scheduling algorithms.

1. First-Come,First-Served(FCFS)Scheduling
2. Shortest-Job-First(SJF) Scheduling
3. PriorityScheduling
4. Round Robin(RR)Scheduling

First-Come,First-Served(FCFS)Schedulingalgorithm.

□ This is the simplest CPU -schedulingalgorithm.

□ According to this algorithm, the process that requests the CPU first is allocated the CPUfirst.

□ The implementation of FCFS is easily managed with a FIFO queue.

□ When a process enters the ready queue,itsPCBislinkedontothetailofthequeue.

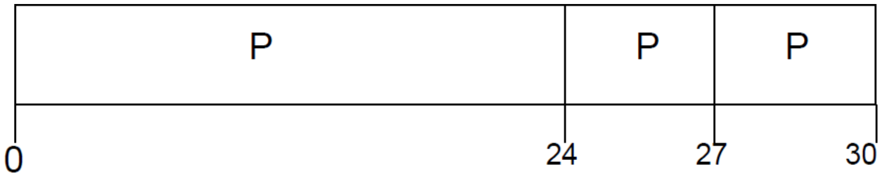
□ When the CPU is free, it is allocated to the process at the head of the queue. The runningprocessis thenremovedfromthequeue.

ExampleProblem

Considerthefollowingsetofprocessesthatarriveattime0,withthelengthoftheCPUburst time given in milliseconds:

Process	**Burst Time(ms)**
P1	24
P2	3
P3	3

Supposethattheprocesses arriveintheorder:P1,P2 ,P3 TheGantt Chart:



Waiting time

□ Waiting time for P1 = 0; P2 = 24; P3 = 27

□ Average waiting time:(0 +24 +27)/3=17ms.

TurnaroundTime=WaitingTime+Burst Time

□ Turnaround Time for P1 = (0+24)=24; P2 = (24+3)=27; P3 = (27+3)=30

□ Average Turnaround Time = (24+27+30)/3 = 27 ms

Shortest-Job-First(SJF)Scheduling

□ This algorithm associateswitheachprocessthelengthofitsnextCPUburst.Use these lengths to schedule the process with the shortest time.

□ When the CPU is available, it is assigned to the process that has the smallest next CPUburst.Itisalso calledasshortestnextCPUburst.

□ If two processes have the same length next CPU burst, FCFS scheduling is used tobreakthetie.

Process	BurstTime
P1	24
P2	3
P3	3

GanttChart

P2	P3	P1

0 3 6 30

Waiting time

ForP1=6,P2=0,P3=3

Average Waiting Time=(6+0+3)/3=3 ms.

TurnaroundTime=WaitingTime+BurstTime

TurnaroundTimeforP1=(6+24)=30,P2=(0+3)=3,P3=(3+3)=6

AverageTurnaroundTime=(30+3+6)/3=13ms

PriorityScheduling

- TheSJFalgorithm isaspecial caseofthegeneralpriority-schedulingalgorithm.
- Aprioritynumber(integer)isassociatedwitheachprocessandtheCPUisallocatedto the process with the highest priority.
- Equal-priorityprocessesarescheduledinFCFSorder.
- TheCPUisallocatedtotheprocesswiththehighestpriority(smallestintegerºhighest priority) .

Process	BurstTime	Priority
P1	24	2
P2	3	1
P3	3	3

GanttChart

P2	P1	P3

0 3 27 30

Waiting time

ForP1=3,P2=0,P3=27

Average Waiting Time=(3+0+27)/3=10ms

TurnaroundTime=WaitingTime+BurstTime

TurnaroundTimeforP1=(3+24)=27,P2=(0+3)=3,P3=(27+3)=30

AverageTurnAround Time=(27+3+30)/3=20ms.

Roundrobinscheduling

- Roundrobinschedulingisdesignedespeciallyfortime-sharing systems.
- ItissimilartoFCFS,butpreemptionisaddedtoswitchbetween processes.
- EachprocessgetsasmallunitofCPUtimecalledatimequantumor

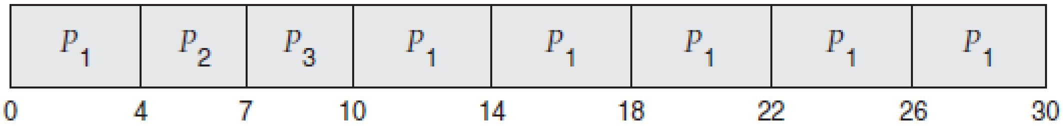
timeslice.

- To implement RR scheduling, the ready queue is kept as a FIFO queue of processes. New processes are added to the tail of the ready queue. The CPU scheduler picks the first process from the ready queue, sets a timer to interrupt after 1 time quantum and dispatches the process.

- If the CPU burst time is less than the time quantum, the process itself will release the CPUvoluntarily.Otherwise,iftheCPUburstofthecurrentlyrunningprocessislonger thanthetimequantumacontextswitchwillbeexecutedandtheprocesswillbe putat the tail of the ready queue.

Process	**Burst Time**
P_1	24
P_2	3
P_3	3

GanttChart



Waiting time

Average waiting time = [6+4+7]/3=17/3=5.66

Turnaround Time = Waiting Time + Burst Time

TurnaroundTimeforP1=(6+24)=30,P2=(4+3)=7,P3=(7+3)=10

AverageTurnaroundTime=(30+7+10)/3=15.6ms.

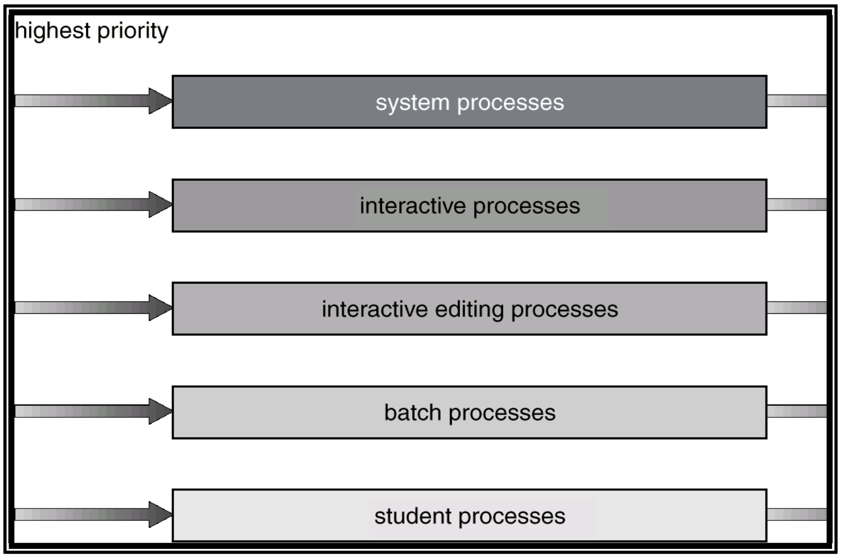
MultilevelQueueScheduling

- Itpartitionsthereadyqueueintoseveralseparatequeues.
- The processes are permanently assigned to one queue, generally based onsome propertyoftheprocess,such asmemorysize,process priority,or processtype.
- There must be scheduling between the queues, which is commonly implemented as a fixed-priority preemptive scheduling.
- For example the foreground queue may have absolute priorityover the background queue.

Example:ofamultilevel queueschedulingalgorithmwithfivequeues

1. Systemprocesses
2. Interactiveprocesses
3. Interactiveeditingprocesses
4. Batchprocesses
5. Studentprocesses

Eachqueuehasabsolutepriorityoverlower-priorityqueue.



MultilevelFeedbackQueueScheduling

□It allows a process to move between queues.

□The idea is to separate processes with different CPU-burst characteristics.

□IfaprocessusestoomuchCPUtime,itwillbemovedtoalower-priority queue.

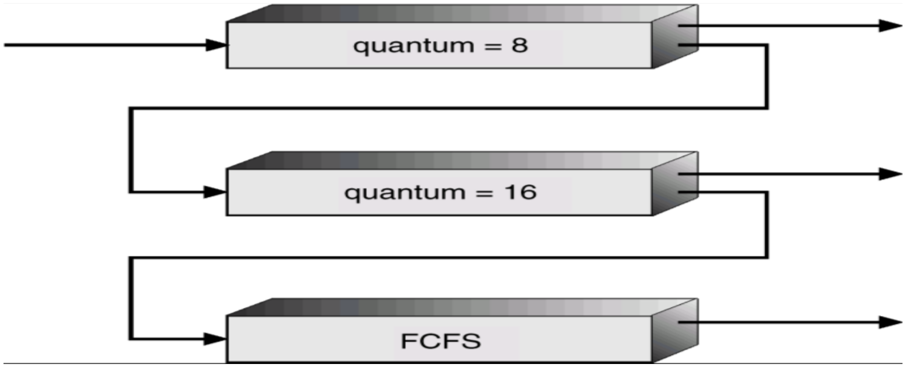
□This scheme leaves I/O-bound and interactive processes in the higher- priority queues.

□Similarly,aprocessthatwaitstoolonginalowerpriorityqueuemaybe moved to a higher-priority queue.

□Thisformofagingpreventsstarvation.

Example:

□ Consider a multilevel feedback queue scheduler with three queues,numbered from 0 to 2 .



□ The scheduler first executes all processesinqueue 0.

□ Onlywhen queue0isemptywillitexecuteprocessesin queue1.

□

□ Similarly,processes in queue2 will be executed only if queues0and1areempty. Aprocessthatarrives forqueue1 willpreempt aprocessinqueue2.

□ Aprocessthatarrivesforqueue0will,inturn,preemptaprocessin queue1.

- A multilevel feedback queue scheduler is define dbythe following parameters:

1. Thenumberofqueues
2. Theschedulingalgorithmforeach queue
3. The method used to determine when to upgrade a process to a higher priority queue
4. The method used to determine when to demote a process to a lower-priority queue
5. The method used to determine whichqueueaprocesswillenter when that process needs service

MultipleProcessorScheduling

- If multiple CPUs are available, the scheduling problem is correspondinglymore complex.
- Ifseveralidenticalprocessorsareavailable,thenload-sharingcanoccur.
- It is possible to provide a separate queue for each processor.
- In this case however, one processor could be idle, with an emptyqueue,whileanother processor was very busy.
- Topreventthissituation,weuseacommonreadyqueue.
- All processes go intoonequeueandarescheduledontoanyavailableprocessor.
- In such a scheme, one of two scheduling approaches may be used.

1. **Self Scheduling** - Each processor is self-scheduling. Each processor examines the common ready queue and selects a process to execute.Wemustensurethattwoprocessorsdonotchoosethesameprocess, and that processes are not lost from the queue.
2. **Master – Slave Structure** - This avoids the problem by appointing one processor as scheduler for the other processors, thus creating a master-slave structure.

Real-TimeScheduling

- Real-time computing is divided into two types.

1. Hardreal-timesystems
2. Softreal-timesystems

Hardreal-timesystems

- Hard RTS are required to complete a criticaltaskwithinaguaranteed amount of time.
- Generally, a process is submitted along with a statement of the amountof time in which it needs to complete or perform I/O.
- Theschedulertheneitheradmitstheprocess,guaranteeingthatthe process will complete on time, or rejects the request as impossible. This is known as **resource reservation**.

Softreal-timesystems

- Softreal-timecomputingislessrestrictive.Itrequiresthatcriticalprocessesrecieve priorityoverlessfortunateones.

The system must have priority scheduling, and real-time processes must have the highest priority.

 Thepriorityofreal-timeprocessesmustnotdegradeovertime,eventhoughthe priority of non-real-time processes may.

 Dispatch latency must be small. The smaller the latency, the faster areal-time process can start executing.

The high-priorityprocess would be waiting for a lower-priority one to finish. Thissituationisknownas**priority inversion**.

DEAD LOCK

Definition:

A process request resources, if the resources are not available at that time, the process enters in to a wait state. It may happen that waiting processes will never again change the state, because theresourcestheyhaverequestedareheldbyotherwaitingprocesses. *Thissituationiscalledas* ***dead lock.***

SystemModel

 A system consists ofafinitenumberofresources tobe distributedamong anumber of competingprocesses.

 The resources may be partitioned into several types (or classes), each consisting of somenumberofidenticalinstances.

 CPUcycles, files,and I/O devices(suchasprintersandDVDdrives)areexamplesof resource types.

Aprocessmustrequest aresourcebeforeusingitandmustreleasetheresourceafterusingit.

Underthenormalmodeofoperation,aprocessmayutilizearesourceinonlythefollowing sequence:

1. **Request**. The process requests the resource. If the request cannot be granted immediately then the requesting process must wait until it can acquire the resource.
2. **Use**.Theprocesscanoperateonthe resource
3. **Release**.Theprocessreleases the resource.

DeadlockCharacterizations:-

Inadeadlock,processesneverfinishexecuting,andsystemresourcesaretiedup,preventing other jobs from starting.

NecessaryConditionsforDeadlock:-

Adeadlocksituation canariseifthefollowingfourconditions hold simultaneouslyinasystem.

1) MUTUALEXCLUSION:-Atleastoneresourcemustbeheld inaon-sharablemode.i.eonly one process can hold this resource at a time . other requesting processes should wait till it is released.

2) HOLD&WAIT:-theremustexistaprocessthatisholdingatleastoneresourceandiswaiting to acquire additional resources that are currently being held by other processes.

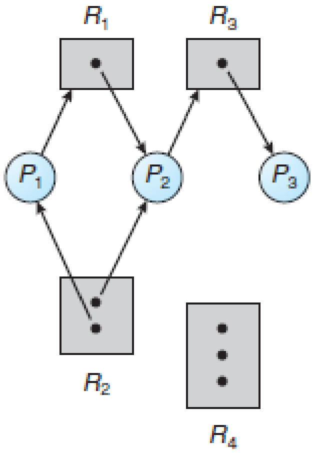
3) NO PREEMPTION:- Resources cannot be preempted, that is a resource can be released voluntarily by the process holding it, after that process has completed its task.

4) CIRCULAR WAIT:- There must exist a set {p0,p1,p1….pn} of waiting processes such that p0iswaitingforaresourcethatisheldbythep1, p1iswaitingfortheresourcethatisheldbythe p2…. And so on. pn is waiting for a resource that is held by the p0.

Resource-AllocationGraph

A deadlock can be described in terms of a directed graph called system resource-allocationgraph.

• AsetofverticesVandasetofedgesE. – V
is partitioned into two types:
7P={P1,P2,…,Pn}, thesetconsistingof all theprocesses inthesystem.
7R ={R1,R2,…, Rm},theset consistingof allresource types inthesystem.
– request edge– directededgePi→ Rj
– assignmentedge–directed edgeRj→ P



Theresource-allocation graphdepicts thefollowingsituation.

Thesets P, R, and E:

- □ P = {P1, P2, P3}
- □ R = {R1, R2, R3, R4}
- □ E = {P1 → R1, P2 → R3, R1 → P2, R2 →P2, R2 → P1, R3 → P3}

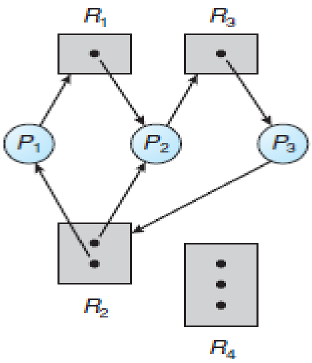
Resourceinstances:

- □ One instance of resource type R1
- □ Two instances of resource type R2
- □ One instance of resource type R3
- □ Three instances of resource type R4

Processstates:

- □ Process P1 is holding an instance of resou rcetypeR2andiswaitingforaninstanceof resourcetypeR1.
- □ Process P2 is holding an instance of R1 and an instance of R2 and is waiting for an instanceofR3.
- □ Process P3 is holding an instance of R3.

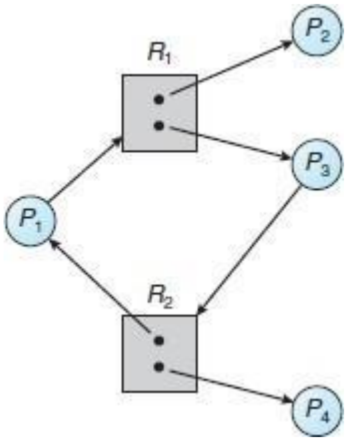
Resource-allocationgraphwithadeadlock.



- ProcessesP1,P2,andP3aredeadlocked.ProcessP2iswaitingfortheresourceR3,which is held by process P3. Process P3 is waiting for either process P1 or process P2 to release resource R2. In addition, process P1 is waiting for process P2 to release resource R1.
- Wealso haveacycle:P1 → R1 → P3 → R2 →P1
- Ifthegraphcontainsnocycles,thennoprocessin thesystemis deadlocked.

Ifthegraphdoescontainacycle,thenadeadlockmayexist.

Resource-allocation graph with a cycle but no deadlock.



MethodsforHandlingDeadlocks

Wecandealwiththedeadlock problemin oneofthreeways:

1. Wecanuseaprotocoltopreventoravoiddeadlocks,ensuringthatthesystemwill never enter a deadlocked state
2. Wecanallow thesystemto enteradeadlockedstate,detect it, and recover.
3. We canignore theproblemaltogether andpretend thatdeadlocksneveroccur inthe system. Thethirdsolutionistheoneusedbymostoperatingsystems,includingLinux and Windows.

□**Deadlockprevention**providesasetofmethodstoensurethatatleastoneofthe necessary conditions cannot hold.

□**Deadlock avoidance** requiresthattheoperatingsystembegivenadditionalinformation inadvanceconcerningwhichresourcesaprocesswill requestanduseduringits lifetime.

DEADLOCKPREVENTION

□ For a deadlock to occur, each of the four necessary conditions must hold.

□ By ensuring that at least one of these conditions cannot hold, we can prevent the occurrence of adeadlock.

1. MutualExclusion

– notrequiredforsharableresources; mustholdfornon-sharable resources.

– Forexample,aprinter cannotbesimultaneouslysharedbyseveralprocesses.

– Aprocessneverneedstowait forasharable resource.

2. Holdand Wait

– must guarantee that whenever a process requests a resource, it does not hold any other resources.

– Oneprotocol requires eachprocess to request and beallocated all its resourcesbeforeit begins execution,

– Oranotherprotocolallowsaprocesstorequestresourcesonly whentheprocesshas none. So, before it can request any additional resources, it must release all the resources that it is currently allocated.

3. DenyingNopreemption

– Ifaprocessthatisholdingsomeresourcesrequestsanotherresourcethatcannotbe immediately allocated to it, then all resources currently being held are released.

– Preemptedresources areaddedtothelist ofresourcesforwhichtheprocessiswaiting.

– Process will be restarted only when it can regain its old resources, as well as the newones that it is requesting.

4. DenyingCircularwait

- Imposeatotalorderingofallresourcetypesandalloweachprocessto request for resources in an increasing order of enumeration.
- Let R = {R1,R2,...Rm} be the set of resource types.
- Assigntoeachresourcetypeauniqueintegernumber.
- IfthesetofresourcetypesRincludestapedrives,diskdrivesandprinters.

  F(tapedrive)=1,
  F(diskdrive)=5,
  F(Printer)=12.
- Eachprocesscanrequestresourcesonlyinanincreasingorderof enumeration.

DEADLOCKAVOIDANCE

- An alternative method for avoiding deadlocks is to require additional information abouthowresources aretoberequested.
- Each request requires that in making this decision the system consider
  - theresourcescurrentlyavailable,
  - theresourcescurrentlyallocatedtoeachprocess,
  - thefuturerequests andreleasesofeachprocess.

Adeadlock-avoidancealgorithmdynamicallyexaminestheresource-allocationstateto ensure that a circular-wait condition can never exist.

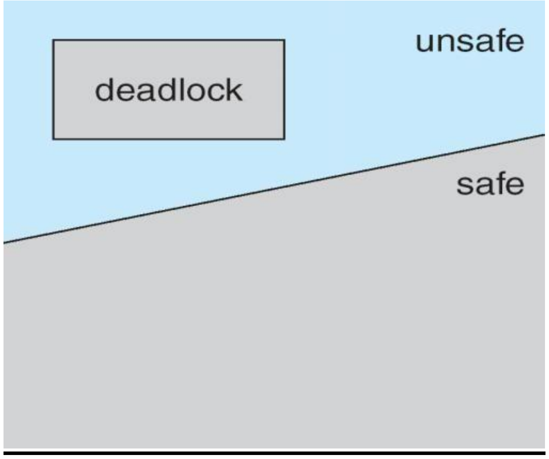
Theresource-allocationstateisdefinedbythenumberofavailableand allocated resources and the maximum demands of the processes.

SafeState

• When a process requests an available resource, system must decide if immediate allocation leaves the system in a safe state.

• System is in safe state if there exists a sequence <P1, P2, …, Pn> of ALL the processes is the systems such that for each Pi, the resources that Pi can still request can be satisfied by currently available resources + resources held by all the Pj, with j < i.

- Thatis:
  - – IfPiresourceneedsarenotimmediatelyavailable,thenPicanwaituntilallPj have finished.
  - – WhenPjisfinished,Picanobtainneededresources,execute,returnallocated resources, and terminate.
  - – WhenPiterminates,Pi+1canobtainitsneededresources,andsoon.



Banker'sAlgorithm

 The resource-allocation-graphalgorithmisnotapplicabletoaresourceallocation system with multiple instances of each resource type.

 The name was chosen because the algorithm could be used in a banking system to ensurethatthebankneverallocateditsavailablecashinsuchawaythatitcouldno longer satisfy the needs of all its customers.

Multipleinstances.

Eachprocessmust apriori claimmaximumuse.

- ➢ Whenaprocessrequestsaresourceitmayhavetowait.
- ➢ Whenaprocessgets all its resourcesitmustreturnthem inafiniteamountoftime.
- ➢ Letn=numberofprocesses, andm=numberofresources types.
  1. **Available:**indicatesthenumberofavailableresourcesofeachtype.
  2. **Max:**Max[i,j]=kthenprocessPimayrequestatmostkinstancesof resource type Rj
  3. **Allocation:**Allocation[i.j]=k,thenprocessPiiscurrentlyallocated K instances of resource type Rj
  4. **Need:**ifNeed[i,j]=kthenprocessPimayneedKmoreinstancesof resource type Rj ,Need [i, j]=Max[i, j]-Allocation[i, j]

Need[i,j]= Max[i,j]–Allocation [i,j].

Safetyalgorithm

```
1. Initializework:=availableandFinish[i]:=falsefori=1,2,3.. n
2.Findanisuchthat both
    a. Finish[i]=false
    b. Needi<=Work        ifnosuch iexists, gotostep 4
3. work:=work+allocationi; Finish[i]:=true        goto step 2
```

4. Iffinish[i]=trueforall i,then thesystemisinasafestate

Example:

GiventhefollowingstatefortheBanker'sAlgorithm.

5 processesP0throughP4

3resourcetypesA(6instances),B(9instances)andC(5instances).

Snapshot at time T0:

	Max	Allocation
	A B C	A B C
P_0	6 7 3	1 1 1
P_1	2 2 2	1 1 2
P_2	2 6 3	0 3 0
P_3	2 2 2	2 1 1
P_4	4 6 3	1 1 1

a) Calculatetheavailablevector.

b) CalculatetheNeedmatrix.

c) Isthesysteminasafestate? Ifso,showonesequenceofprocesseswhichallows the system to complete. If not, explain why.

d) Giventherequest(1,2,0)fromProcessP2.Shouldthisrequestbegranted?Why or why not?

a) Calculatetheavailablevector.

Available
A B C
1 2 0

b) CalculatetheNeedmatrix.

	Need A B C
P_0	5 6 2
P_1	1 1 0
P_2	2 3 3
P_3	0 1 1
P_4	3 5 2

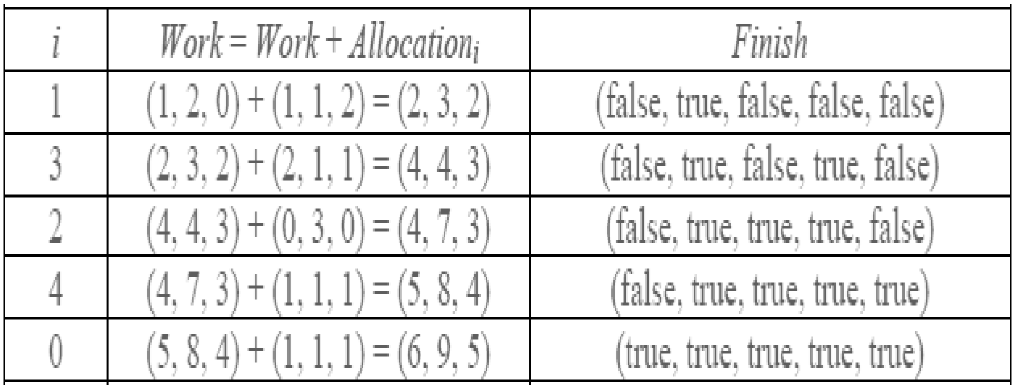
c) Isthesysteminasafestate? Ifso,showonesequenceofprocesseswhichallows the system to complete. If not, explain why.

1. Initializethe *Work*and*Finish*vectors.

Work=*Available*=(1,2,0)

Finish=(false,false,false,false,false)

2. Findindex*i*suchthat*Finish*[*i*]=false and*Needi* <=*Work*



i	$Work = Work + Allocation_i$	*Finish*
1	(1, 2, 0) + (1, 1, 2) = (2, 3, 2)	(false, true, false, false, false)
3	(2, 3, 2) + (2, 1, 1) = (4, 4, 3)	(false, true, false, true, false)
2	(4, 4, 3) + (0, 3, 0) = (4, 7, 3)	(false, true, true, true, false)
4	(4, 7, 3) + (1, 1, 1) = (5, 8, 4)	(false, true, true, true, true)
0	(5, 8, 4) + (1, 1, 1) = (6, 9, 5)	(true, true, true, true, true)

3. Since*Finish*[*i*]=trueforall*i*,hencethesystemisinasafestate.Thesequence of processes which allows the system to complete is P1, P3, P2, P4, P0**.**

d) Giventherequest(1,2,0)fromProcessP2.Shouldthisrequestbegranted?Why or why not?

1. Checkthat*Request2 <=Need2*.

Since(1,2, 0)<=(2,3, 3),hence,thiscondition is satisfied.

2. Checkthat*Request2<=Available*.

Since(1, 2,0)<=(1, 2,0), hence, thisconditionis satisfied.

3. Modifythesystem'sstateasfollows:

Available=Available–Request2 =(1,2,0)–(1,2,0)=(0,0, 0)

Allocation2 =*Allocation2*+*Request2*=(0, 3,0) +(1,2,0)= (1, 5,0)

Need2=*Need2* –*Request2* =(2,3, 3)–(1,2, 0)=(1, 1,3)

4. Applythesafetyalgorithmtocheckifgrantingthisrequestleavesthe system in a safe state.

1. Initializethe*Work*and*Finish*vectors.

Work=*Available*=(0,0,0)

Finish=(false,false,false, false, false)

2. Atthispoint,theredoesnotexistanindex*i*suchthat*Finish*[*i*]=false and *Needi* <= *Work*.

Since*Finish*[*i*] ≠truefor all *i*, hencethesystem isnot inasafe state.

Therefore,thisrequestfrom process*P2*shouldnotbe granted.

Resource-RequestAlgorithm

Let Requesti be the request vector for process Pi . If Requesti [j] == k, then process Pi wants k instances of resource type Rj . When a request for resources is made by processPi , the following actions are taken:

1. IfRequesti≤Needi,gotostep2.Otherwise,raiseanerrorcondition,sincethe process has exceeded its maximum claim.

2. IfRequesti≤Available,gotostep3.Otherwise,Pimustwait,sincethe resources are not available.

3. HavethesystempretendtohaveallocatedtherequestedresourcestoprocessPi by modifying the state as follows:

Available = Available–Requesti ;

Allocationi=Allocationi+Requesti;

Needi=Needi–Requesti;

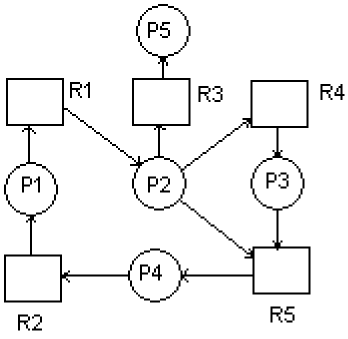
DEADLOCKDETECTION

DeadlockDetection

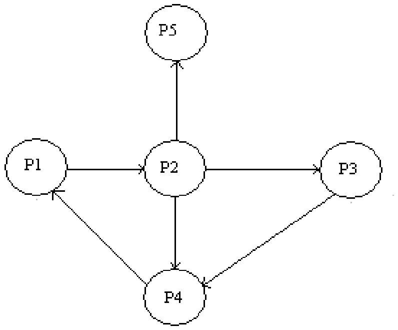
(i) Singleinstanceofeachresourcetype

Ifallresourceshaveonlyasingleinstance,thenwecandefineadeadlock detectionalgorithmthatuseavariantofresource-allocationgraphcalledawaitforgraph.

ResourceAllocationGraph



WaitforGraph



(ii) SeveralInstanceofaresourcetype

Available:Numberofavailableresourcesofeachtype

Allocation:numberofresourcesofeachtypecurrentlyallocatedtoeachprocess

Request: Currentrequestofeachprocess

If Request [i,j]=k, then process $P_i$is requesting K more instances of resource typeRj.

```
1. Initializework:=available
        Finish[i]=false,otherwisefinish[i]:=true
 2. Findanindex isuchthatboth
 a. Finish[i]=false
 b. Request_i<=work
        ifnosuch iexistsgoto step4.
 3. Work:=work+allocation_i
        Finish[i]:=truegotostep2
 4. Iffinish[i]=false    thenprocessP_iisdeadlocked
```

DEADLOCKRECOVERY.

- Therearethreebasicapproachestorecoveryfromdeadlock:
    1. Informthesystemoperator,andallowhim/hertotakemanualintervention.
    2. Terminateoneormoreprocessesinvolvedinthedeadlock
    3. Preemptresources.

1. ProcessTermination

Two basic approaches, both of which recover resources allocated to terminatedprocesses:

➔Terminateallprocessesinvolvedinthedeadlock.Thisdefinitelysolvesthe deadlock, but at the expense of terminating more processes than would be absolutely necessary.

➔Terminateprocessesonebyoneuntilthedeadlockisbroken.Thisismore conservative, but requires doing deadlock detection after each step.

➔Inthelattercasetherearemanyfactorsthatcangointodecidingwhich processes to terminate next:

- Processpriorities.
- Howlongtheprocesshasbeenrunning,andhowcloseitistofinishing.
- How many and what type of resources is the process holding. (Are theyeasy to preempt and restore?)
  1. Howmany moreresourcesdoestheprocessneedtocomplete.
  2. How many processeswillneedtobeterminated
  3. Whethertheprocessisinteractiveorbatch.
  4. (Whether or not the process has made non-restorable changes to anyresource.)

2. ResourcePreemption

➔Whenpreemptingresourcesto relievedeadlock,therearethreeimportant issues to be addressed:

1. Selecting a victim - Deciding which resources to preempt from which processes involves many of the same decision criteria outlined above.
2. Rollback - Ideally one would like to roll back a preempted process to a safe state prior to the point at which that resource was originally allocated to the process. Unfortunately itcanbedifficultorimpossibleto determinewhatsucha safe state is, and so the only safe rollback is to roll back all the way back to the beginning. (I.e. abort the process and make it start over.)
3. Starvation-Howdoyouguaranteethataprocesswon'tstarvebecauseitsresources areconstantlybeingpreempted?Oneoptionwouldbetouseaprioritysystem,and increasethepriorityofaprocesseverytimeitsresourcesgetpreempted.Eventually it should get a high enough priority that it won't get preempted any more.

WINDOWS7–THREADANDSMPMANAGEMENT

ThenativeprocessstructuresandservicesprovidedbytheWindowsKernelarerelativelysimple and general purpose, allowing each OS subsystem to emulate a particular process structure and functionality.

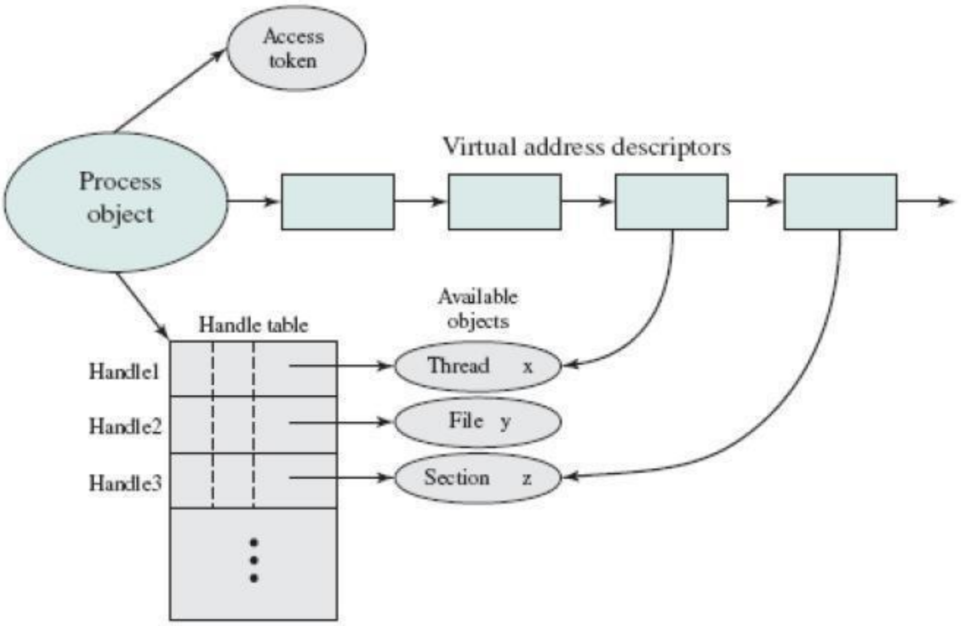
CharacteristicsofWindowsprocesses:

- Windowsprocessesareimplementedasobjects.
- Aprocesscanbecreatedasnewprocess,oras a copyofanexisting process.
- Anexecutableprocess maycontain oneormore threads.
- Bothprocessandthreadobjectshavebuilt-insynchronizationcapabilities.

AWindowsProcess andItsResources

- Eachprocessisassignedasecurityaccesstoken,calledtheprimarytokenofthe process.Whenauserfirstlogson,Windowscreatesanaccesstokenthat includesthe securityIDfortheuser.

- Everyprocessthatiscreatedbyorrunsonbehalfofthisuserhasacopyofthis accesstoken.
- Windowsusesthetokentovalidatetheuser’sabilitytoaccesssecuredobjectsor to perform restricted functions on the system and on secured objects. The access token controls whether the process can change its own attributes.



- Related to the process is a series of blocks that define the virtual address space currently assigned to this process.

- The process cannot directly modify these structures but mustrely on the virtualmemory manager, which provides a memory allocation service for the process.

- The process includes an object table, with handles to other objects known to this process.Theprocesshasaccesstoafileobjectandtoasectionobjectthatdefines a section of shared memory.

ProcessandThreadObjects

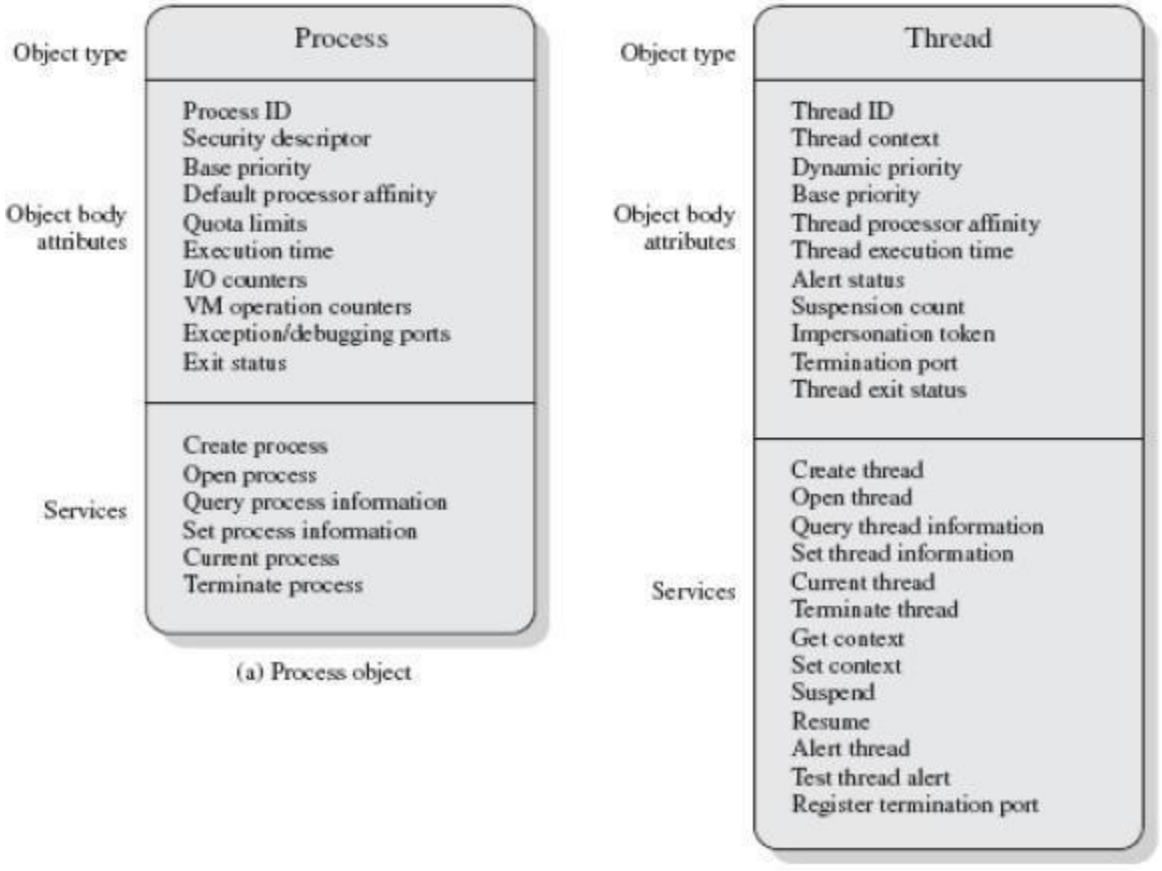
□ The object-orientedstructureofWindowsfacilitatesthedevelopmentofageneral-purpose process facility.

□ Windows makes use of two types of process-related objects: processes and threads.

□ A process is an entity corresponding to a user job orapplicationthatownsresources, such as memory and open files.

□ A thread is a dispatchable unit of work that executes sequentially and is interruptible, sothattheprocessorcanturntoanotherthread.

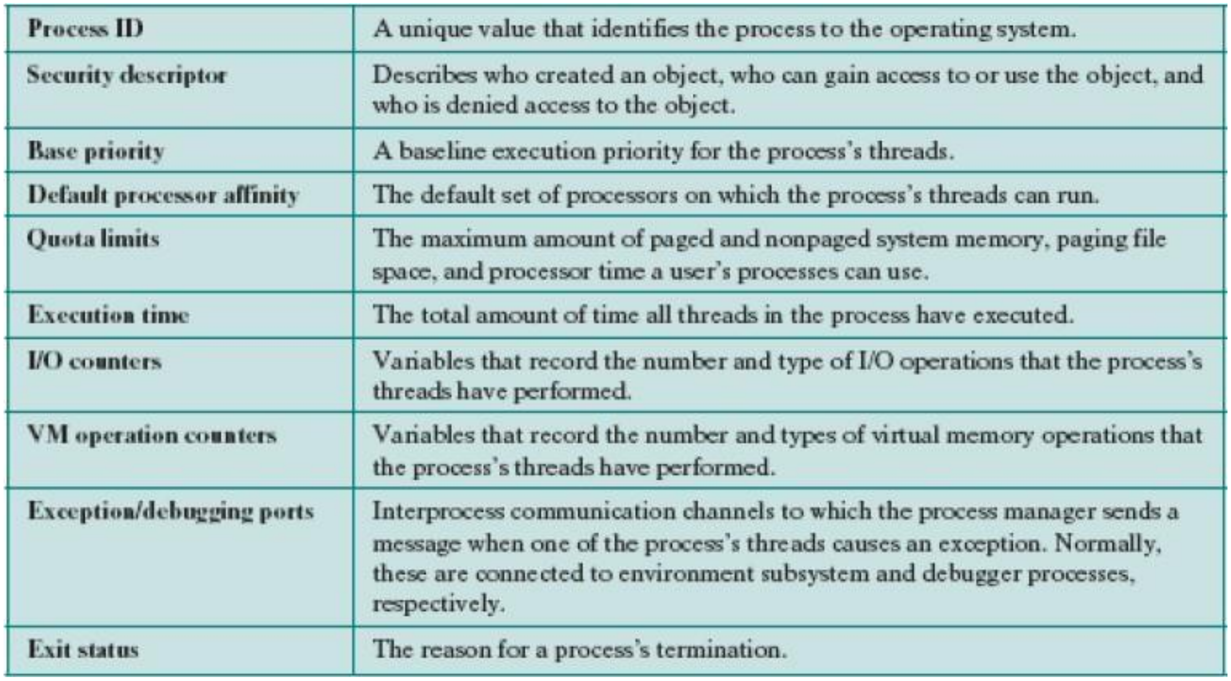
WindowsProcessandThreadObjects



(a) Process object

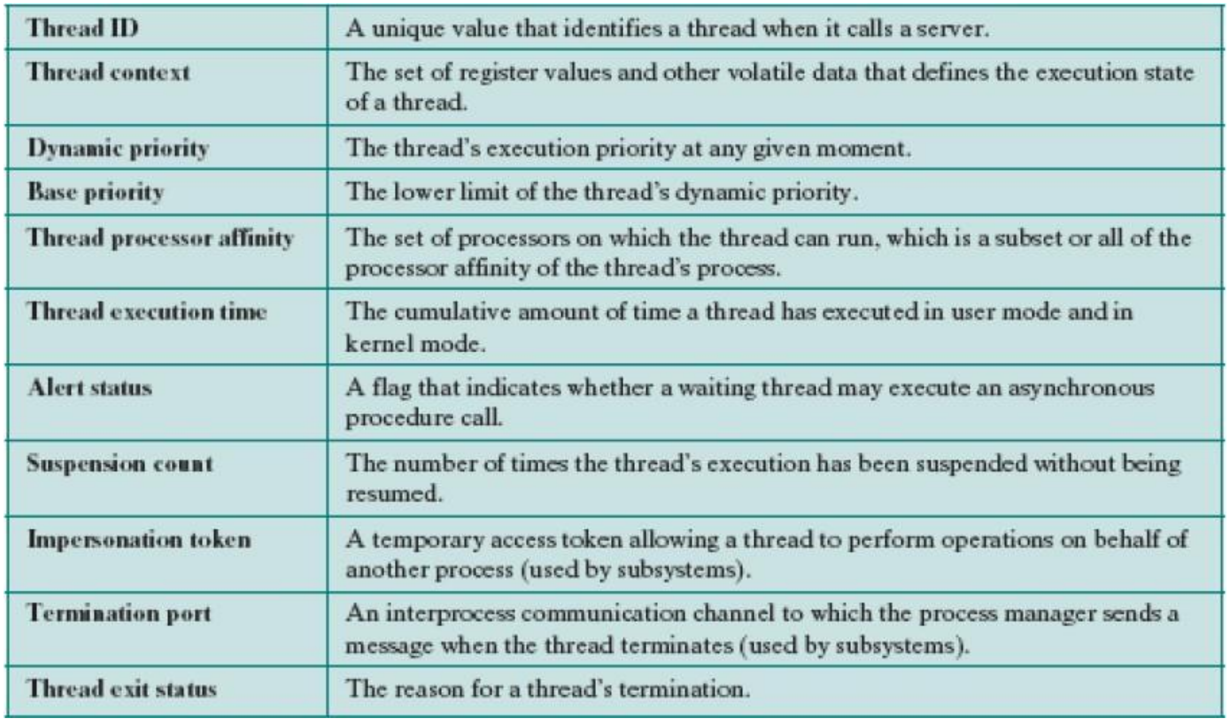
(b) Thread object

WindowsProcessObjectAttributes



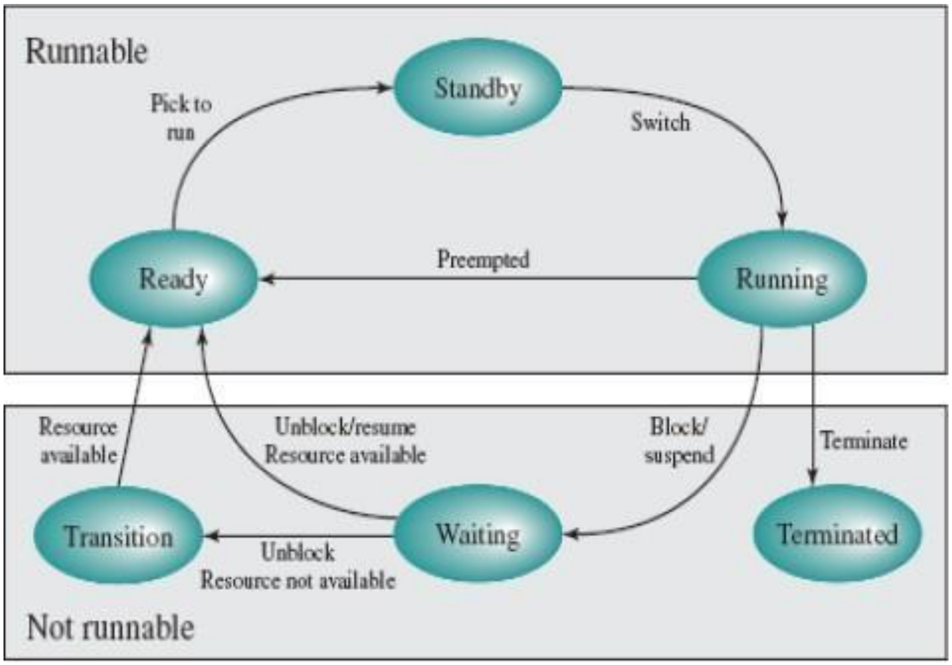
Process ID	A unique value that identifies the process to the operating system.
Security descriptor	Describes who created an object, who can gain access to or use the object, and who is denied access to the object.
Base priority	A baseline execution priority for the process's threads.
Default processor affinity	The default set of processors on which the process's threads can run.
Quota limits	The maximum amount of paged and nonpaged system memory, paging file space, and processor time a user's processes can use.
Execution time	The total amount of time all threads in the process have executed.
I/O counters	Variables that record the number and type of I/O operations that the process's threads have performed.
VM operation counters	Variables that record the number and types of virtual memory operations that the process's threads have performed.
Exception/debugging ports	Interprocess communication channels to which the process manager sends a message when one of the process's threads causes an exception. Normally, these are connected to environment subsystem and debugger processes, respectively.
Exit status	The reason for a process's termination.

WindowsThreadObjectAttributes



Thread ID	A unique value that identifies a thread when it calls a server.
Thread context	The set of register values and other volatile data that defines the execution state of a thread.
Dynamic priority	The thread's execution priority at any given moment.
Base priority	The lower limit of the thread's dynamic priority.
Thread processor affinity	The set of processors on which the thread can run, which is a subset or all of the processor affinity of the thread's process.
Thread execution time	The cumulative amount of time a thread has executed in user mode and in kernel mode.
Alert status	A flag that indicates whether a waiting thread may execute an asynchronous procedure call.
Suspension count	The number of times the thread's execution has been suspended without being resumed.
Impersonation token	A temporary access token allowing a thread to perform operations on behalf of another process (used by subsystems).
Termination port	An interprocess communication channel to which the process manager sends a message when the thread terminates (used by subsystems).
Thread exit status	The reason for a thread's termination.

ThreadStates



Problem

1. Considerthefollowingsetofprocesses,withthelengthoftheCPU-bursttimegivenin milliseconds:

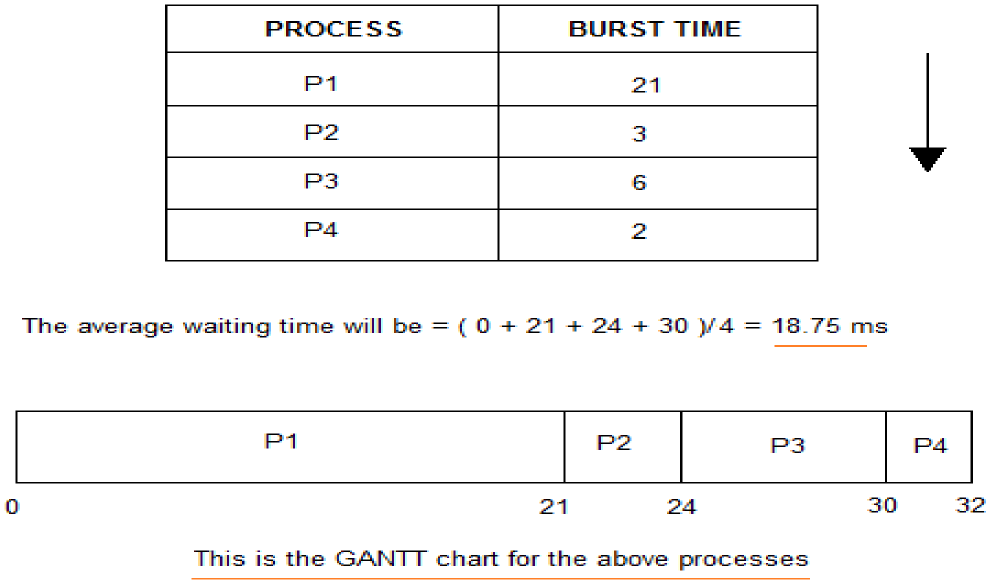
Process	**Burst Time**	**Arrival Time**	**Priority**
P1	23	0	2
P2	3	1	1
P3	6	2	4
P4	2	3	3

a. Draw four *Gantt charts*illustrating the execution of these processes using FCFS, SJF(Preemptive),anon-preemptivepriority(asmallerprioritynumberimpliesahigher priority), and RR (quantum = 1) scheduling.

c.Whatisthe*waitingtime*ofeach processforeach ofthescheduling algorithms inparta?

d.Whichoftheschedulesinpartaresultsinthe*minimalaveragewaitingtime*(overall processes)?

FCFSSCHEDULING



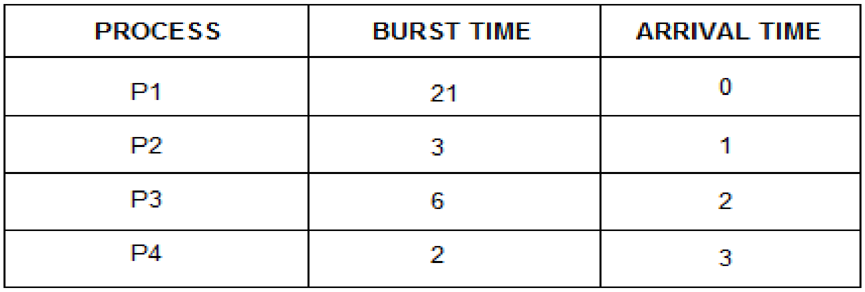
PROCESS	BURST TIME
P1	21
P2	3
P3	6
P4	2

The average waiting time will be = (0 + 21 + 24 + 30)/4 = 18.75 ms

This is the GANTT chart for the above processes

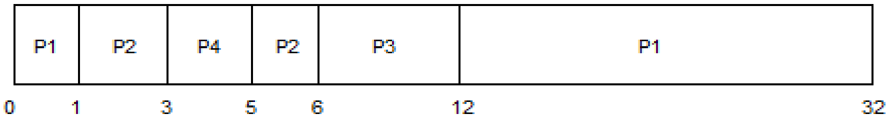
SJF(SHORTESTJOBFIRST)

InPre-emptiveShortestJobFirstScheduling,jobsareputintoreadyqueueastheyarrive,butas a process with short burst time arrives, the existing process is pre-empted.



PROCESS	BURST TIME	ARRIVAL TIME
P1	21	0
P2	3	1
P3	6	2
P4	2	3

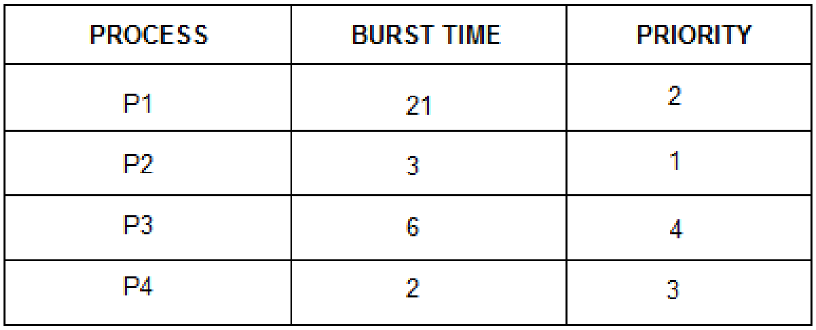
The GANTT chart for Preemptive Shortest Job First Scheduling will be,



The average waiting time will be, ((5-3) + (6-2) + (12-1))/4 = 4.25 ms

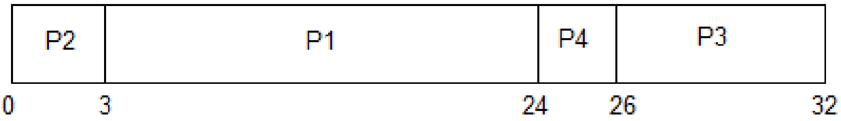
The average waiting time for preemptive shortest job first scheduling is less than both, non-preemptive SJF scheduling and FCFS scheduling.

PRIORITY



PROCESS	BURST TIME	PRIORITY
P1	21	2
P2	3	1
P3	6	4
P4	2	3

The GANTT chart for following processes based on Priority scheduling will be,

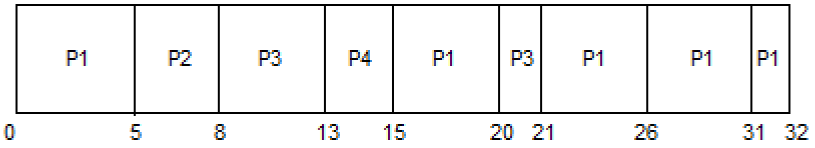


The average waiting time will be, (0 + 3 + 24 + 26)/4 = 13.25 ms

ROUND ROBIN

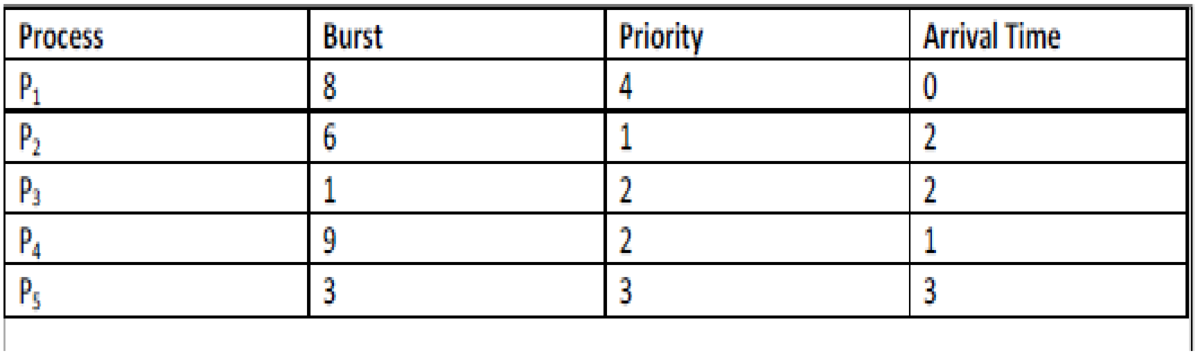
PROCESS	BURST TIME
P1	21
P2	3
P3	6
P4	2

The GANTT chart for round robin scheduling will be,



The average waiting time will be, 11 ms.

2.



Process	Burst	Priority	Arrival Time
P_1	8	4	0
P_2	6	1	2
P_3	1	2	2
P_4	9	2	1
P_5	3	3	3

First Come First Served

0	8	17	23	24	27
P_1	P_4	P_2	P_3	P_5	

Avg. Wait = 0+8-1+17-2+23-2+24-3 = 0+7+15+21+21=64/5 = 12.8 AVG TAT = 8+17-1+23-2+24-2+27-3 = 8+16+21+22+24=91/5=18.2

SJF

0	8	14	15	24	27
P_1	P_2	P_3	P_4	P_5	

Avg. Wait = 8-2+14-2+15-1+24-3 = 6+12+14+21 = 53/5=10.6ms AVG TAT = 8+14-2+15-2+24-1+27-3 = 8+12+13+23+24=80/5=16ms

Priority

0	1	2	8	9	17	20	27
P_1	P_4	P_2	P_3	P_4	P_5	P_1	

Avg. Wait Time = 0+20-1+2-2+8-2+9-2+17-3 = 0+19+0+6+7+14 = 46/5=9.2ms AVG TAT = 27+8-2+9-2+16+20-3 = 73/5 = 14.6ms

Round Robin

Round Robin (1ms Quantum)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
P_1	P_4	P_2	P_3	P_5	P_1	P_4	P_2	P_5	P_1	P_4	P_2	P_5	P_1	P_4	P_2	P_1	P_4	P_2	P_1	P_4	P_2	P_1	P_4	P_1	P_4	P_4	

Wait Time P_1 = 0+4+3+3+2+2+1+1 = 16

Wait Time P2 = 0+4+3+3+2+2+2+1 = 17

Wait Time P3 = 1

Wait Time P4 = 4+4+3+2+3+2+1 = 19

Wait Time P5 = 1+3+3 = 7

Avg Wait Time = 60/5 = 12ms

Avg TAT =25+21+2+26+10 = 84/5 = 16.8

UNITIIISTORAGE MANAGEMENT

MainMemory-ContiguousMemoryAllocation,Paging,Segmentation,Segmentationwithpaging,32 and64bitarchitectureExamples;VirtualMemory-Background,DemandPaging,PageReplacement, Allocation, Thrashing; Allocating Kernel Memory, OS Examples.

1. MEMORYMANAGEMENT:BACKGROUND

Memorymanagementisthefunctionalityofanoperatingsystemwhichhandlesormanagesprimary memory and moves processes back and forth between main memory and disk during execution.

Memory management keeps track of each and every memory location, regardless of either it is allocatedtosomeprocess oritisfree.Itcheckshowmuchmemoryistobeallocatedtoprocesses.

Itdecideswhichprocesswillgetmemoryatwhattime.

Ittrackswheneversomememorygetsfreedorunallocatedandcorrespondingly itupdatesthestatus.

BasicHardware

Programmust bebrought(from disk)intomemory andplaced withinaprocessforit toberun

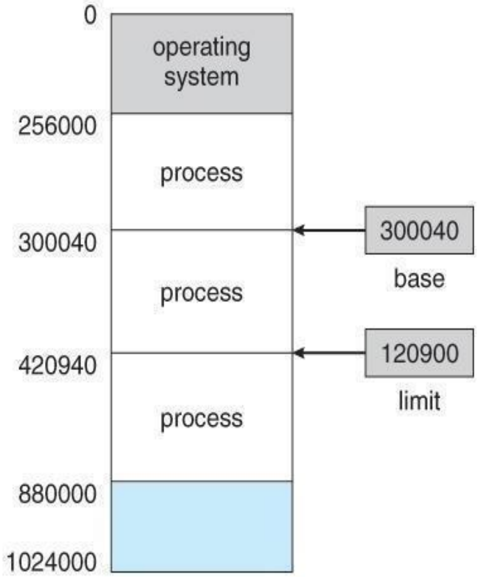
- MainmemoryandregistersareonlystorageCPUcanaccessdirectly
- Memoryunit onlyseesastreamofaddresses +readrequests,oraddress+dataandwriterequests
- RegisteraccessinoneCPUclock(or less)
- Mainmemorycantakemanycycles,causinga**stall**
- Cachesitsbetweenmainmemory andCPUregisters
- Protectionofmemoryrequiredtoensurecorrectoperation

➔ Wecan providethis protectionbyusingtwo registers,usually a **base**anda**limit**

Thebaseregisterholdsthesmallestlegalphysicalmemory address;

The limit register specifies the size of the range.

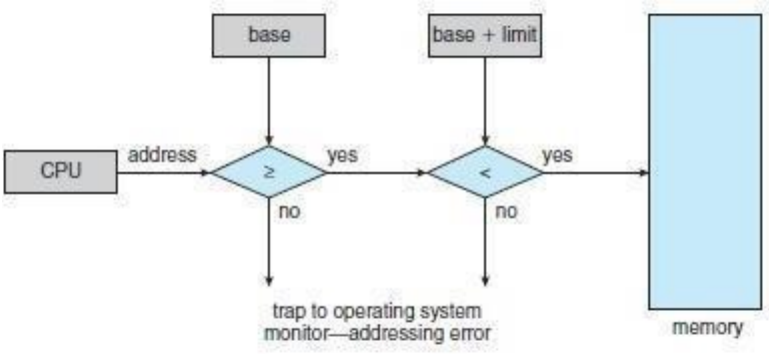
Forexample,ifthebaseregisterholds300040andlimitregisteris120900,thenthe program can legally access all addresses from 300040 through 420940 (inclusive).



ProtectionofmemoryspaceisaccomplishedbyhavingtheCPUhardwarecompareeveryaddress generated in user mode with the registers.

Anyattemptbyaprogramexecutinginusermodetoaccessoperating-systemmemoryorother users'memoryresultsin atraptotheoperatingsystem,whichtreatstheattemptasafatal error.

Thisschemepreventsauserprogramfrom(accidentallyordeliberately)modifyingthecodeordata structures of either the operating system or other users.



<u>AddressBindingDefi</u>

<u>nition</u>

Convertingtheaddressused in aprogram toanactual physicaladdress.

Addressbindingistheprocessofmappingtheprogram'slogicalorvirtualaddressesto corresponding physical or main memory addresses.

Inotherwords,agivenlogicaladdressismappedbytheMMU(MemoryManagementUnit)toa physical address.

Userprogramstypicallyrefertomemoryaddresseswithsymbolicnamessuchas"i","count",and "average Temperature".

Thesesymbolicnamesmustbemappedorboundtophysicalmemoryaddresses,whichtypically occurs in several stages:

Threedifferentstagesof binding:

1. **Compiletime**.Thecompilertranslatessymbolicaddressestoabsoluteaddresses.Ifyouknow at compile time where the process will reside in memory, then absolute code can be generated (Static).

2. **Load time**. The compiler translates symbolic addresses to relative (relocatable) addresses. The loader translates these to absolute addresses. If it is not known at compile time where the process will reside in memory, then the compiler must generate relocatable code (Static).

3. **Executiontime**.Iftheprocesscanbemovedduringitsexecutionfromonememorysegment toanother,thenbindingmustbedelayeduntilruntime.Theabsoluteaddressesaregeneratedby hardware. Most general-purpose OS use this method (Dynamic).

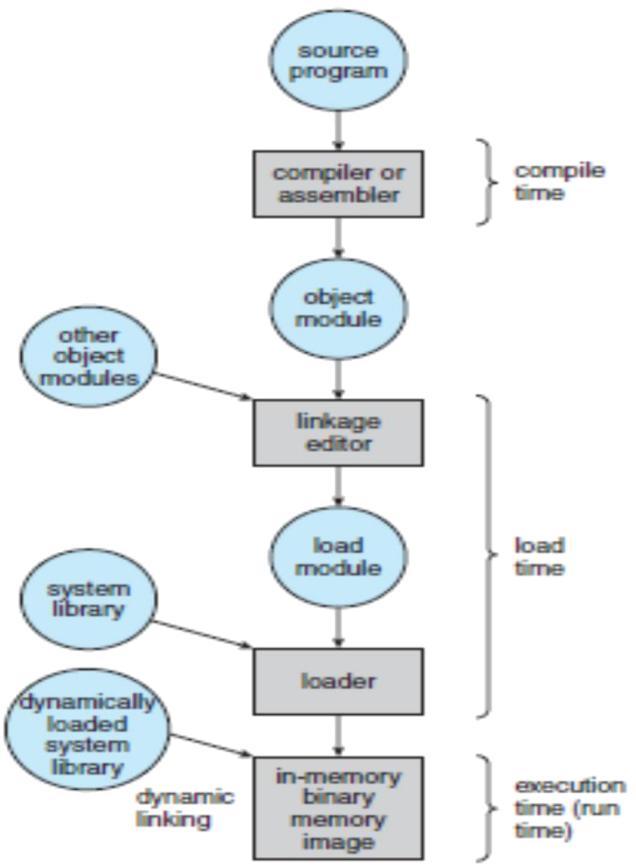


Figure 8.3 Multistep processing of a user program.

Logicalvs.PhysicalAddress Space

Logicaladdress–generatedbytheCPU;alsoreferredtoas**"virtualaddress" Physical**

address – address seen by the memory unit.

Logicalandphysicaladdressesarethe**same**incompile-timeandload-timeaddress-binding schemes

Logical(virtual)andphysical addresses**differ**inexecution-timeaddress- bindingscheme

Memory-ManagementUnit(MMU)

Itisahardwaredevicethatmapsvirtual/Logicaladdresstophysical address.
Inthisscheme,therelocationregister'svalueisaddedtoLogicaladdressgeneratedbya user process.

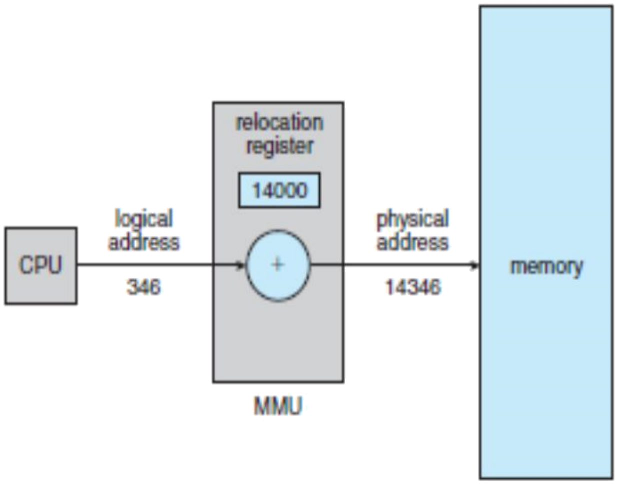
TheBaseregisteriscalled arelocation register.

□ The value intherelocationregisterisaddedtoeveryaddressgeneratedbyauserprocess at the time it is sent to memory

□ For example, if the base is at 14000, then an attempt by the user to address location 0 is dynamicallyrelocatedtolocation14000;anaccesstolocation346ismappedtolocation 14346.

□ The user program never sees the real physical addresses. The program can create a pointer tolocation346,storeitinmemory,manipulateit,andcompareitwithotheraddresses –allas the number 346.

□ The user program deals with logical addresses.



1.4DynamicLoading

Dynamicloadingisamechanismby whichacomputerprogram can,atruntime,load alibrary(orotherbinary)intomemory, retrievetheaddressesoffunctionsandvariables containedin thelibrary, execute those functions oraccess thosevariables, and unloadthe library from memory.

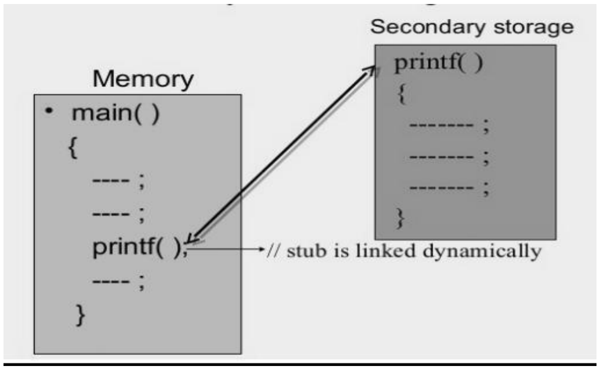
Dynamicloadingmeans loadingthelibrary(oranyotherbinaryforthatmatter)intothememory during load or run-time.

Dynamic loading can be imagined to be similar to plugins, that is an exe can actually executebeforethedynamicloadinghappens(Thedynamicloadingforexamplecanbe created using Load Library call in C or C++)

DynamicLinkingandsharedlibraries

Dynamiclinkingreferstothelinkingthatisdoneduringloadorrun-timeandnotwhenthe exe is created.
In case of dynamic linking the linker while creating the exe does minimal work. For the dynamiclinkertoworkitactuallyhastoloadthelibrariestoo.Henceit'salsocalledlinking loader. Smallpieceofcode, *stub*,used toindicate howto loadlibraryroutine.
Stub replaces itself with the address of the routine, and executes the routine. Operatingsystemneededtocheckifroutineisinprocessesmemoryaddress. Dynamic linking is particularly useful for libraries.

- Sharedlibraries:Programslinkedbeforethenewlibrarywasinstalledwillcontinueusing the older library.



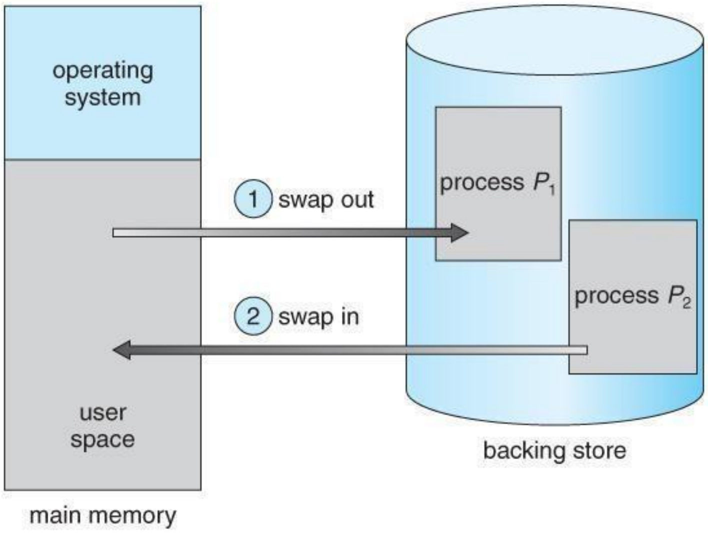
2. SWAPPING

Basic

- Aprocesscanbeswappedtemporarilyoutofmemorytoabackingstore(SWAPOUT)and then brought back into memory for continued execution (SWAP IN).
- **Backingstore**–fastdisklargeenoughtoaccommodatecopiesofall memory imagesforallusers & itmustprovide direct access to these memory images
- **Rollout,rollin**–swappingvariantusedforpriority-basedschedulingalgorithms;lower-priority process is swapped out so higher-priority process can be loaded and executed
- **Transfertime:**Majorpartofswaptimeistransfertime.Totaltransfertimeis directly proportional to the amount of memory swapped.

□ **Example**:Letusassumetheuserprocessisofsize1MB&thebacking storeisastandardhard disk with a transfer rate of 5MBPS.

Transfertime =1000KB/5000KBper second
=1/5 sec=200ms



A process with dynamic memory requirements will need to issue system calls (request memory()andreleasememory())toinformtheoperatingsystemofitschangingmemoryneeds.

SwappingonMobile Systems

Swappingistypicallynotsupportedonmobileplatforms,forseveralreasons:

Mobiledevicestypicallyuseflashmemoryinplaceofmorespaciousharddrives for persistent storage, so there is not as much space available.

Flashmemorycanonlybewrittentoalimitednumberoftimesbeforeit becomes unreliable.

Thebandwidthtoflashmemoryisalso lower.

Apple'sIOSasksapplications tovoluntarilyfreeupmemory

Read-onlydata,e.g.code,issimplyremoved,andreloadedlaterifneeded. Modified data, e.g. the stack, is never removed.

Appsthatfailtofreeupsufficientmemorycanberemovedbythe OS

Android follows a similar strategy.

Priortoterminatingaprocess,Androidwritesitsapplicationstatetoflashmemoryfor quick restarting.

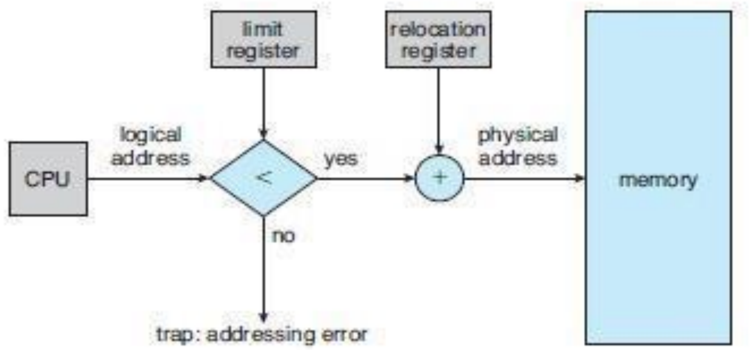
3. CONTIGUOUSMEMORYALLOCATION

Oneapproachtomemorymanagementistoloadeachprocessintoacontiguous space.

Theoperatingsystemisallocatedspacefirst,usuallyateitherlowohighmemorylocations,andthen the remaining available memory is allocated to processes as needed.

MemoryProtection

Protection against user programs accessing areas that they should not, allows programs to be relocatedtodifferentmemorystartingaddressesasneeded,andallowsthememoryspacedevotedto the OS to grow or shrink dynamically as needs change.



MemoryAllocation

Incontiguousmemoryallocationeachprocessiscontainedinasinglecontiguousblockofmemory. Memory is divided into several fixed size partitions. Each partition contains exactly one process.

Whenapartitionis free,aprocessisselectedfromtheinput queueandloadedinto it.

Thereare twomethodsnamely:

- □ Fixed–P**a**rtition Method
- □ Variable–PartitionMethod

- **Fixed–Partition Method:**

Dividememoryintofixedsizepartitions,whereeachpartitionhasexactlyone process. The drawback is Memory space unused within a partition is wasted.(eg. When process size < partition size)

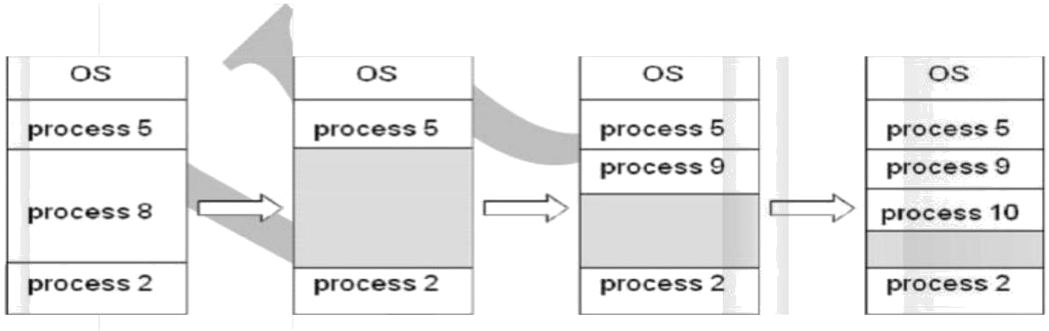
- **Variable-partitionmethod:**

o Dividememoryintovariablesizepartitions,dependinguponthesizeofthe incoming process.

o Whenaprocessterminates,thepartitionbecomesavailableforanotherprocess.

o Asprocessescomplete and leavetheycreateholesinthemain memory.

o ***Hole***–blockofavailablememory;holesofvarioussizearescatteredthroughout memory.



DynamicStorage-AllocationProblem:

Howtosatisfyarequestofsize n'fromalistoffree holes?

➔Thefreeblocksofmemoryareknownasholes.Thesetofholesissearched to determine which hole is best to allocate.

Solution:

o First-fit:Allocatethefirstholethatisbig enough.

o Best-fit:Allocatethesmallestholethatisbigenough;mustsearchentirelist,unless ordered by size. Produces the smallest leftover hole.

o Worst-fit:Allocatethelargesthole;mustalsosearchentirelist.Producesthe largest leftover hole.

Example:

Givenfivememorypartitionsof100KB,500KB,200KB,300KB,and600KB(inorder),howwould eachofthefirst-fit,best-fit,andworst-fitalgorithmsplaceprocessesof212KB,417KB,112KB,and 426 KB (in order)?Which algorithm makes the most efficient use of memory?

a. First-fit:
1. 212Kis put in 500K partition
2. 417Kis put in 600K partition
3. 112Kisputin 288Kpartition(newpartition288K =500K−212K)
4. 426Kmustwait

b. Best-fit:
1. 212Kis put in 300K partition
2. 417Kis put in 500K partition
3. 112Kis put in 200Kpartition
4. 426Kis put in 600K partition

c. Worst-fit:
l.212Kisputin600Kpartition
2. 417Kis put in 500K partition
3. 112Kis put in 388K partition
4. 426Kmustwait

Inthisexample,best-fitturns outtobethebest.

NOTE:First-fitandbest-fitarebetterthanworst-fitintermsofspeedandstorageutilization

Fragmentation:

Fragmentationisaphenomenoninwhichstoragespaceisused inefficiently, reducing capacity or performance and often both.

1. **ExternalFragmentation**–Thistakesplacewhenenoughtotalmemoryspace exists to satisfy a request, but it isnot contiguous i.e, storage is fragmented into a large number of small holes scattered throughout the main memory.
2. **InternalFragmentation**–Allocatedmemorymaybeslightlylargerthan requested memory.

Example:hole=184 bytesProcesssize= 182 bytes.

Weareleft with ahole of2 bytes.

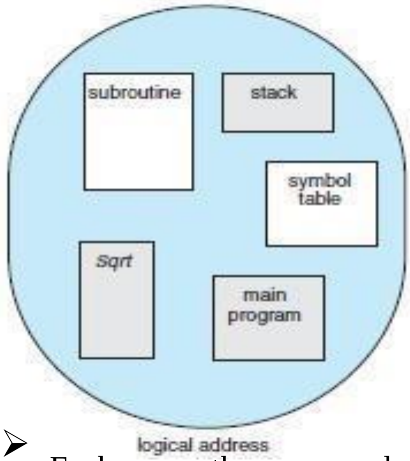
➔**Solutions**

Compaction: Moveallprocessestowardsoneendofmemory,holetowardsother end of memory, producing one large hole of available memory. This scheme is expensive as it can be done if relocation is dynamic and done at execution time.

4. SEGMENTATION

BasicMethod

- o Memory-managementschemethatsupportsuserviewof memory
- o Aprogramisacollectionofsegments.Asegmentisalogicalunitsuchas:Mainprogram, Procedure, Function, Method, Object, Local variables, global variables, Common block, Stack, Symbol table, arrays



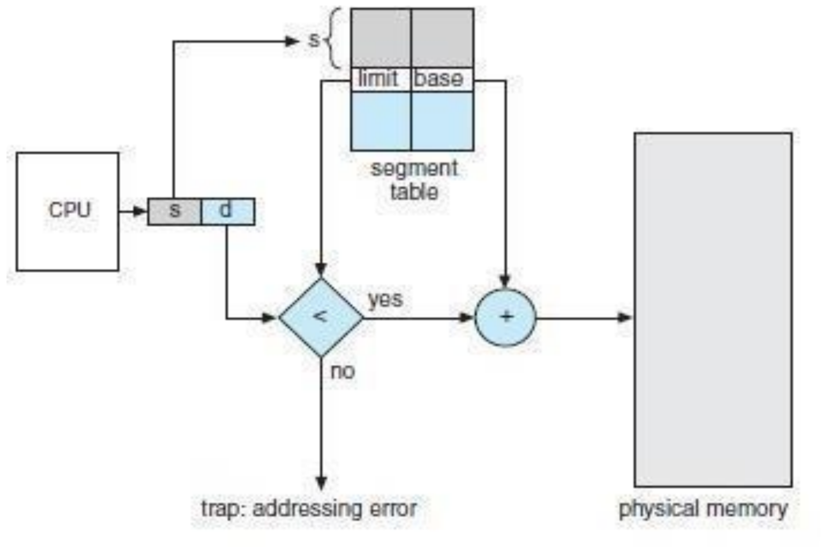
- ➢ Eachsegmenthasanameand alength.
- ➢ Theaddresses specifybooththesegmentnameandtheoffsetwithinthesegment.
- ➢ Theprogrammerthereforespecifieseachaddressbytwo quantities: asegmentnameandanoffset.

Alogical address consistsofatwo tuple: <segment-number,offset>.

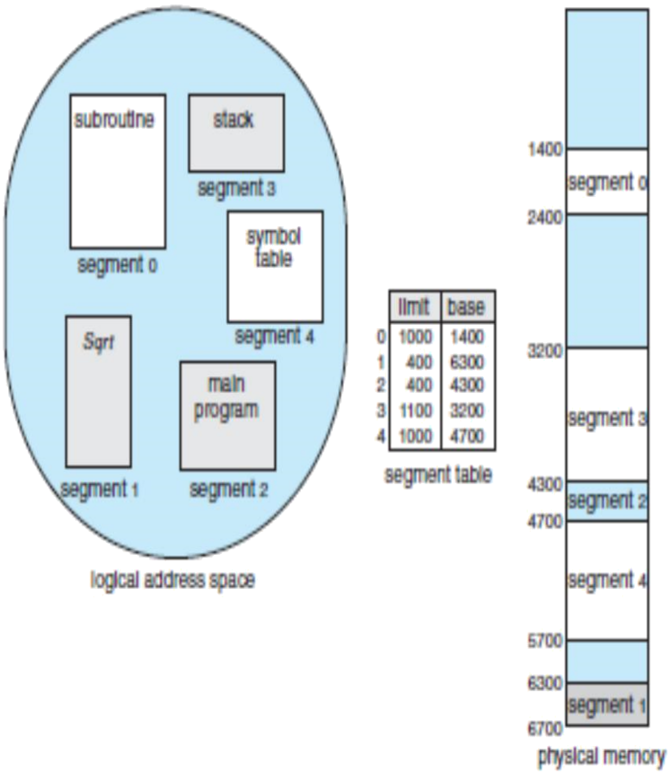
SegmentationHardware

Eachentryinthesegment tablehas asegmentbaseandasegment limit.
Thesegmentbase containsthestartingphysicaladdresswherethesegmentresidesin memory, and the segment limit specifies the length of the segment
Alogicaladdressconsists of two➔parts: ➔
a**segmentnumber**
s,andan**offset**intothatsegment d. The
segmentnumberisusedasanindextothesegmenttable.

The**offset**d ofthelogical address mustbebetween 0and thesegment limit. Ifitisnot,wetraptotheoperatingsystem(logicaladdressingattemptbeyondendofsegment). Whenanoffsetislegal,itisaddedtothesegmentbasetoproducetheaddressinphysical memory of the desired byte.



Forexample,
segment 2 is 400 bytes long and begins at location 4300. Thus, a reference to byte 53 of segment 2 is mapped onto location 4300 + 53 = 4353. A reference to segment 3, byte 852, is mapped to 3200 (the base of segment 3) + 852 = 4052. A reference to byte 1222 of segment 0 would result in a trap to the operating system, as this segment is only 1,000 bytes long.



	limit	base
0	1000	1400
1	400	6300
2	400	4300
3	1100	3200
4	1000	4700

5. PAGING

- Itisamemorymanagementschemethatpermitsthephysicaladdressspaceof a process to be noncontiguous.
- Itavoidstheconsiderableproblemoffittingthevaryingsizememorychunkson to the backing store.

BasicMethod

o Dividelogicalmemoryintoblocks ofsamesizecalled**“pages”**.

o Dividephysicalmemoryintofixed-sizedblockscalled**“frames”**

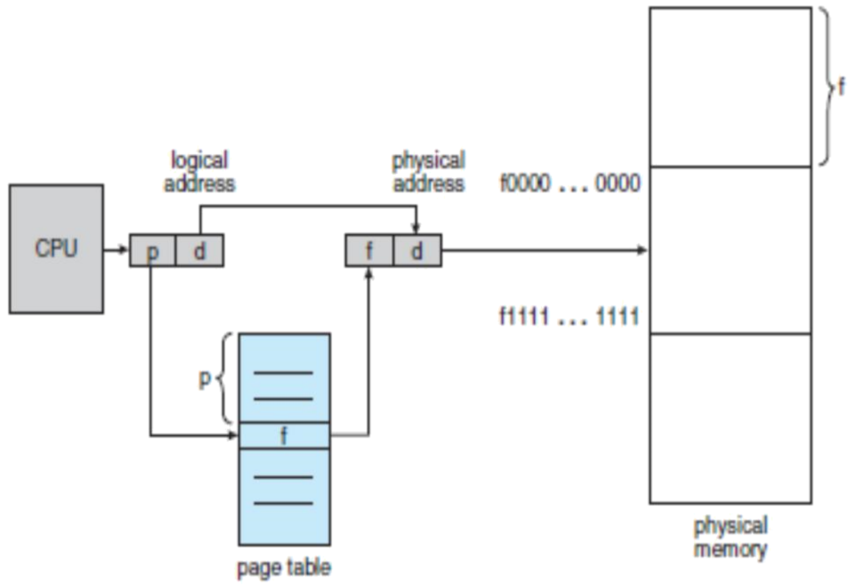
o Pagesizeisapowerof2,between 512bytesand 16MB.

AddressTranslationScheme

AddressgeneratedbyCPU(logicaladdress)isdividedinto:

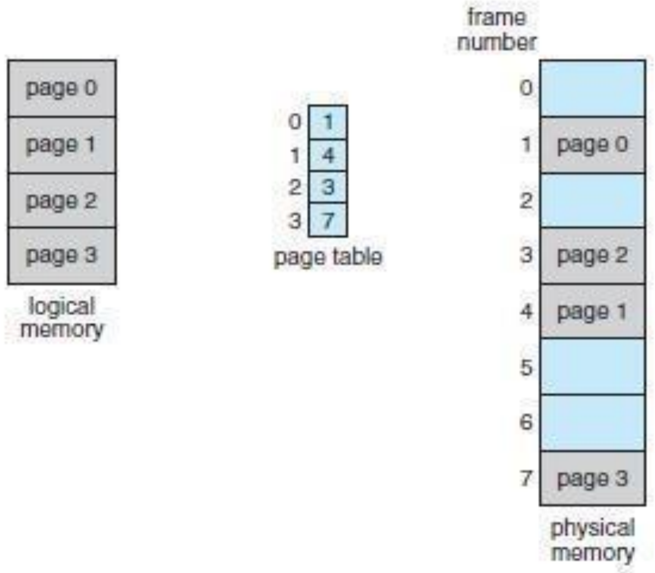
Pagenumber*(p)*–usedasanindexintoapagetablewhichcontains baseaddressof eachpage inphysical memory

Pageoffset*(d)*–combinedwithbaseaddresstodefinethephysicaladdress i.e.,
Physical address = base address + offset



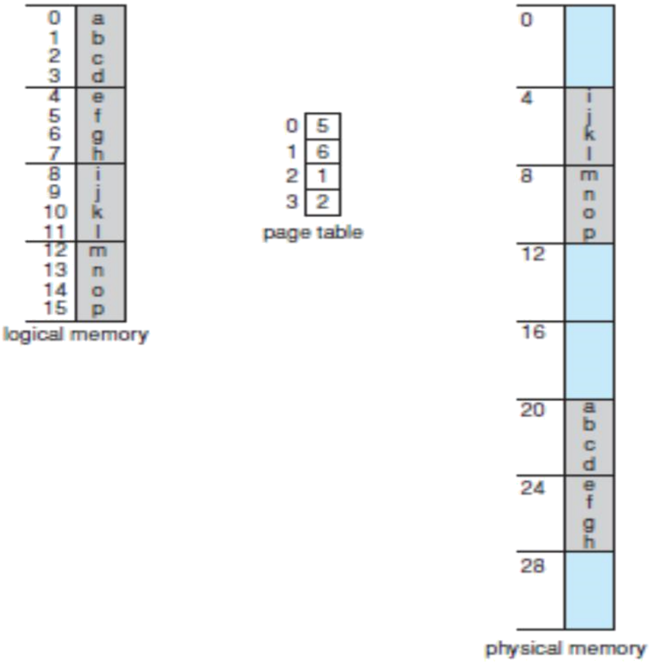
Thepagenumberisused asanindexintoapagetable.Thepagetablecontainsthebaseaddress of each page in physical memory.

Thisbaseaddressiscombinedwiththepageoffsettodefinethephysicalmemoryaddressthatis sent to the memory unit.



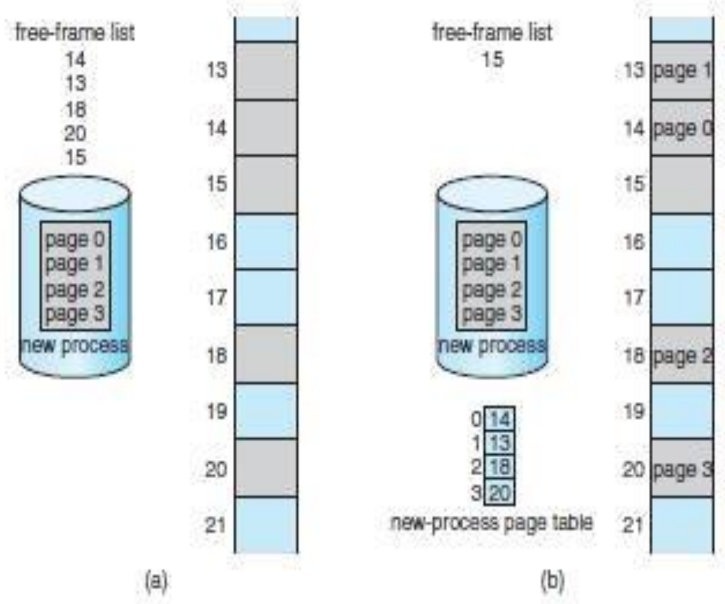
Consider the memory in the logical address, n= 2 and m = 4. Using a page size of 4 bytes and a physicalmemoryof32bytes(8pages),weshowhowtheprogrammer’sviewofmemorycanbe mapped into physical memory. Logical address 0 is page 0, offset 0. Indexing into the page table, we find that page 0 is in frame 5.

Thus,logicaladdress0mapstophysical address20[=(5×4)+0].Logicaladdress 3(page0, offset3)mapstophysicaladdress23[=(5×4)+ 3].Logicaladdress4ispage1,offset0; according to the page table, page 1 is mapped to frame 6. Thus, logical address 4 maps to physical address 24 [= (6 × 4) + 0]. Logical address 13 maps to physical address 9.



Sincetheoperating system is managing physical memory, it must be awareof the allocation details ofphysicalmemory,whichframesareallocated,whichframesareavailable,howmanytotal frames there are, and so on.

Thisinformationisgenerally keptinadatastructurecalledaframe table.
The frame table has one entry for each physical page frame, indicating whether the latter is free or allocated and, if it is allocated, to which page of which process or processes.



HardwareSupport

- TheTLBisassociative,high-speedmemory.
- EachentryintheTLBconsistsoftwoparts:
  - akey(ortag)anda value.
- Whentheassociativememoryispresentedwithanitem,theitemiscomparedwithall keyssimultaneously.
- Iftheitemisfound,thecorrespondingvaluefieldisreturned.
- TheTLBcontainsonlyafewofthepage-table entries.
- WhenalogicaladdressisgeneratedbytheCPU,itspagenumberispresentedtotheTLB.
- IfthepagenumberisnotintheTLB(knownasaTLBmiss),amemoryreference to the page
  tablemustbemade.
- DependingontheCPU,thismaybedoneautomaticallyinhardwareorviaaninterrupttothe operatingsystem.
- Ifthepagenumberisfound,itsframenumberisimmediatelyavailableandisusedtoaccess Memory.

  HitRatio-ThepercentageoftimesthatthepagenumberofinterestisfoundintheTLBis called the hitratio.

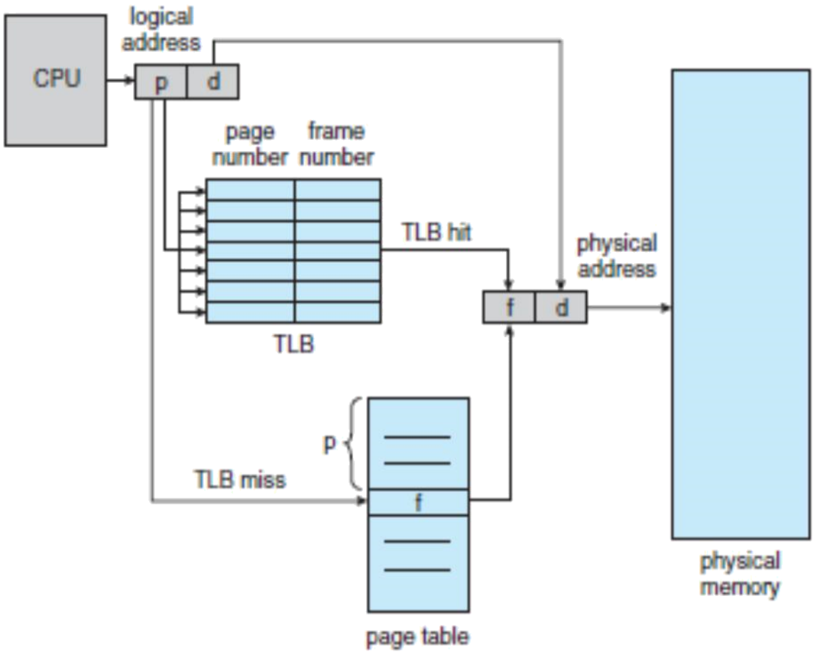
- An80-percenthitratio,forexample,meansthatwefindthedesiredpagenumberintheTLB80 percent of the time. If it takes 100 nanoseconds to access memory, then a mapped-memory access takes 100 nanoseconds when the page number is in the TLB.

- IfwefailtofindthepagenumberintheTLBthenwemustfirstaccessmemoryforthepage table and frame number (100 nanoseconds) and then access the desired byte in memory (100 nanoseconds), for a total of 200 nanoseconds.

  effectiveaccesstime=0.80×100+0.20×200

  =120 nanoseconds

  Fora99-percenthit ratio, whichismuchmorerealistic,wehaveeffectiveaccesstime=
  0.99 ×100 +0.01 ×200 =101 nanoseconds

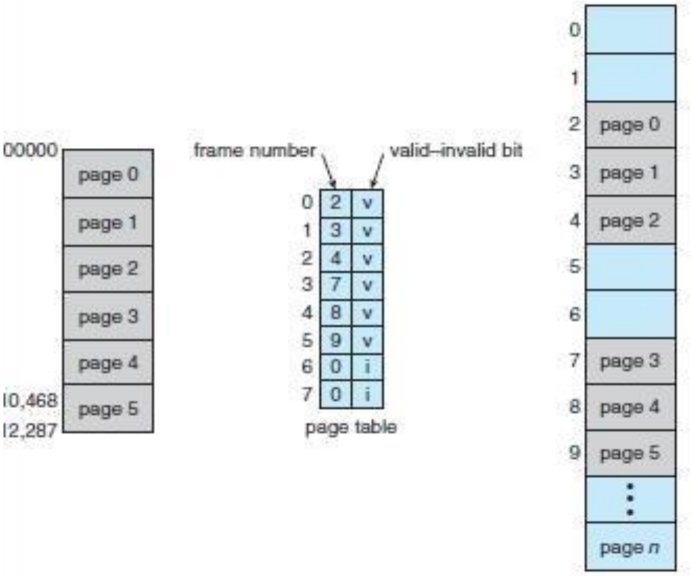


Protection

Memoryprotectioninapagedenvironmentisaccomplishedbyprotectionbitsassociatedwitheach frame. Oneadditionalbit isgenerally attachedto eachentry inthepagetable: avalid–invalid bit.

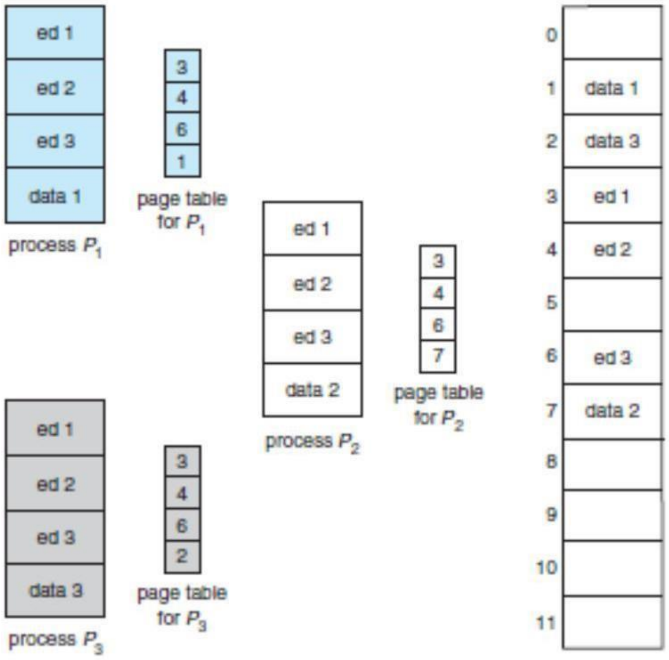
Whenthisbitissettovalid,theassociatedpageisintheprocess'slogicaladdressspaceandisthus a legal (or valid) page.

Whenthebitissettoinvalid,thepageisnotintheprocess'slogicaladdressspace. Illegaladdresses are trapped by use of the valid–invalid bit.



SharedPages

Anadvantageofpaging isthepossibilityofsharingcommoncode.



6. STRUCTUREOFPAGETABLE

Themostcommontechniquesforstructuringthepagetable,includinghierarchicalpaging,hashed page tables, and inverted page tables.

1. HierarchicalPaging

Thepagetableitselfbecomeslargeforcomputerswith largelogicaladdressspace(232to264).
Example:

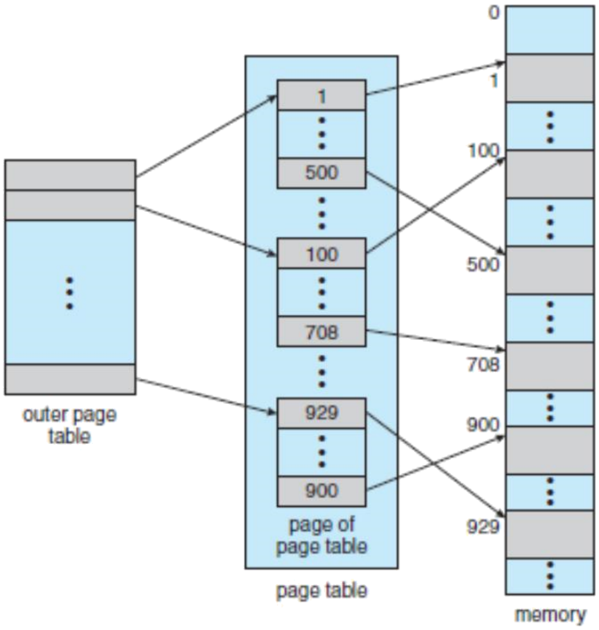
 Considerasystemwitha32-bitlogicaladdressspace. Ifthepagesizeinsuchasystemis4 KB(212), then a page table may consist of up to 1 million entries (232/212).

 Assuming that each entry consists of 4 bytes, each process may need up to 4 MB of physicaladdressspacefor thepagetablealone.

 The page table should be allocated contiguously in main memory.

 The solution to this problem is to divide the page table into smaller pieces.

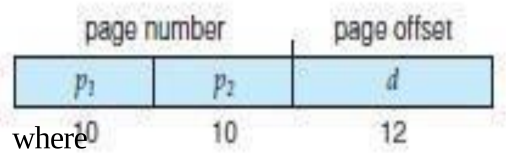
 One way of dividing the page table is to use a two-levelpaging algorithm,



Forexample, consider again thesystem with a32-bit logical address spaceand apagesizeof4 KB. A logical address is divided into a page number consisting of 20bitsand apage offset consisting of 12 bits.

Becausewepagethepagetable, thepagenumber is furtherdivided intoa10-bit pagenumberand a 10-bit page offset.

Thus,alogicaladdressisas follows:

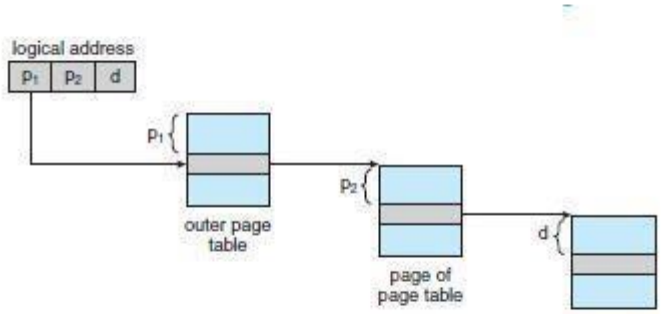


where

p1-anindex into theouterpagetable

p2-thedisplacement within thepageoftheinnerpagetable.

The address-translation method for this architecture is shown in the figure. Because address translation worksfromtheouterpagetableinward,this schemeisalso knownasaforward-mappedpagetable.



2. HashedPageTables

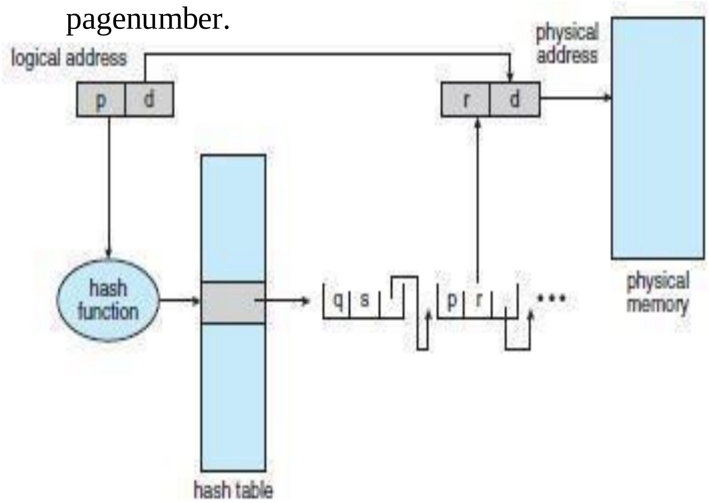
□ A common approach for handling addressspaceslargerthan 32bits istouseahashed page table,withthehashvaluebeingthevirtualpagenumber.

□ Each entry in the hash table contains a linked list of elements that hash to the same location (to handle collisions).

- Each element consistsofthreefields: The
  - virtual page number
  - Thevalueofthemappedpageframe
  - Apointerto the nextelement in thelinked list.

Algorithm:

- The virtual page number in the virtual address is hashed into the hash table.
- The virtual page numb eriscomparedwithfield1inthefirstelementinthelinked list.
- If there is a match, the corresponding page frame (field 2) is used to form the desired physical address.
- If there is no match, subsequent entries in the linked list are searched f oramatching virtual pagenumber.



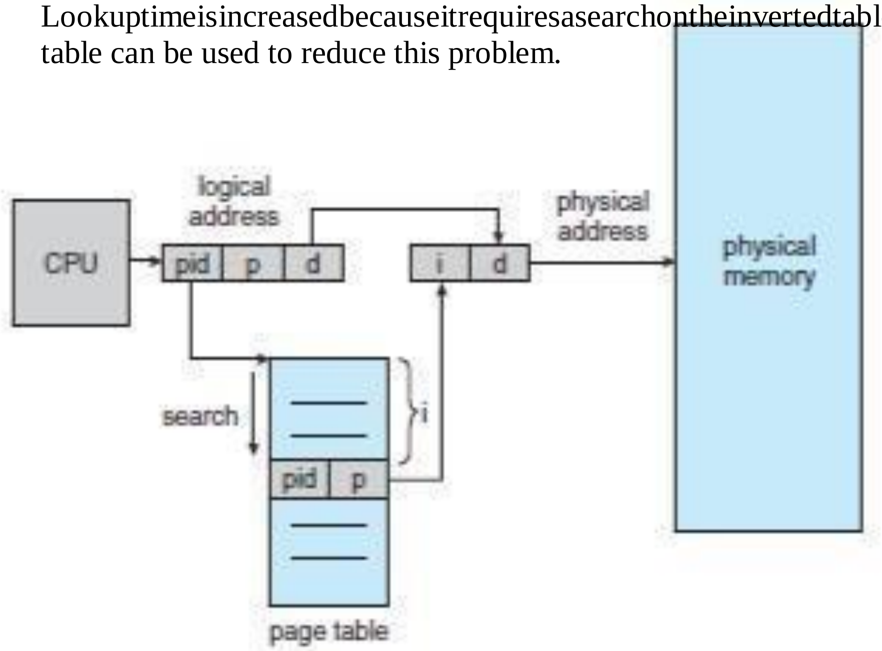
3. InvertedPageTable

Witheachprocesshavingitsownpagetable,andwitheachpagetableconsuming considerable amount of memory

Weusealot of memorytokeep trackofmemory.

Invertedpagetablehas oneentry foreach real pageofmemory.

Lookuptimeisincreasedbecauseitrequiresasearchontheinvertedtable. Hash table can be used to reduce this problem.



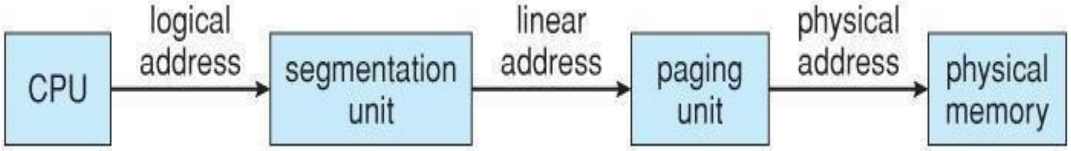
Eachvirtualaddressinthesystemconsistsofatriple:

<process-id,page-number,offset>.

7.INTEL32AND64-BITARCHITECTURES

IA-32Segmentation

ThePentiumCPUprovidesbothpuresegmentationandsegmentationwithpaging.Inthelattercase, theCPUgeneratesalogicaladdress(segment-offsetpair),whichthesegmentationunitconvertsinto a logical linear address, which in turn is mapped to a physical frame by the paging unit



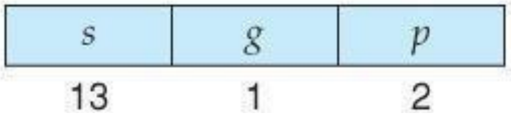
IA-32Segmentation

ThePentiumarchitectureallowssegmentstobeaslargeas4GB,(24bitsofoffset). Processes can have as many as 16K segments, divided into two 8K groups:

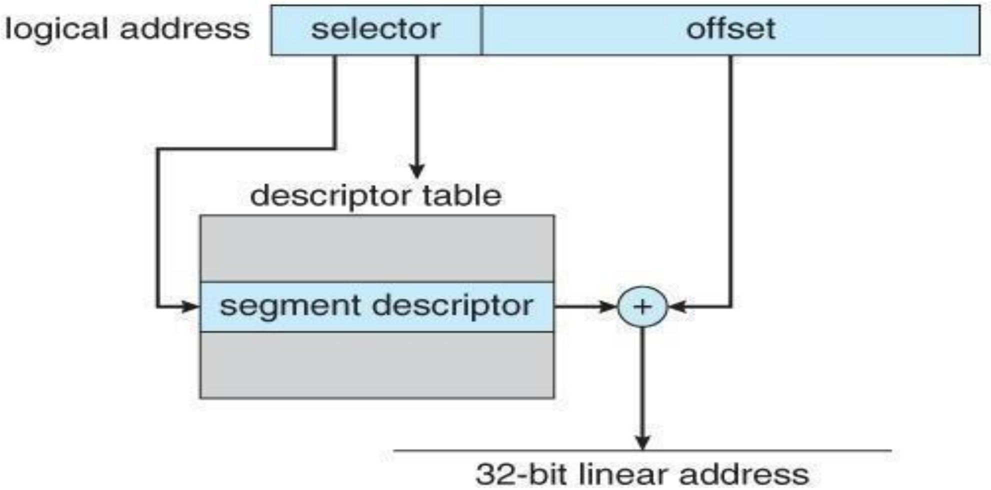
8Kprivatetothatparticularprocess,storedintheLocalDescriptorTable,LDT. 8K shared among all processes, stored in the Global Descriptor Table, GDT.

Logicaladdressesare(selector,offset)pairs,wheretheselectorismadeupof16bits: A 13 bit segment number (up to 8K)

A1bitflagforLDTvs.GDT. 2 bits for protection codes.

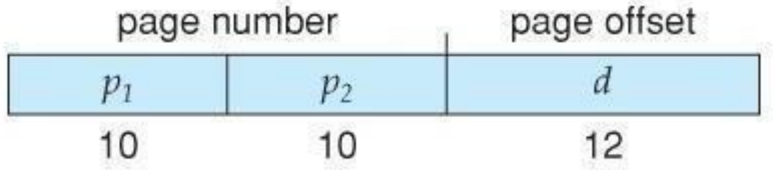


Thedescriptortablescontain8-bytedescriptionsofeachsegment,includingbaseandlimitregisters. Logical linear addresses are generated by looking the selector up in the descriptor table and adding the appropriate base address to the offset.



IA-32Paging

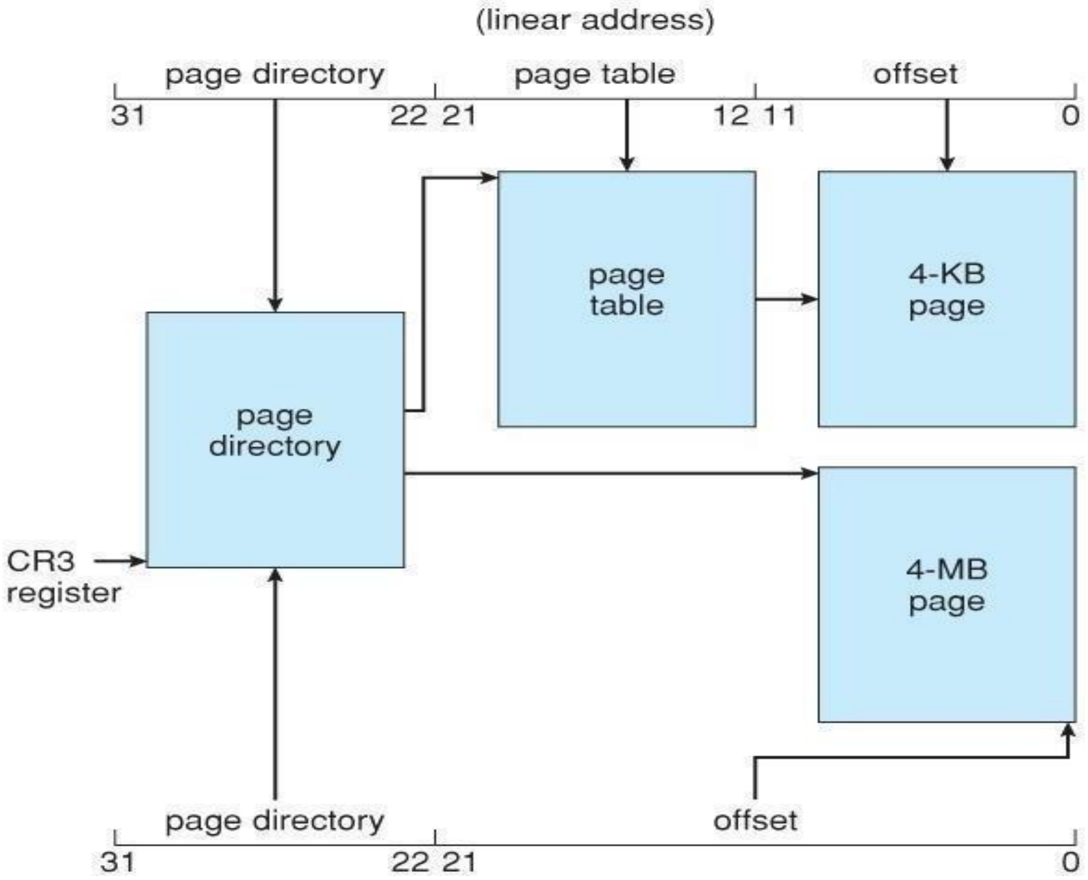
Pentium paging normally uses a two-tier paging scheme, with the first 10 bits being a page number for an outer page table (a.k.a. page directory), and the next 10 bits being a page number within one of the 1024 inner page tables, leaving the remaining 12 bits as an offset into a 4K page.



A special bit in the page directory can indicate that this page is a 4MB page, in which case the remaining 22 bits are all used as offset and the inner tier of page tables is not used.

TheCR3registerpoints tothepagedirectory forthecurrent process.

If the inner page table is currently swapped out todisk, then the page directory will have an "invalid bit" set, and the remaining 31 bits provide information on where to find the swapped out page table on the disk.



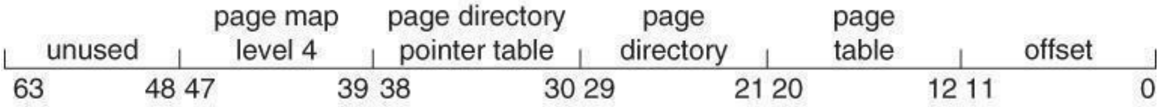
x86-64

TheinitialentryofInteldeveloping64-bitarchitectureswasthe IA-64(laternamed Itanium) architecture, but was not widely adopted.

 Meanwhile, AMD —begandevelopinga64-bitarchitectureknownasx86-64thatwas based on extending the existing IA-32 instruction set.

 The x86-64supportedmuchlargerlogicalandphysicaladdressspaces,aswellasseveral other architectural advances.

 Support for a 64-bitaddressspaceyieldsanastonishing264bytesofaddressable memory— a number greater than 16 quintillion (or 16 exabytes).



8. IRTUAL MEMORY

o Itisatechniquethatallowstheexecutionofprocessesthatmaynotbecompletelyinmain memory.

Virtualmemoryistheseparationofuserlogicalmemoryfromphysicalmemory.This separation allows an extremelylarge virtual memoryto be provided for programmers when only a smaller physical memory is available.

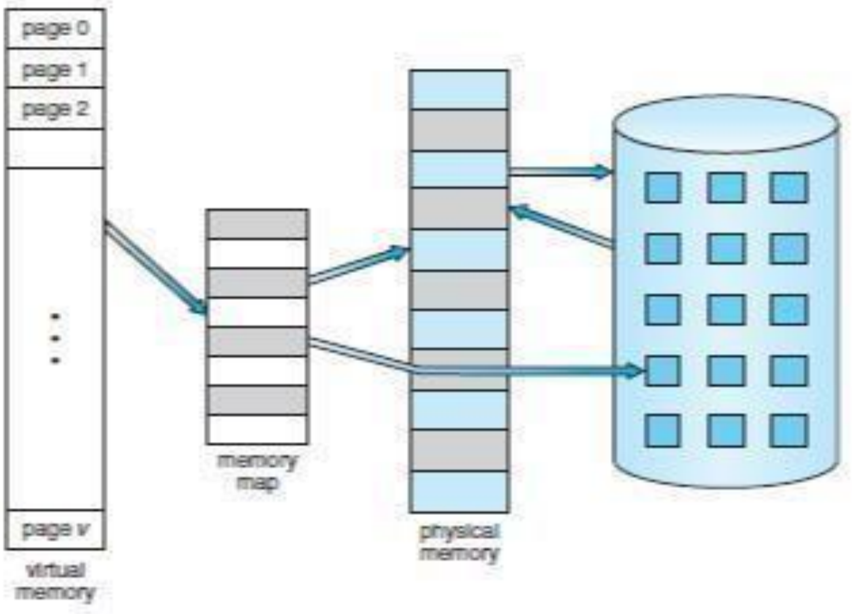
– Onlypart ofthe programneeds tobeinmemory forexecution.

– Logicaladdress spacecanthereforebemuchlargerthanphysicaladdressspace.

– Needtoallowpagestobeswappedinandout.

o **Advantages:**

□ Allowstheprogramthatcanbelargerthanthephysical memory.
□ Separationofuserlogicalmemoryfromphysicalmemory
□ Allowsprocessestoeasilysharefiles&addressspace.
□ Allowsformoreefficientprocess creation.



- o Virtualmemorycanbeimplementedusing
  - □ Demandpaging
  - □ Demandsegmentation

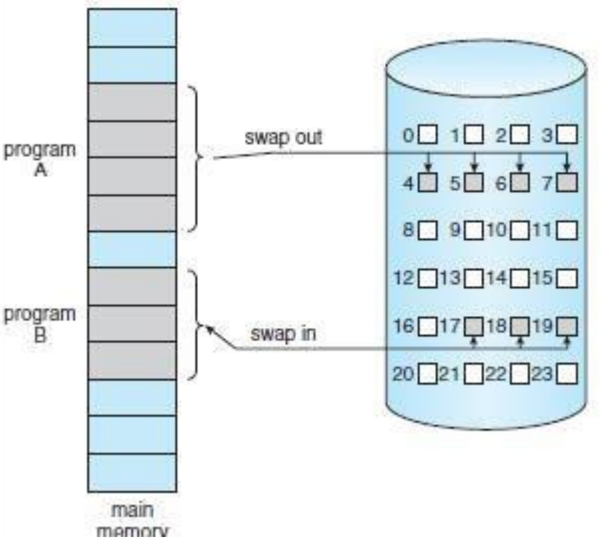
9. DEMANDPAGING

Concept

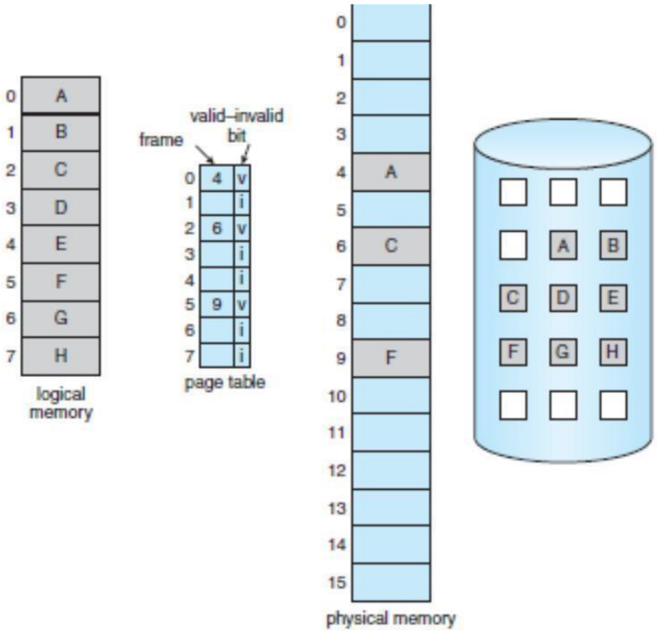
Thebasicideabehind demandpaging is that when aprocess is swapped in, its pages arenot swapped in all at once. Rather they are swapped in only when the process needs them (On demand). This is termed as lazy swapper.

Advantages

- □ LessI/Oneeded
- □ Lessmemoryneeded
- □ Fasterresponse
- □ Moreusers



Pagetablewhen somepagesarenot in mainmemory.



Theprocedureforhandlingthispagefault

1. Wecheckaninternaltable(usuallykeptwiththeprocesscontrolblock)forthisprocessto determine whether the reference was a valid or an invalid memory access.

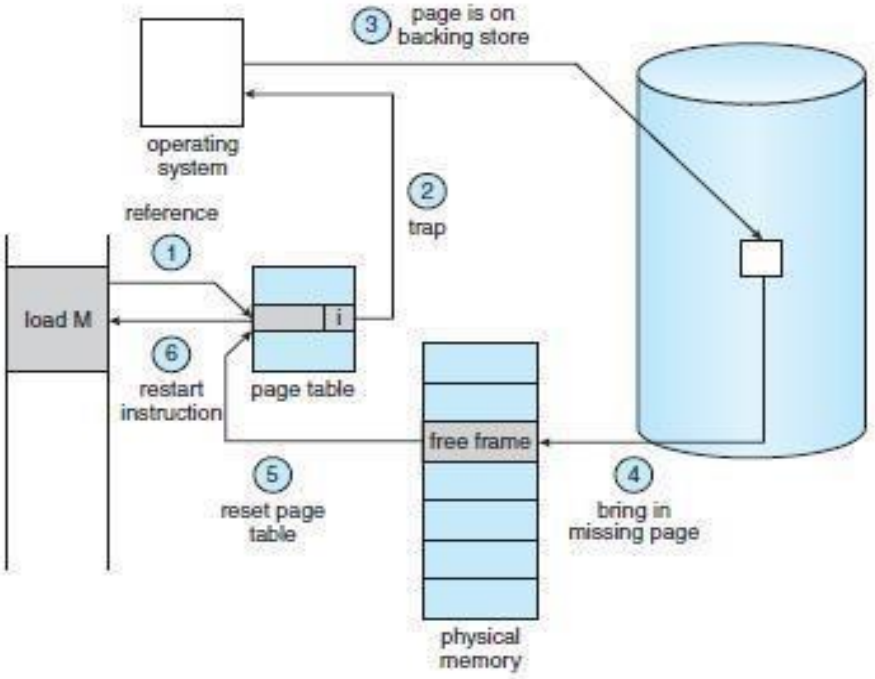
2. Ifthereferencewasinvalid,weterminatetheprocess. Ifitwasvalidbutwehavenotyetbrought in that page, we now page it in.

3. Wefindafreeframe(bytakingonefromthefree-framelist,forexample).

4. Wescheduleadisk operation toread thedesiredpageinto thenewlyallocated frame.

5. Whenthediskreadiscomplete,wemodifytheinternaltablekeptwiththeprocessandthepage table to indicate that the page is now in memory.

6. Werestarttheinstruction thatwasinterruptedbythetrap.Theprocess can nowaccessthepage as though it had always been in memory.



PerformanceofDemandPaging

EffectiveAccessTime(EAT)forademand-pagedmemory.
MemoryAccessTime(ma)formostcomputersnowrangesfrom10to200nanoseconds. If there is no page fault, then EAT = ma.
Ifthereis pagefault, then

EAT=(1–p)x(ma)+px(page-faulttime). p: the probability of a page fault ($0 \leq p \leq 1$), weexpectptobeclosetozero(afewpagefaults). If p=0 then no page faults, but if p=1 then every reference is a fault

Ifapagefaultoccurs,wemustfirstreadtherelevantpagefromdisk,andthenaccessthe desired word.

Example

Assumeanaveragepage-faultservicetimeof25milliseconds(10-3),andaMemoryAccessTime of 100 nanoseconds (10-9). Find the Effective Access Time?

Solution: EffectiveAccess Time(EAT)

= (1 – p)x (ma) +px (pagefault time)
= (1 – p) x100 +p x 25,000,000
=100 – 100 xp +25,000,000 xp
=100 +24,999,900 x p.

•Note:TheEffectiveAccessTimeisdirectlyproportionaltothepage-fault rate.

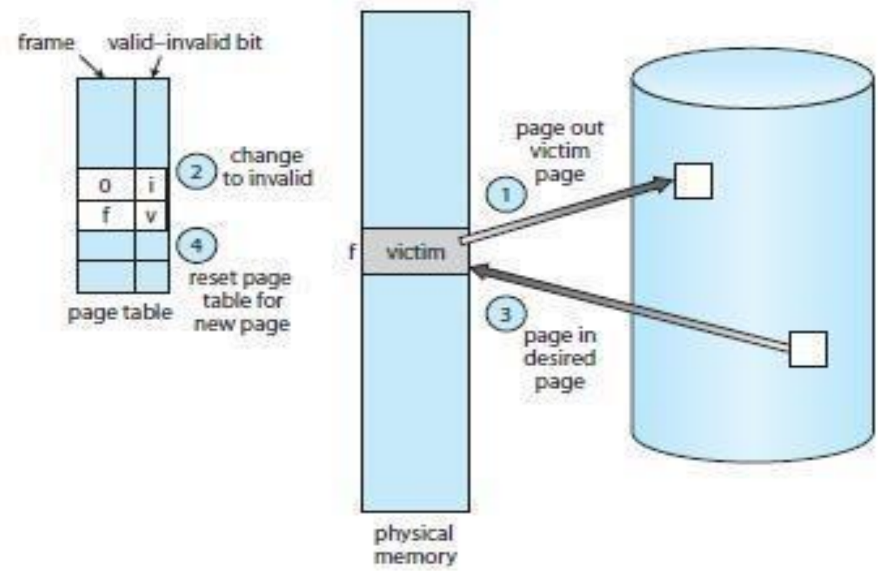
10.PAGEREPLACEMENT

Pagefault

Apagefaultisatypeofinterrupt,raisedbythehardwarewhenarunningprogram accesses a memory page that is mapped into the virtual address space, but not loaded in physical memory.

Needforpage replacement

Pagereplacementisneededtodecidewhichpageneededtobereplacedwhennew page comes in.



1. Findthelocation ofthedesiredpageonthe disk.
2. Findafreeframe:
   a. Ifthereisafreeframe,useit.
   b. Ifthereisnofreeframe,useapage-replacementalgorithmtoselect a victim frame.
   c. Writethevictimframetothedisk;changethepageandframetables accordingly.
3. Readthedesiredpageintothenewlyfreedframe;changethepageand frame tables.
4. Continuetheuserprocessfrom wherethepagefaultoccurred.

Pagereplacementalgorithms

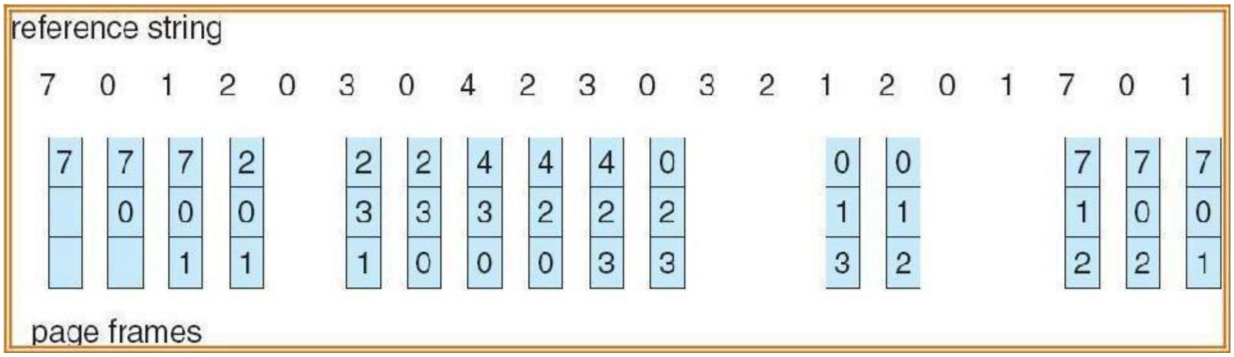
(a) FIFOpagereplacementalgorithm

Thisisthesimplestpagereplacementalgorithm.Inthisalgorithm,operatingsystemkeeps track ofall pages in thememory in aqueue, oldest pageis in thefront ofthequeue. When a page needs to be replaced page in the front of the queue is selected for removal.

Example:

Referencestring:7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

No.ofavailable frames = 3 (3pagescanbeinmemoryatatimeperprocess)

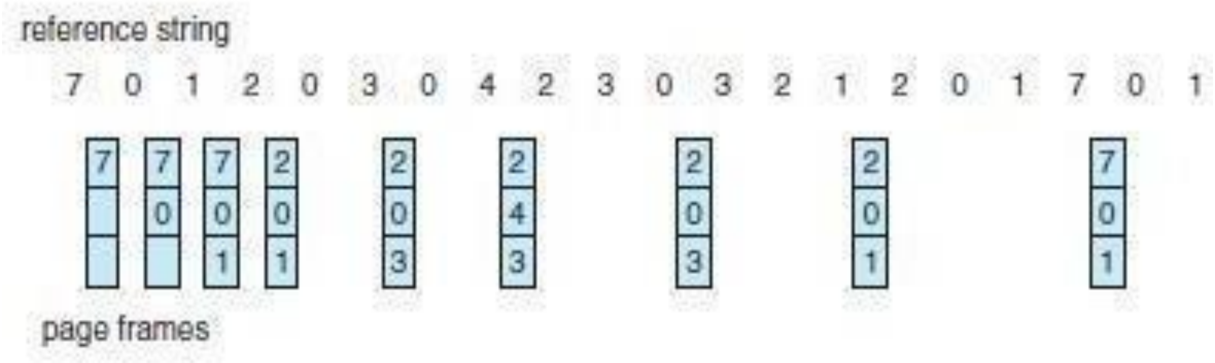


No.ofpagefaults = 15

(b) Optimalpagereplacement algorithm

Inthisalgorithm,pagesarereplacedwhicharenotusedforthelongestdurationoftime in the future.

Example:



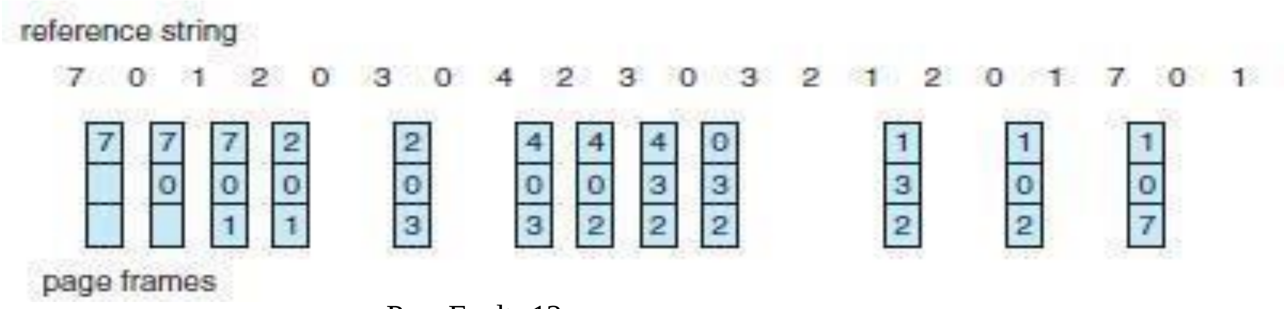
No.ofpagefaults =9

(c) LRU(LeastRecentlyUsed)pagereplacementalgorithm

Inthisalgorithmpagewillbereplacedwhichisleastrecentlyused.

Example:

Referencestring:7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1
No.ofavailableframes=3



PageFault=12

ImplementationofLRU

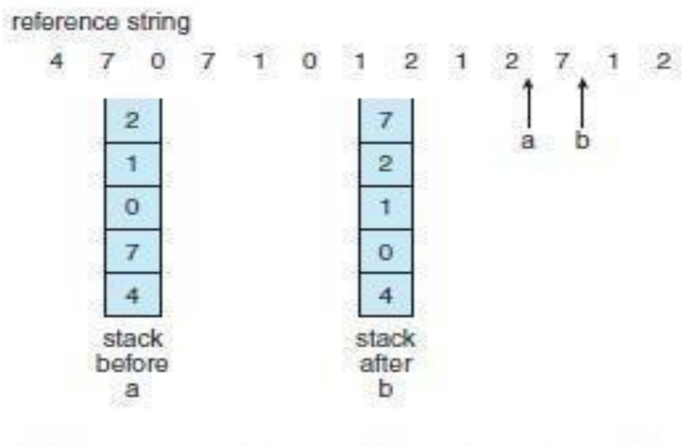
1. Counter

- □ Thecounterorclockisincrementedforevery memory reference.
- □ Eachtimeapageisreferenced ,copythecounterintothetime-of-usefield.
- □ Whenapageneedstobereplaced,replacethepagewiththesmallest counter value.

2. Stack

- □ Keepastackofpagenumbers
- □ Wheneverapageisreferenced,removethepagefromthestackandputit on top of the stack.
- □ Whenapageneedstobereplaced,replacethepagethatisatthebottom of the stack.(LRU page)

UseofAStack toRecordTheMostRecentPageReferences



(d) LRUApproximation PageReplacement

o Referencebit

□ Witheachpageassociate areferencebit,initially setto0

□ Whenpageisreferenced,thebitissetto 1

o Whenapageneeds tobe replaced,replace thepagewhosereferencebitis 0

o Theorderofuseisnotknown,butweknowwhichpageswereusedand whichwerenot used.

(i) AdditionalReferenceBits Algorithm

o Keepan8-bit byteforeach pageinatablein memory.

o Atregularintervals ,atimerinterrupttransferscontroltoOS.

o TheOSshiftsreferencebitforeachpageintohigher-orderbitshifting the other bits right 1 bit and discarding the lower-order bit.

Example:

oIfreferencebitis00000000thenthepagehasnotbeenusedfor8timeperiods.

oIfreferencebitis11111111thenthepagehasbeenusedatleastonceeach time period.

oIf the reference bit of page 1 is 11000100 and page 2 is 01110111 then page 2 is the LRU

page.

(ii) SecondChanceAlgorithm

oBasicalgorithmisFIFO

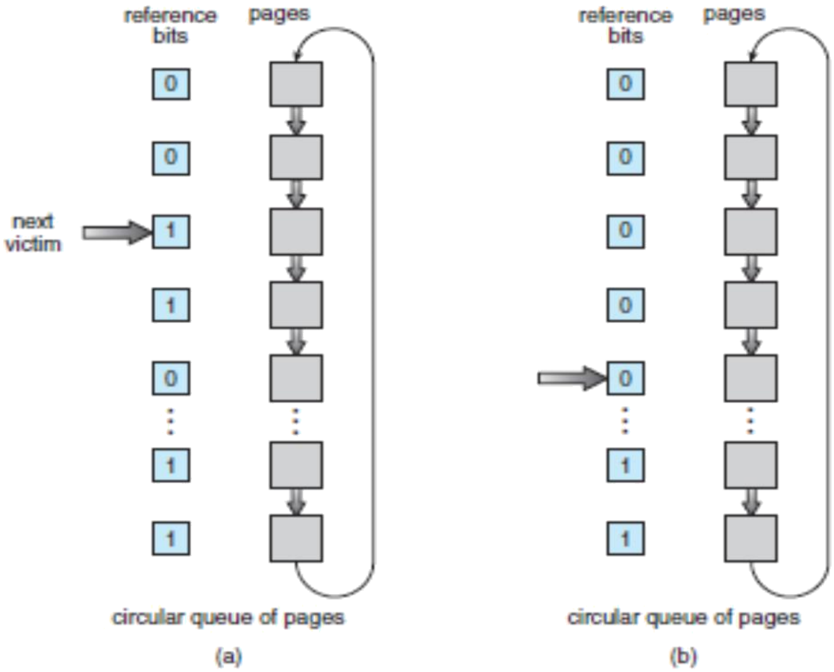
oWhenapagehasbeenselected,checkitsreference bit.

□ If0proceedtoreplacethepage

□ If1 givethepageasecond chanceand move ontothenext FIFO page.

□ Whenapagegetsasecondchance,itsreferencebitisclearedand arrival time is reset to current time.

□ Henceasecondchancepagewillnotbereplaceduntilallother pages are replaced.



(iii) EnhancedSecondChanceAlgorithm

o Consider both reference bit and modify bit o There are four possible classes

1. (0,0)–neither recentlyused normodified estpagetoreplace
2. (0,1)–notrecentlyusedbutmodifiedpagehastobewrittenout before replacement.
3. (1,0)-recently used butnot modified pagemay beused again
4. (1,1)–recentlyusedandmodifiedpagemaybeusedagainand page has to be written to disk

(iv) Counting-BasedPageReplacement

o Keepa counterofthenumberofreferencesthathavebeenmadetoeachpage

1. **LeastFrequently Used (LFU)Algorithm**: replaces page withsmallestcount
2. **MostFrequentlyUsed (MFU)Algorithm**: replaces page withlargestcount
   - □ isbasedontheargumentthatthepagewiththesmallestcountwas probably just brought in and has yet to be used

11. ALLOCATIONOF FRAMES

Allocation ofFrames

o Therearetwomajorallocationschemes

□ EqualAllocation

□ ProportionalAllocation

Equal allocation

- □ Iftherearenprocessesandmframesthenallocatem/nframes to each process.
- □ **Example:**Ifthereare5processesand100frames,giveeach process 20 frames.

- □ Allocateaccordingtothesizeofprocess

Letsibethesizeofprocessi. Let

m be the total no. of frames

Then S = ∑ si

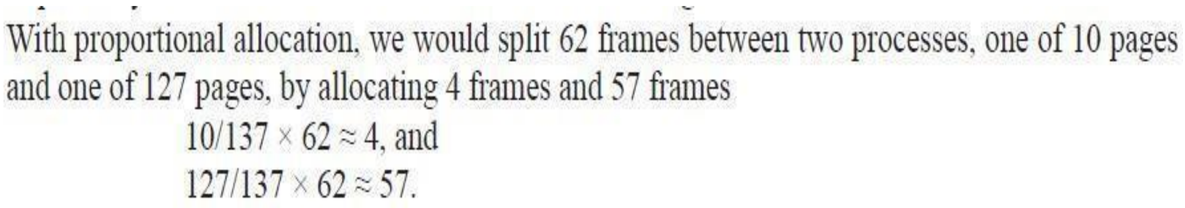
ai=si/S*m

whereaiistheno.offramesallocatedtoprocessi.

<u>Globalvs.Local Replacement</u>

o **Globalreplacement**–eachprocessselectsareplacementframefromthesetof all frames; one process can take a frame from another.

o **Localreplacement**–eachprocessselectsfromonlyitsownsetofallocatedframes.



With proportional allocation, we would split 62 frames between two processes, one of 10 pages and one of 127 pages, by allocating 4 frames and 57 frames

10/137 × 62 ≈ 4, and

127/137 × 62 ≈ 57.

12.THRASHING

Thrashing

o Highpagingactivity iscalled**thrashing**.

o Ifaprocessdoesnothaveenoughpages,thepage-faultrateisvery high.

This leads to:

- □ low CPU utilization
- □ operatingsystemthinksthatitneeds toincreasethedegreeof multiprogramming

- □ anotherprocess is addedtothesystem

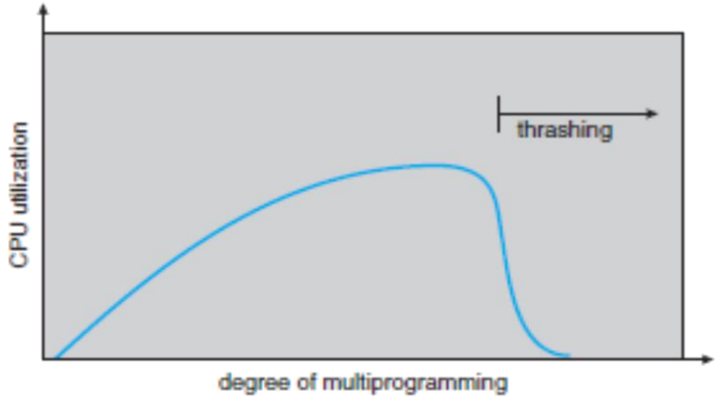
o WhentheCPUutilization islow,theOSincreasesthedegreeof multiprogramming.

oIfglobalreplacementisusedthenasprocessesenterthemainmemorytheytendtosteal frames belonging to other processes.

o Eventuallyallprocesseswillnothaveenoughframesandhencethepagefaultratebecomes very high.

o Thusswappinginandswappingoutofpagesonlytakesplace.

o Thisis thecauseofthrashing.



o To**limitthrashing**,wecanusea**localreplacement** algorithm. o To prevent thrashing, there are two methods namely ,

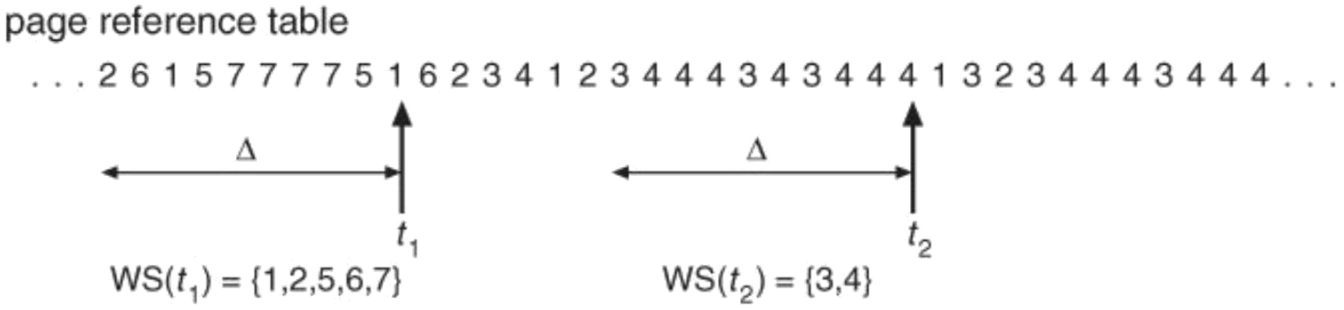
- □ WorkingSetStrategy
- □ PageFaultFrequency

1. Working-SetStrategy

o Itisbased onthe assumption ofthe modelof locality.

o Localityisdefinedasthesetofpagesactivelyusedtogether.

o Whateverpagesareincludedinthemostrecentpagereferencesaresaidtobeinthe processes working set window, and comprise its current working set .



Ifapageisinactiveuse, itwillbeintheworkingset.Ifitisnolongerbeingused,itwilldrop from the working set time units after its last reference.

- □ Thus, the working set is an approximation of the program's locality.
- □ if Δ = 10 memory references, then the working set at time t1 is {1, 2, 5,6, 7}.
- □ By time t2, the working set has changed to {3, 4}.
- □ The accuracy of the working set depends on th eselectionof.
- □ If Δ is too small, it will not encompass the entire locality; if is too large, it may overlap severallocalities.
- □ In the extreme, if Δ is infinite, the working set is the set of pages touched during the process execution.

 Themostimportant property oftheworkingset,then, isits size.

 If we compute the working-setsize,WSSi ,foreachprocessinthesystem,wecanthen considerthat

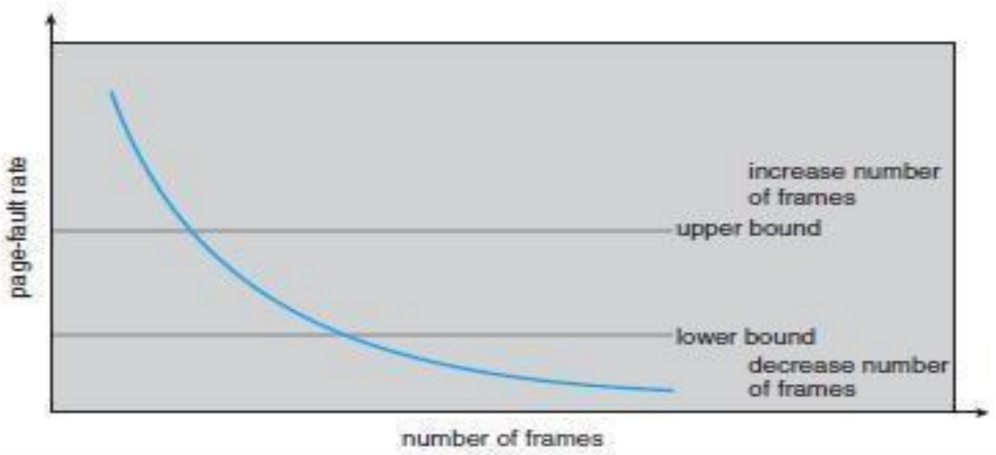
$$D = \sum WSS_i,$$

where D isthetotaldemandforframes.Eachprocessisactivelyusingthepagesin its working set.

 Thus, process i needs WSSi frames. If the total demand is greater than the total number of availableframes(D>m),thrashingwilloccur,becausesomeprocesseswill nothaveenough frames.

2. Page-FaultFrequencyScheme

- ➢ Thrashinghasahighpage-faultrate.Thus,wewanttocontrolthepage-faultrate.
- ➢ Whenitistoohigh,weknowthattheprocessneedsmoreframes.Conversely,ifthepage-faultrateistoo low,thentheprocessmayhavetoomanyframes.
- ➢ Wecanestablishupperandlowerboundsonthedesiredpage-faultrate.
- ➢ Iftheactualpage-faultrateexceedstheupperlimit,weallocatetheprocessanotherframe.
- ➢ Ifthepage-faultratefallsbelowthelowerlimit,weremoveaframefromtheprocess.
- ➢ Thus,wecandirectlymeasureandcontrolthepage-faultratetoprevent thrashing.



13.ALLOCATINGKERNEL MEMORY

AllocatingKernelMemory

When a process running in user mode requests additional memory, pages are allocated from the list of free page frames maintained by the kernel. This list is typically populated using a page-replacement algorithm such as those discussed in Section 9.4 and most likely contains free pages scattered throughout physical memory, as explained earlier. Remember, too, that if a user process requests a single byte of memory, internal fragmentation will result, as the process will be grantedan entire page frame.

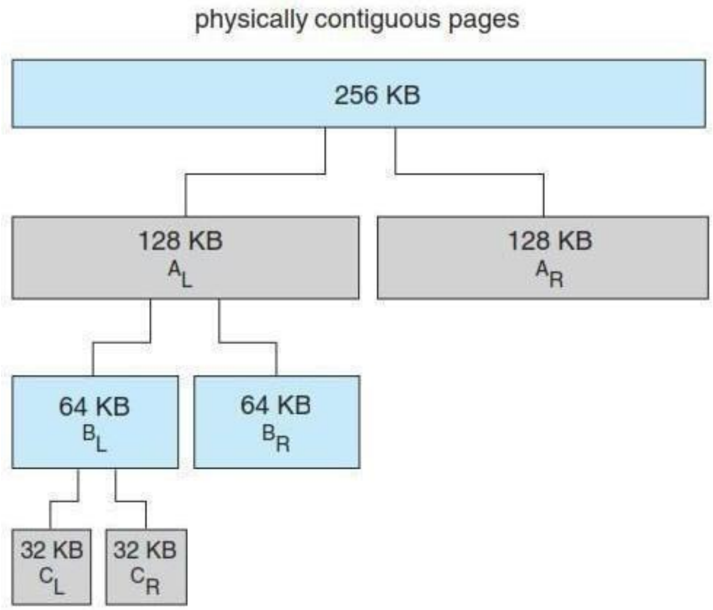
Kernel memory is often allocated from a free-memory pool different from the list used to satisfy ordinary user-mode processes. There are two primary reasons for this:

1. The kernel requests memory for data structures of varying sizes, some of which are less thanapagein size. As a result, thekernel must usememory conservatively and attempt to minimize waste due to fragmentation. This is especially important because many operating systems do not subject kernel code or data to the paging system.

2. Pagesallocatedtouser-modeprocessesdonotnecessarilyhavetobeincontiguousphysical memory. However, certain hardware devices interact directly with physical memory— without the benefitofavirtualmemoryinterface—andconsequentlymayrequirememoryresiding inphysically contiguous pages.

BuddySystem

The buddy system allocates memory from a fixed -size segment consisting of physically contiguous pages. Memory is allocated from this segment using a **power-of-2 allocator**, which satisfies requests in units sized as a power of 2 (4KB, 8KB, 16KB, and so forth). A request in units notappropriatelysizedisroundeduptothenexthighestpowerof2.Forexample,arequestfor11KB is satisfied with a 16K segment



Let'sconsiderasimpleexample.Assumethesizeofamemorysegmentisinitially256KBandthe kernel requests 21 KB of memory.

Thesegmentisinitiallydividedintotwobuddies—whichwewillcallALandAR—each128KBin size. One of these buddies is further divided into two 64-KB buddies— BL and BR.

However,thenext-highestpowerof2from21KBis32KBsoeitherBLorBRisagaindividedinto two 32-KB buddies, CL and CR. One of these buddies is used to satisfy the 21-KB request.

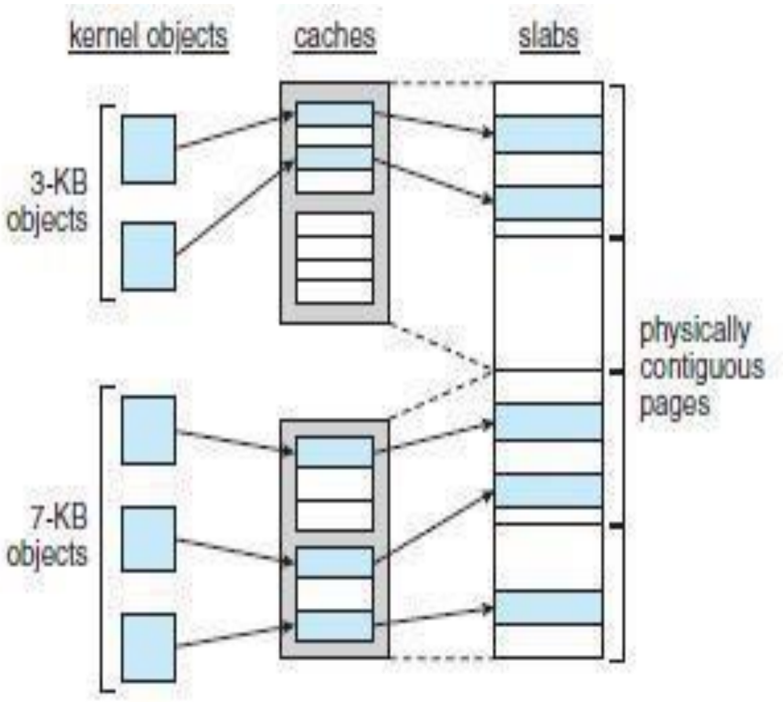
SlabAllocation

Asecondstrategyforallocatingkernelmemoryisknownasslaballocation.Aslabismadeupofone or more physically contiguous pages. A cache consists of one or more slabs.

Theslab-allocationalgorithmusescachestostorekernelobjects.

Whenacacheiscreated,anumberofobjectswhichareinitiallymarkedasfreeareallocatedtothe cache. The number of objects in the cache depends on the size of the associated slab.

Forexample,a12-KB slab(madeupofthree contiguous 4-KBpages)couldstoresix2-KB objects.



InLinux,aslabmay bein oneofthreepossible states:

1. Full.All objectsintheslabaremarkedasused.
2. Empty.Allobjectsintheslabaremarkedasfree.
3. Partial.Theslabconsists ofbothusedandfree objects.

Theslaballocatorfirstattemptstosatisfytherequestwithafreeobjectinapartialslab. If none

exists, a free object is assigned from an empty slab.

Ifnoemptyslabsareavailable,anewslabisallocatedfromcontiguousphysicalpagesandassigned to a cache; memory for the object is allocated from this slab.

14.SEGMENTATIONWITHPAGING

- TheIBMOS/2.32bitversionisanoperatingsystemrunningontopoftheIntel386architecture. The 386 uses segmentationwith paging formemory management. The maximum number of segments per process is 16 KB, and each segment can be as large as 4 gigabytes.
- Thelocal-addressspaceofaprocessisdivided intotwopartitions.

  Thefirstpartitionconsistsofupto8KBsegmentsthatareprivateto thatprocess.

  The second partition consists of up to 8KB segments that are sharedamong all the processes.
- Informationabout the firstpartitioniskept in the **local descriptortable (LDT)**, information about the second partition is kept in the **global descriptor table (GDT)**.
- EachentryintheLDTandGDTconsistof8bytes,withdetailedinformationabout a particular segment including the base location and length of the segment.

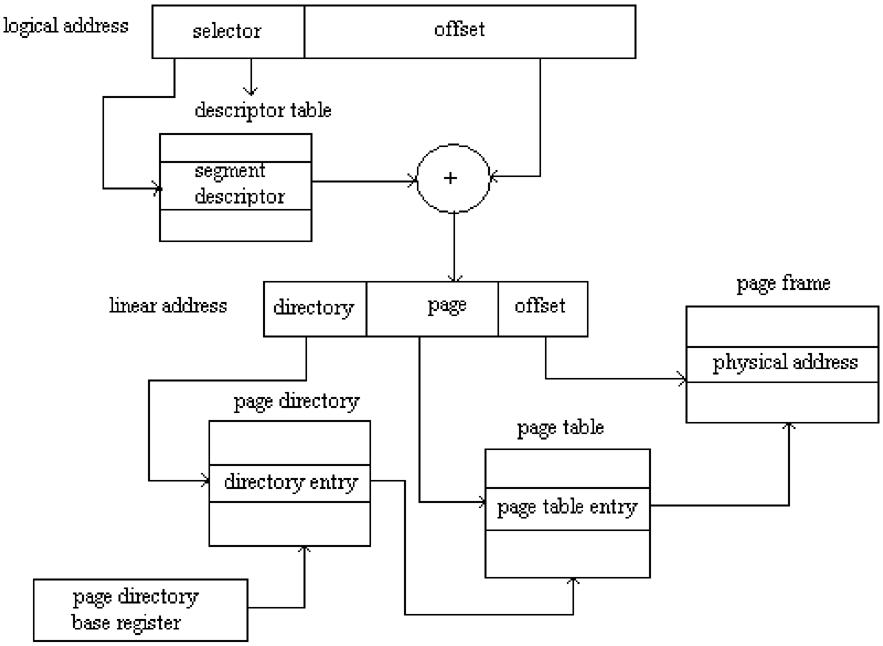
Thelogicaladdressisapair(selector,offset)wheretheselectorisa16-bitnumber:

s	g	p
13	1	2

Wheresdesignatesthesegmentnumber,gindicateswhetherthe segmentisintheGDTorLDT,andpdealswithprotection.Theoffsetisa32-bitnumberspecifying the location of the byte within the segment in question.

- The base and limit information about the segment in question are used to generate a linear-address.
- First, the limit is used to check for address validity. If the address is not valid, a memory fault is generated,resultinginatraptotheoperatingsystem.Ifitisvalid,thenthevalueoftheoffset isaddedtothevalueofthebase,resultingina32-bitlinearaddress.Thisaddressisthentranslated into a physical address.
- Thelinearaddressisdividedintoapagenumberconsistingof20bits,andapageoffsetconsisting of12bits.Sincewepagethepagetable,thepage number is further divided intoa10-bit pagedirectory pointeranda10-bit pagetablepointer.Thelogicaladdressisasfollows.

p1	p2	d
10	10	12



oToimprovetheefficiencyofphysicalmemoryuse.Intel386pagetablescan beswappedtodisk.Inthiscase,aninvalidbitisusedinthepagedirectoryentry to indicate whether the table to which the entry is pointing is in memory or on disk.

oIf the table ison disk, the operatingsystem canuse the other 31 bits to specify thedisklocation ofthetable;thetablethencanbebroughtintomemoryon demand.

UNITIVFILESYSTEMSANDI/OSYSTEMS

Mass Storage system – Overview of Mass Storage Structure, Disk Structure, Disk SchedulingandManagement,swapspacemanagement;File-SystemInterface– Fileconcept,Accessmethods, Directory Structure, Directory organization, File system mounting, File Sharing and Protection; File System Implementation- File System Structure, Directory implementation, Allocation Methods, Free Space Management, Efficiency and Performance, Recovery; I/O Systems – I/O Hardware, Application I/O interface, Kernel I/O subsystem,Streams, Performance.

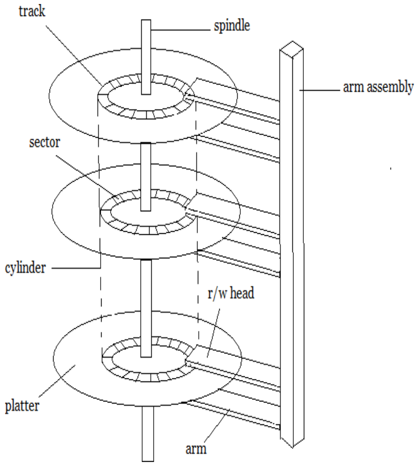
MASSSTORAGESTRUCTURE

1. OverviewofMassStorageStructure

Magnetic Disks

- Inmoderncomputers, mostof the secondarystorage is in theform of magnetic disks.
- Amagneticdiskcontainsseveralplatters.Eachplatterisdividedintocircular shaped tracks.
- The length of the tracks near the centre is less than the length ofthe tracks farther from the centre.
- Eachtrackisfurtherdividedintosectors.
- Tracksofthesamedistancefromcentreformacylinder.
- Aread-writeheadisusedtoreaddatafromasectorofthemagneticdisk.
- Thespeedofthedisk ismeasuredastwo parts:

Transferrate:Thisis therateatwhich thedatamovesfrom disk tothecomputer.
Randomaccess time:Itisthesumoftheseektimeandrotational latency.
Seek time is the time taken by the arm to move to the required track.
Rotationallatencyisdefinedasthetimetakenbythearmtoreachtherequired sector in the track.



Solid-StateDisks

SDisnon-volatile memorythatisusedlike aharddrive.SSDshavethesame characteristicsastraditionalharddisksbutcanbemorereliablebecausetheyhaveno moving parts and faster because they have no seek time or latency. In addition, they consume less power.

SSDs have less capacity than the larger hard disks, and may have shorter life spans. use for SSDs is in storage arrays, where they hold file- system metadata that requirehighperformance.SomeSSDsaredesignedtoconnectdirectlytothesystembus.

Magnetic Tapes

Magnetic tape was used as an early secondary-storage medium. Although it is relatively permanent and can hold large quantities of data, its access time is slow compared with that of main memory and magnetic disk.

Inaddition,randomaccesstomagnetictapeisaboutathousandtimesslowerthan random access to magnetic disk, so tapes are not very useful for secondary storage.

2. Disk Structure

In Disk drives are addressed as large 1-dimensional arrays of logical blocks, wherethelogicalblockisthesmallestunitoftransfer.nThe1-dimensionalarrayof logical blocksis mapped into the sectors of the disk sequentially.

➔InSector0isthefirstsectorofthefirsttrackontheoutermostcylinder.

➔InMappingproceedsinorderthroughthattrack,thentherestofthetracksinthat cylinder, and then through the rest of the cylinders from outermost to innermost.

3. Disk Scheduling

Theoperatingsystemisresponsibleforusing hardwareefficiently.
Forthediskdrives,thismeanshaving afast accesstime&disk bandwidth.

➔Accesstimehastwo majorcomponents:

➔Seektimeisthetimeforthedisktomovetheheadstothecylindercontaining the desired sector

➔Rotationallatencytimewaitingforthedisktorotatethedesiredsectortothe disk head

➔Weliketominimizeseek time.

➔Diskbandwidthisthetotalnumberofbytestransferreddividedbythetotaltime between the first request for service and the completion of the last transfer.

➔Severalalgorithmsexist toschedulethe servicingofdiskI/Orequests.

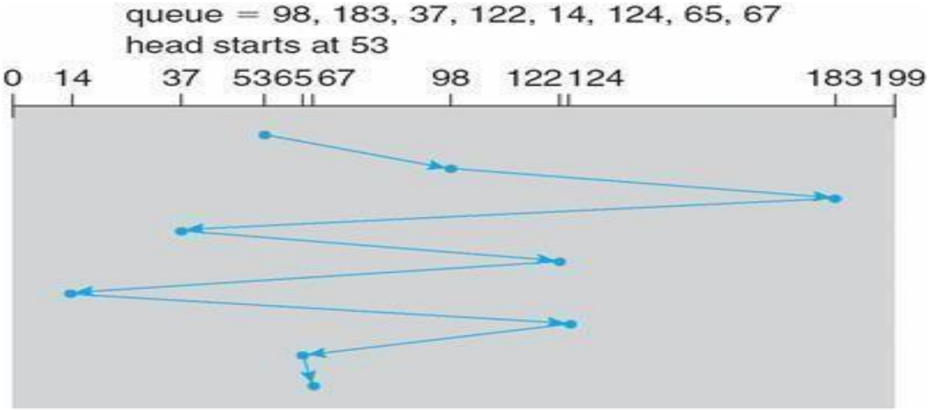
WeillustratethemwithaRequestQueue(cylinderrange0-199): 98, 183, 37, 122, 14, 124, 65, 67
Headpointer:cylinder53

1. FirstComeFirst Serve

Thisalgorithmperformsrequestsinthesameorderaskedbythesystem.Let's take an example where the queue has the following requests with cylinder numbers as follows:

98, 183, 37, 122, 14, 124, 65, 67

Illustrationshowstotalheadmovement of640cylinders

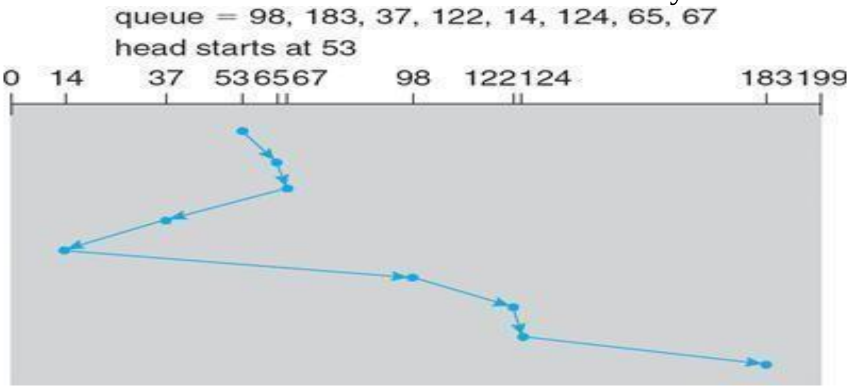


2. SSTF(ShortestSeekTimeFirst)

Selectstherequestwith theminimumseektimefromthecurrentheadposition.

SSTFschedulingisaform ofSJFscheduling;maycausestarvationofsome requests.

Illustrationshowstotalheadmovementof236cylinders.

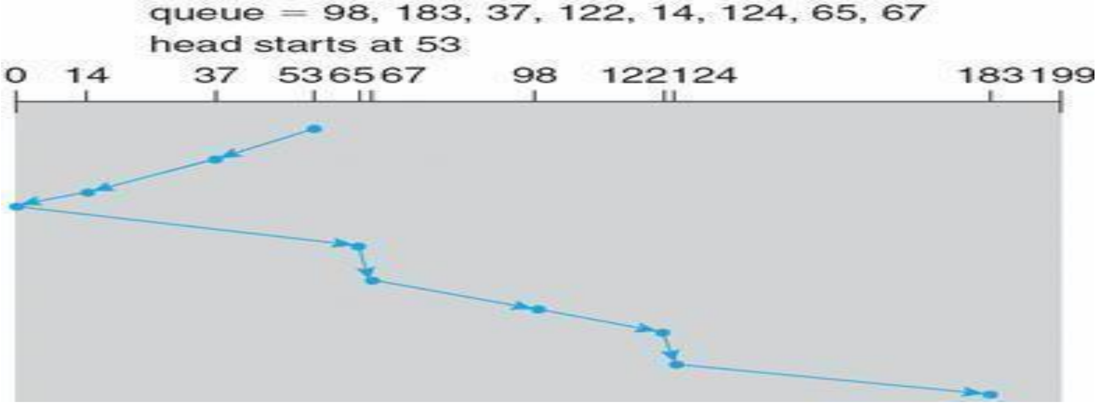


3. SCAN

Thediskarmstartsatoneendofthedisk,andmovestowardtheotherend,servicingrequests until it gets tothe other end of the disk, wherethe head movement is reversed and servicing continues.

SCANalgorithmsometimescalledtheelevatoralgorithm.

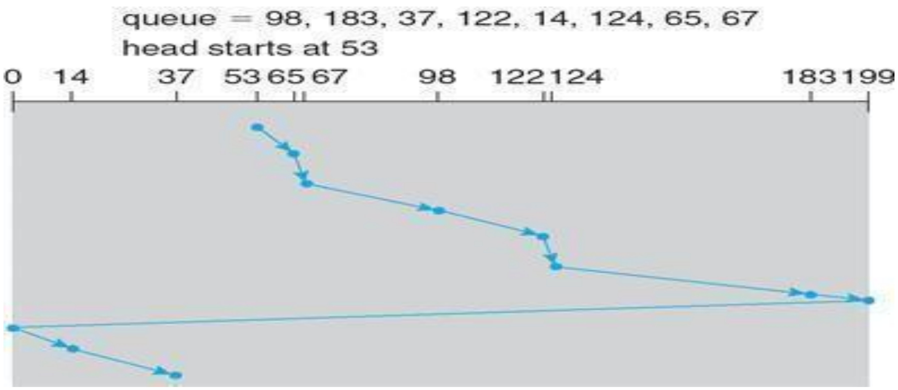
Illustration shows total head movement of 208 cylinders



4. C-SCAN

Providesamoreuniform waittimethanSCAN.Theheadmovesfromone endofthedisk to the other, servicing requests as it goes When it reaches the other end, however, it immediatelyreturnstothebeginningofthedisk,withoutservicinganyrequestsonthereturn trip.

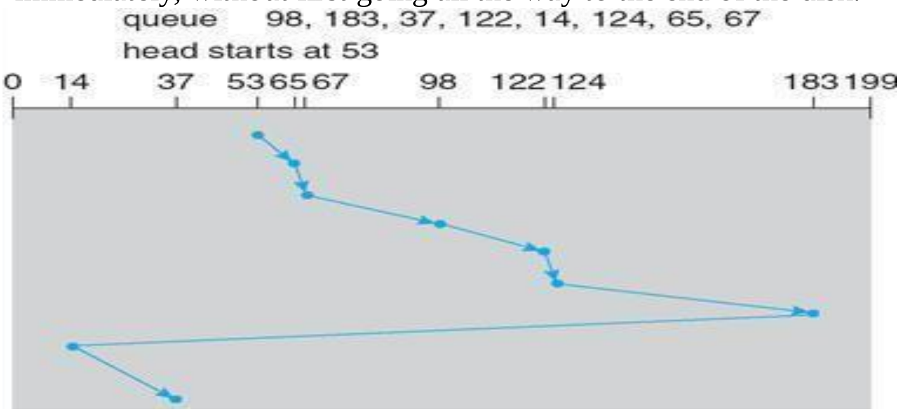
Treats the cylinders as a Circular list that wraps around from the last cylinder to the first one .



5. <u>LOOK</u>

➔VersionofC-SCAN

➔Armonlygoesasfarasthe**lastrequest**ineachdirection,thenreversesdirection immediately, without first going all the way to the end of the disk.



4. DiskManagement

Theoperatingsystemisresponsiblefordiskinitialization,bootingfromdisk,and bad-block recovery.

DiskFormatting

Anewmagneticdiskmustbedividedintosectorsthatthediskcontrollercanread and write. This process is called **low-level formatting, or physical formatting**. Low-levelformatting fills the disk with a special data structure for each sector. The data structure for a sector typically consists of a header, a data area (usually 512 bytesin size), and a trailer.

The header and trailer contain information used by the disk controller, such as a sector number and an **error-correcting code (ECC).**

This formatting enables the manufacturer to 1. Test the disk and 2. To initialize the mapping from logical block numbers

Touseadisktoholdfiles,theoperatingsystemstillneedstorecorditsown data structures on the disk.

Itdoessointwosteps.

(a) Thefirststepis**Partition**thediskintooneormoregroupsofcylinders. Amongthepartitions,onepartitioncanholdacopyoftheOS's executable code, while another holds user files.

(b) Thesecondstepis**logicalformatting**.Theoperatingsystemstoresthe initial file-system data structures onto the disk. These datastructures may includemapsoffreeand allocatedspace andan initial empty directory.

BootBlock

For a computer to start running-for instance, when it is powered up or rebooted-itneedstohaveaninitialprogramtorun.Thisinitialprogramiscalledbootstrapprogram & it should be simple.

Itinitializesallaspectsofthesystem,fromCPUregisterstodevicecontrollers and the contents of main memory, and then starts the operating system.

The bootstrap is stored in read-only memory (ROM). This location is convenient, because ROM needs no initialization and is at a fixed location that the processorcanstartexecutingwhenpowereduporreset.And,sinceROMisreadonly, it cannot be infected by a computer virus.

The full bootstrap program is stored in the "**boot blocks**" at a fixed locationon the disk. A disk that has a boot partition is called a boot disk or system disk. The work of boot block as follows

1. Findstheoperatingsystem kernel ondisk,
2. Loadsthatkernelintomemory,and
3. Jumpstoaninitialaddresstobegintheoperating-systemexecution.

The full **bootstrap program** is stored in a partition called the boot blocks, at a fixed location on the disk. A disk that has a boot partition is called a boot disk or system disk.

Thecode in the boot ROM instructs the disk controller to read the boot blocks into memory and then starts executing that code.

Bootstrap loader - load the entire operating system from a non-fixed location on disk, and to start the operating system running.

BadBlocks

Thediskwithdefectedsectoriscalledasbadblock.Dependingonthediskand controller in use, these blocks are handled in a variety of ways;

Method1:"Handledmanually‖

If blocks go bad during normal operation, a **special program** must be run manuallytosearchforthebadblocksandtolockthemawayasbefore.Datathatresided on the bad blocks usually are lost.

Method2:"sectorsparingorforwarding"

The controller maintains a list of bad blocks on the disk. Then the controller can be told to replace each bad sector logically with one of the spare sectors. This scheme is known as sector sparing or forwarding.

Atypicalbad-sector transactionmightbeasfollows:

- Theoperating systemtriestoreadlogical block87.
- Thecontrollercalculates the ECC and findsthatthesectoris bad.
- Itreportsthis findingtotheoperatingsystem.
- Thenexttimethatthesystemisrebooted,aspecialcommandisruntotell the controllertoreplacethe badsectorwitha spare.
- Afterthat,wheneverthesystemrequestslogicalblock87,therequestis translated into the replacement sector's address by the controller.

Method3:"sectorslipping"

For an example, suppose that logical block 17 becomes defective, and the first available spare follows sector 202. Then, sector slipping would remap all the sectors from 17 to 202, moving them all down one spot. That is, sector 202 would be copied into the spare, then sector 201 into 202, and then 200 into 201, and so on, until sector 18iscopiedintosector19.Slippingthesectors in thiswayfrees upthespaceofsector 18, so sector 17 can be mapped to it.

5. **Swap-SpaceManagement**

➔Swap-space—virtualmemoryusesdiskspaceasanextensionofmainmemory.

➔Maingoalforthedesignandimplementationofswapspaceistoprovidethebest throughput for VM system

1. **Swap-space use**

Swapping –use swap space to hold entire process image

Paging–storepagesthathavebeenpushedoutofmemory

➔SomeOSmaysupportmultipleswap-space

–Putonseparatediskstobalancethe load

➔Bettertooverestimatethan underestimate

–Ifoutofswap-space,someprocessesmustbeabortedorsystemcrashed

2. **Swap-SpaceLocation**

➔Swap-spacecanbecarvedoutofthenormalfilesystem,orinaseparatedisk partition

➔Alargefilewithinthefilesystem: simple but inefficient

–Navigatingthedirectorystructureandthedisk-allocationdata structure takes time and potentially extra disk accesses

–Externalfragmentationcangreatlyincreaseswappingtimesby forcing multiple seeks during reading or writing of a process image

–Improvement

Cachingblocklocationinformationinmainmemory

•Contiguousallocationfortheswapfile

But,thecostoftraversingFS datastructurestillremains

➔Inaseparatepartition:raw partition

–Createaswap spaceduringdisk partitioning

–Aseparateswap-spacestoragemanagerisusedtoallocateandde-allocate blocks

–Usealgorithms optimizedforspeed,ratherthanstorageefficiency

–Internalfragmentmayincrease

Linuxsupports both approaches

➔Swap-spaceManagement:Example

Solaris 1

–Text-segmentpagesarebroughtinfromthefilesystemandare thrown away if selected for paged out

Moreefficienttore-read fromFSthanwriteittotheswapspace

-Swapspace:onlyusedasabackingstoreforpagesofanonymous memory

Stack,heap,anduninitializeddata

Solaris 2

–Allocatesswapspaceonlywhenapageisforcedoutofphysical memory

Notwhenthe virtualmemorypageisfirst created.

FILESYSTEMINTERFACE

1. FileConcepts

Afileisanamedcollectionofrelatedinformationthatisrecorded onsecondary storage. From user’s perspective a , a file is the smallest allotmentofthat logical secondary storage; unless they are within a file.

Commonly,filesrepresentprograms(bothsourceandobjectforms)and data. Data files may be numeric, alphabetic, alphanumeric, or binary.

Ingeneral,afileisasequenceofbits,bytes,lines,orrecords,themeaningofwhich isdefined by the file’s creator and user.

A **text file** is a sequence of characters organized into lines (and possibly pages). An **executable** file is a series of code sections that the loader can bring into memory and execute.

2. FileAttributes

Theinformationaboutallfilesiskeptinthedirectorystructure,adirectoryentry consistsofthefile’snameanditsuniqueidentifier.Theidentifierinturnlocatestheother file attributes.

- **Name:** The symbolic file name isthe onlyinformation keptinhuman readable form.

- **Identifier:** This unique tag, usually a number identifies the file within the file system. It is the non-human readable name for the file.
- **Type:**Thisinformation isneededforthosesystems thatsupportdifferenttypes.
- **Location:** This information is a pointer to a device and to the location of the fileon that device.
- **Size:**Thecurrentsize ofthefile(inbytes,wordsorblocks)andpossiblythe maximum allowed size are included in this attribute.
- **Protection:**Access-controlinformationdetermineswhocandoreading, writing, executing and so on.
- **Time, date and user identification:** This information may be kept for creation,lastmodificationandlastuse.Thesedatacanbeusefulforprotection, security and usage monitoring.

3. **FileOperations**

Theoperating system can providesystem calls to create, write, read, reposition, delete, and truncate files.

Creating a file - First, spacein the file system must befoundforthe file, Second, an entry for the new file must be made in the directory.

Writingafile -System callspecifyingboth thenameofthe fileandtheinformation to be written to the file.

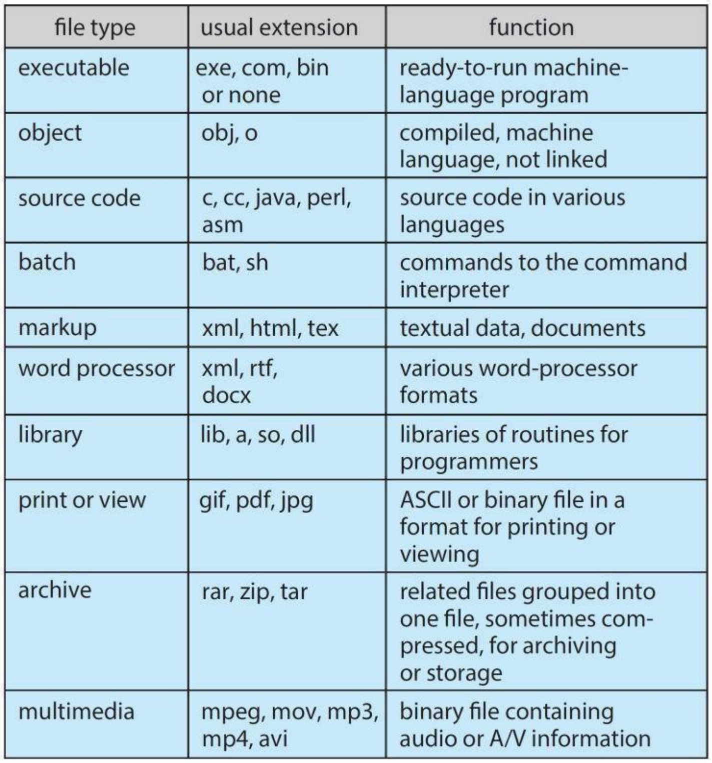
Readingafile-weuseasystemcallthatspecifiesthenameofthefileand where (in memory) the next block of the file should be put.

Repositioningwithinafile-Thedirectoryissearchedfortheappropriate entry, and the current-file-position pointer is repositioned to a given value.

Deletingafile-searchthedirectoryforthenamedfile.Havingfoundthe associated directory entry, we release all file space

Truncating a file - this function allows all attributes to remain unchanged—except for file length—but lets the file be reset to length zero and its file space released.

4. FileTypes

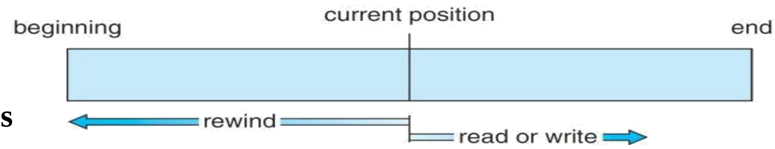


file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, perl, asm	source code in various languages
batch	bat, sh	commands to the command interpreter
markup	xml, html, tex	textual data, documents
word processor	xml, rtf, docx	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	gif, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	rar, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, mp3, mp4, avi	binary file containing audio or A/V information

5. AccessMethods

1. SequentialAccess

- Dataisaccessedonerecordrightafteranotherisanorder.
- Readcommand causeapointer tobemovedaheadby one.
- Writecommand allocatespacefortherecordand movethepointertothenewEnd Of File.
- Suchamethodisreasonablefor tape.



2. DirectAccess

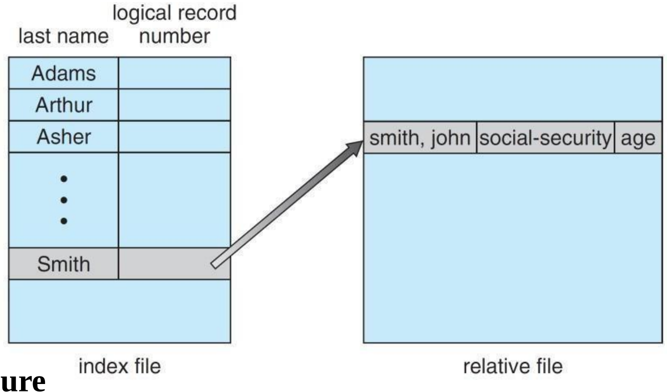
- Thismethod is useful fordisks.
- Thefileisviewed asanumberedsequence ofblocks or records.
- There are no restrictions on which blocks are read/written, it can be dobe in anyorder.
- Usernowsays"readn"ratherthan"readnext".
- "n"isanumberrelativetothebeginningoffile,notrelativetoanabsolute physical disk location.

As a simple example, on an **airline – reservation system**, we might store all the information about a particular flight (for example, flight 713) in the block identified by the flight number.

Thus, the number of available seats for flight 713 is stored in block 713 of the reservationfile.Tostoreinformationaboutalargerset,suchaspeople,wemight compute a hash function on the people's names, or search a small in-memory index to determine a block to read and search.

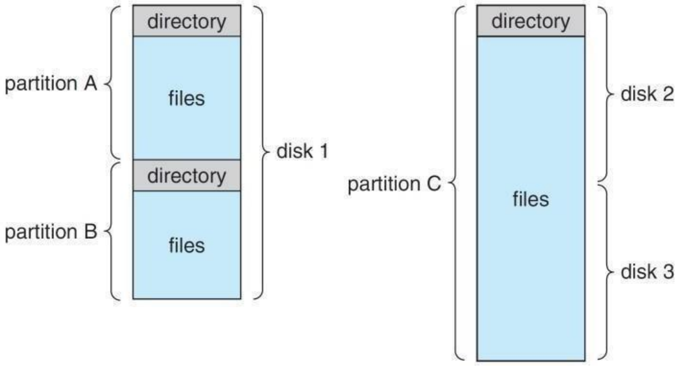
3. IndexedAccess

- ❖ Ifafilecan be sorted on any ofthe filed then an index can be assigned to agroup of certain records.
- ❖ However,Aparticular recordcanbeaccessed byitsindex.
- ❖ Theindexis nothing buttheaddressof arecordinthe file.
- ❖ In index accessing, searching in a large database became very quick and easy but we need to have some extra space in the memory to store the index value.



6. <u>DirectoryStructure</u>

- ❖ Directory can bedefinedasthelisting oftherelatedfilesonthedisk.
- ❖ Thedirectorymay storesomeortheentirefileattributes.
- ❖ Each partition must have at least one directory in which, all the files ofthe partition can be listed.
- ❖ Adirectoryentryismaintainedforeachfileinthedirectory whichstores all the information related to that file.



<u>Operationsthataretobeperformedonadirectory</u>

Search for a file. We need to be able to search a directory structure to find the entryforaparticularfile.Sincefileshavesymbolicnames,andsimilarnames may indicate a relationship among files, we maywant to be able to find all files whose names match a particular pattern.

Createafile.Newfilesneedtobecreated andaddedtothedirectory.
Delete a file. When a file is no longer needed, we want to be able to remove it from the directory.
List a directory. We need to be able to list the files in a directory and the contents of the directory entry for each file in the list.
Renamea file. Becausethenameofa filerepresents its contents to itsusers,we must be able to change the name when the contents or use of the file changes. Renaming a file may also allow its position within the directory structure to be changed.
Traverse the file system. We may wish to access every directory and every file within a directory structure. For reliability, it is a good idea to save the contents and structure of the entire file system at regular intervals.

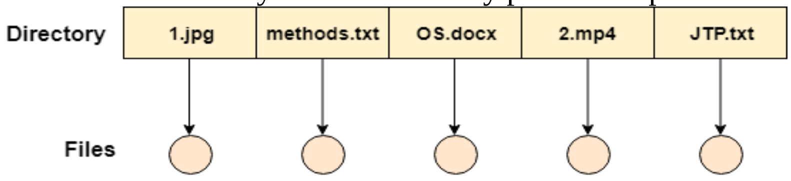
➔Often,wedo thisby copyingall files to magnetictape.Thistechnique provides a backup copy in case of system failure.
➔In addition, if a file is no longer in use, the file can be copied to tape and the disk space of that file released for reuse by another file.

LogicalStructure(or) LevelofDirectory

- Single-leveldirectory
- Two-leveldirectory
- Tree-Structureddirectory
- AcyclicGraphdirectory
- GeneralGraphdirectory

Single–Level Directory

❖ Thesimplestmethod isto haveonebiglistofallthefilesonthedisk.
❖ The entire system will contain only one directory which is supposed tomention all the files present in the file system.
❖ Thedirectorycontains oneentry pereachfilepresenton thefilesystem.

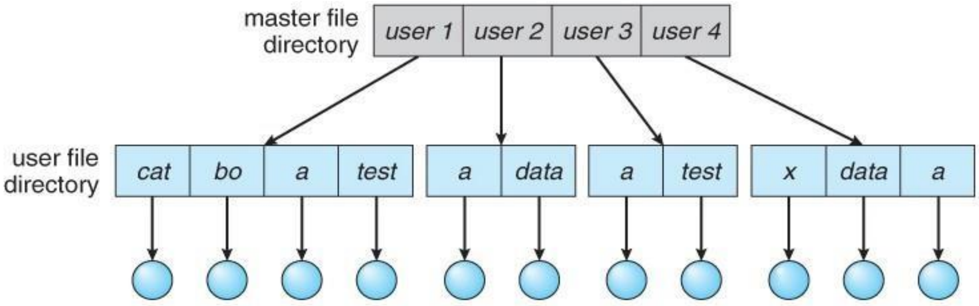


Single Level Directory

Disadvantages
1. Wecannot havetwofileswith thesame name.
2. The directory may be very big therefore searching for a file may take so much time.
3. Protectioncannotbeimplementedformultipleusers.
4. Therearenо waysto groupsamekind of files.

TwoLevelDirectory

- ❖ Intwoleveldirectory systems,wecancreateaseparatedirectoryforeachuser.
- ❖ There is one master directory which contains separate directories dedicated to eachuser.Foreachuser,thereisadifferentdirectorypresentatthesecondlevel, containing group of user's file.
- ❖ The system doesn't let a user to enter in the other user's directory without permission.

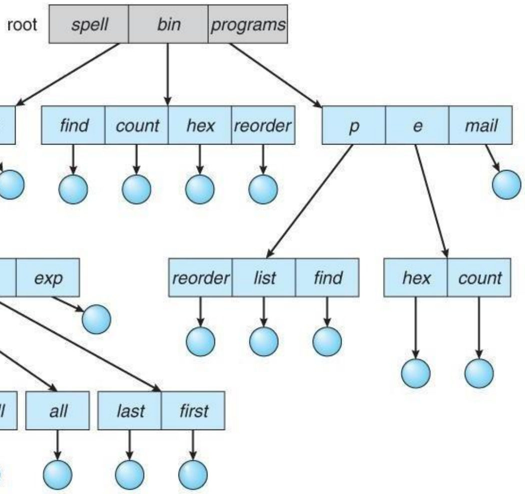


Characteristicsoftwoleveldirectorysystem

1. Eachfileshas apathnameas/User-name/directory-name/
2. Differentuserscanhave thesamefilename.
3. Searchingbecomesmoreefficientasonlyoneuser'slistneedstobe traversed.

TreeStructuredDirectory

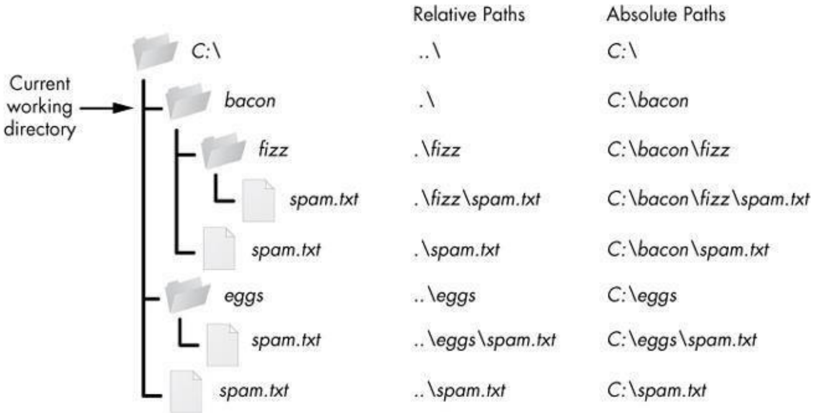
- ❖ Tree structured directory system overcomes the drawbacks of two level directory system.
- ❖ Thesimilar kindoffiles cannow begroupedin onedirectory.
- ❖ Eachuserhasitsowndirectoryanditcannotenterintheotheruser's directory.
- ❖ Searchingismoreefficientinthisdirectory structure



Afilecan beaccessedbytwo typesofpath,either➔ 1.Relative or2. Absolute.

1. **Absolutepath**is thepath ofthefilewithrespecttotherootdirectory ofthesystem.

2. **Relativepath**isthepathwithrespectto thecurrentworkingdirectoryofthesystem



Acyclic-GraphStructuredDirectories

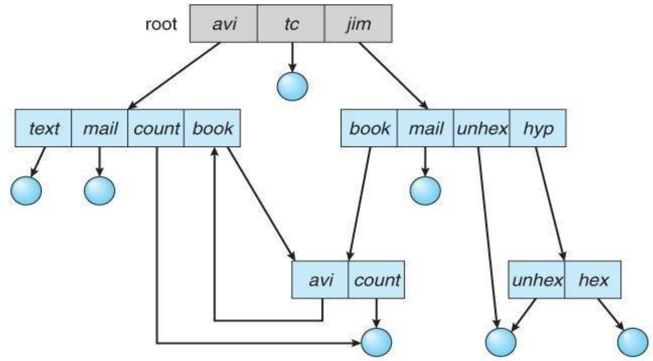
- When the **same files need to be accessed** in **more than one place** in the directory structure it can be useful to provide an acyclic-graph structure.
- In this system two or more directory entry can point to the same file orsub directory. That file or sub directory is shared between the two directory entries. It provides two types of ***links*** for implementing the acyclic-graph structure

Softlink,thefilejustgetsdeleted andweareleft withadangling pointer.

Hardlink,theactualfilewillbedeletedonlyifallthereferencestoit gets deleted.

GeneralGraph Directory

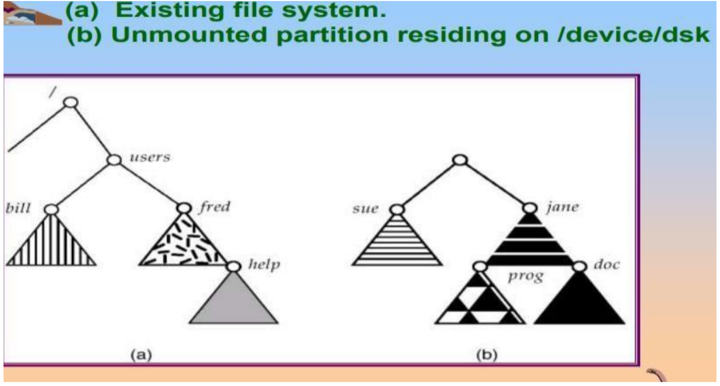
- Ingeneral graphdirectorystructure,cyclesareallowedwithinadirectory structure where multiple directories can be derived from more than one parent directory
- Themainproblemwiththiskindofdirectorystructureistocalculatetotal size or space that have been taken by the files and directories.

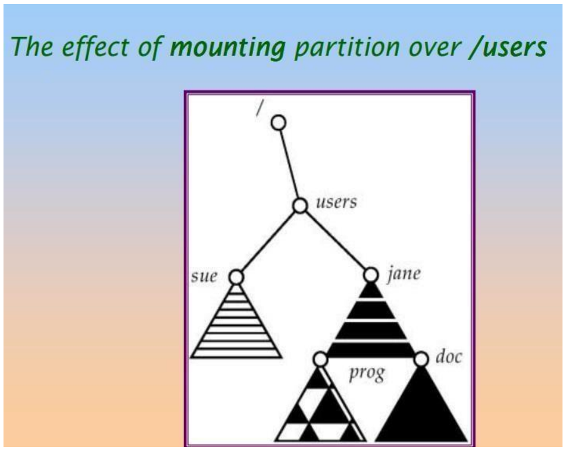


7. <u>FileSystem Mounting</u>

- Before you can access the files on a file system, you need to mount thefile system.

- Mountingafilesystemattachesthatfilesystemtoadirectory(mount point) and makes it available to the system.
- The root (/) file system is always mounted. Any other file system can be connected or disconnected from the root (/) file system.
- When you mount a file system, any files or directories in the underlying mount point directory are unavailable as long as the file system is mounted.
- Thesefilesare notpermanentlyaffectedbythemountingprocess,and they become available again when the file system is unmounted.
- However, mount directories are typically empty, because you usually do not want to obscure existing files.


(a) Existing file system.
(b) Unmounted partition residing on /device/dsk
/
users
bill
fred
help
sue
jane
prog
doc
(a)
(b)



8. FileSharing

- Filesharing is theaccessing or sharingoffiles byoneor moreusers.
- Filesharingisperformedoncomputernetworksasaneasyandquickwayto transmit data.

Forexample,ausermayshareaninstructiondocumentonhis computer that is connected to a corporate network allowing all other employees to access and read that document.

1. MultipleUsers

- Onamulti-usersystem,moreinformationneedstobestoredforeach file:The owner (user) who owns the file, and who can control its access.

- ❖ Thegroupofotheruser IDsthat mayhavesomespecial accesstothefile.
- ❖ Whataccessrights areaffordedto the owner(User), theGroup,andto the rest of the world.

2. **RemoteFileSystems**

The advent of the Internet introduces issues for accessing files stored on remotecomputers

- ❖ The original method was ftp, allowing individual files to be transported across systems as needed.
- ❖ TheClient-ServerModel(thesystemwhichphysicallyownsthefiles acts as a *server*, and the system which mounts them is the *client.)*
- ❖ Distributed Information Systems➔ service that runs on a single central location.
- ❖ Failure Modes ➔ When a local disk file is unavailable, the resultisgenerallyknownimmediately,andisgenerallynon-recoverable. The only reasonable response is for the response to fail. Remote access systems allow for blocking ordelayed response.

3. **ConsistencySemantics**

*ConsistencySemantics*dealswiththeconsistencybetweentheviewsofsharedfiles onanetworkedsystem.Whenoneuserchangesthefile,whendootheruserssee the changes?

1. UNIXSemantics

➔Writestoanopenfileareimmediatelyvisibletoanyotheruser who has the file open.

2. Session Semantics

AFSusesthefollowingsemantics:

➔Writesto anopen filearenotimmediately visibleto otherusers.

➔Whenafileisclosed,anychangesmadebecomeavailableonlyto users who open the file at a later time.

3. Immutable-Shared-FilesSemantics

➔whenafileisdeclaredas*shared*byitscreator,itbecomes immutable and the name cannot be re-used for any other resource.Henceit becomesread-only,andsharedaccessissimple.

9. <u>FileProtection</u>

- ❖ Files must be kept safe for reliability (against accidental damage), and protection(against deliberate malicious access.) The former is usually managed with backup copies. This section discusses the latter.
- ❖ Onesimpleprotectionschemeistoremoveallaccesstoafile.Howeverthismakesthefile unusable, so some sort of controlled access must be arranged.

TypesofAccess

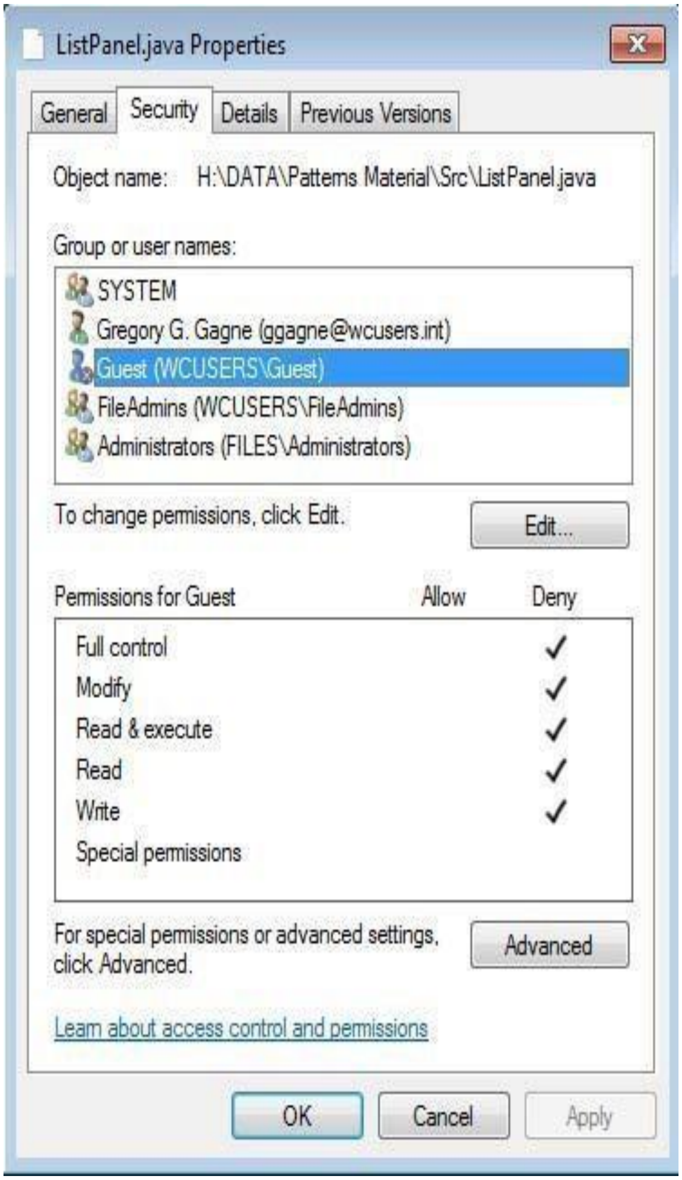
- Thefollowinglow-leveloperationsareoftencontrolled:

    - Read-View thecontentsofthe file
    - Write-Changethecontents of the file.
    - Execute-Loadthefileonto theCPUandfollowtheinstructionscontained therein.
    - Append-Addtotheendofanexistingfile.
    - Delete-Remove afilefrom thesystem.
    - List -Viewthenameandotherattributes offilesonthesystem.
- Higher-level operations, such as copy, can generally be performed through combinations of the above.

AccessControl

- One approach is to have complicated ***Access Control Lists, ACL,*** which specify exactly what access is allowed or denied for specific users or groups.
    - TheAFS usesthis systemfordistributed access.
    - Control is very finely adjustable, but may be complicated, particularly when the specificusersinvolvedareunknown.(AFSallowssomewildcards,soforexample all users on a certain remote system may be trusted, or a given username may be trusted when accessing from any remote system.)
- UNIX uses a set of 9 access control bits, in three groups of three. These correspond to R, W, and X permissions for each of the Owner, Group, and Others. (See "man chmod" for full details.) The RWX bits control the following privileges for ordinary files and directories:

bit	**Files**	**Directories**
R	**Read (view) file contents.**	**Read directory contents. Required to get a listing ofthe directory.**
W	**Write (change) file contents.**	**Change directory contents. Required to create or delete files.**
X	**Execute file contents as a program.**	**Accessdetaileddirectoryinformation.Requiredtoget a long listing, or to access any specific file in the directory. Note that if a user has X but not R permissionsonadirectory,theycanstillaccessspecific files,butonlyiftheyalreadyknowthenameofthefile they are trying to access.**



ListPanel.java Properties
General
Security
Details
Previous Versions
Edit...
Advanced
Learn about access control and permissions
OK
Cancel
Apply

Object name: H:\DATA\Patterns Material\Src\ListPanel.java

Group or user names:

- SYSTEM
- Gregory G. Gagne (ggagne@wcusers.int)
- Guest (WCUSERS\Guest)
- FileAdmins (WCUSERS\FileAdmins)
- Administrators (FILES\Administrators)

To change permissions, click Edit.

Permissions for Guest	Allow	Deny
Full control		✓
Modify		✓
Read & execute		✓
Read		✓
Write		✓
Special permissions		

For special permissions or advanced settings, click Advanced.

FILESYSTEMIMPLEMENTATION

1. FileSystem Structure

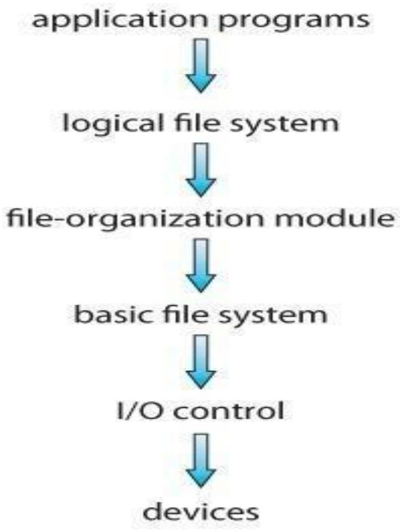
- FileSystemprovideefficientaccesstothediskbyallowingdatatobe stored, located and retrieved in a convenient way.
- A file System must be able to store the file, locate the file and retrieve the file.
- Most of the Operating Systems use layering approach for every task including file systems.
- Everylayerofthefilesystemisresponsibleforsomeactivities.

Logical filesystem

✄ **Provides** users the view of a contiguous sequence of words, bytes stored somewhere.

✄Usesadirectorystructure,symbolicname
✄Providesprotectionandsecurity
✄OS/userinterface

☎E.g., to create a new file the API provides a call that calls the logical filesystem



Thefileorganization module

✄Knowsaboutfilesandtheirlogicalblocks(say1,..N)
✄Filesareorganized inblocksof32bytes to4Kbytes
✄Translateslogical blocksinto physical
✄Knowslocation offile,fileallocationtype
✄Includesafreespace managesthat tracksunallocatedblocks

Basicfilesystem

✄Issuescommandstothedevicedriver(layerofsoftwarethatdirectlycontrolsdisk hardware) to read and write physical blocks on the disk,
✄Eachphysicalblockidentifiedbyadiskaddress(e.g.,drive2,cylinder34,track2, sector 11)

IO control

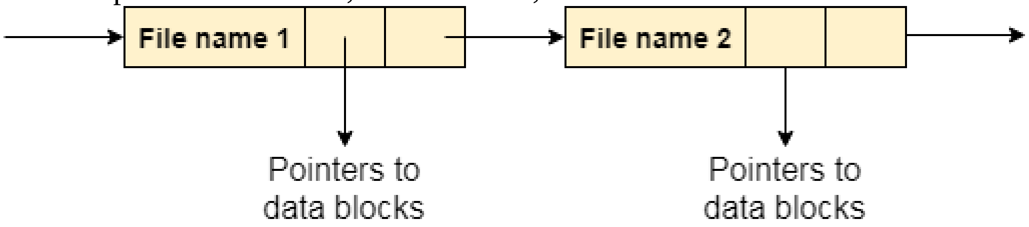
✄Thelowest levelin thefilesystem
✄Consistsofdevicedriversand interrupt handlers to transfer information betweenthe memory and the disk
✄Adevicedrivertranslatescommandssuchas“getmeblock111”intohardwarespecific ISA used by hardware controller. This is accomplished by writing specificbits into IO registers

2. DirectoryImplementation

- Directories need to be fast to search, insert, anddelete, with a minimum of wasted diskspace.

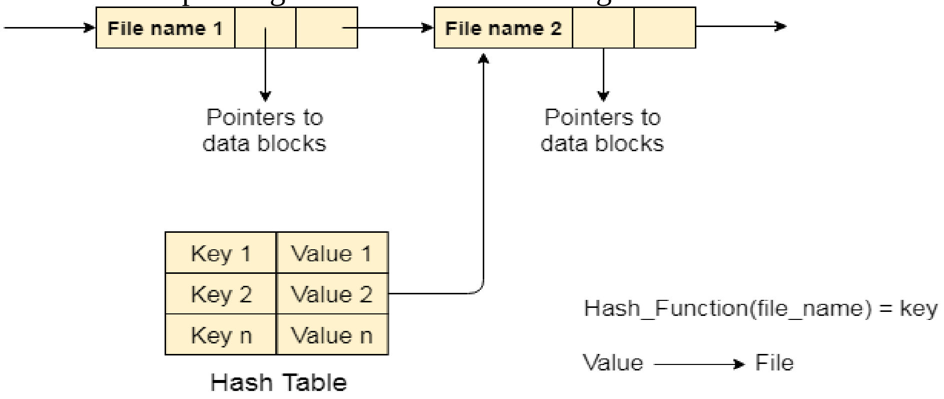
LinearList

- ❖ Alinearlist is thesimplest andeasiest directory structureto set up,but it does have some drawbacks.
- ❖ Findingafile(orverifyingonedoes notalreadyexist uponcreation)requires a linear search.
- ❖ Deletions can be done by moving all entries, flagging an entry as deleted, orby moving the last entry into the newly vacant position.
- ❖ Sorting the list makes searches faster, at the expense of more complex insertions and deletions.
- ❖ Alinkedlistmakesinsertionsanddeletionsintoasortedlisteasier,with overhead for the links.
- ❖ Morecomplexdatastructures,suchasB-trees,couldalsobe considered.



HashTable

- ❖ Ahashtablecan alsobe used tospeedup searches.
- ❖ Hashtablesaregenerallyimplementedinadditiontoalinearorother structure.
- ❖ A key-value pair for each file in the directory gets generated and stored in the hash table.
- ❖ Thekeycanbedeterminedbyapplyingthehashfunctiononthefilename while the key points to the corresponding file stored in the directory.
- ❖ **Searching**➔Onlyhashtableentriesarecheckedusingthekeyandifanentry found then the corresponding file will be fetched using the value.

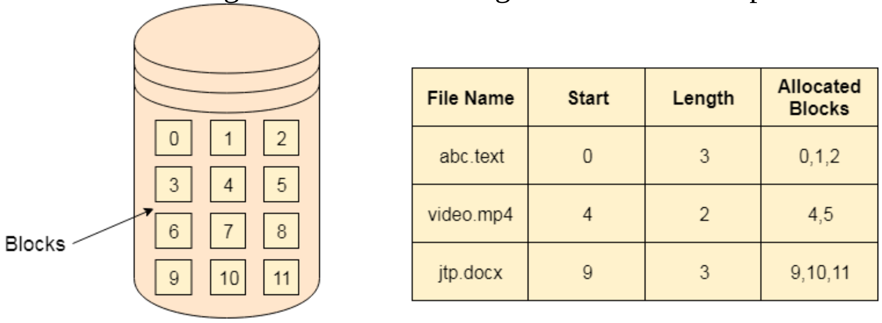


3. **AllocationMethods**

- Therearevarious methods which can be used to allocate disk space to the files. Selection of an appropriate allocation method will significantly affect the performance and efficiency of the system.
- Allocation method providesawayin whichthediskwillbeutilizedandthefileswillbeaccessed.

Contiguous Allocation

- If the blocks are allocated to the file in such a way that all the logical blocks of the filegetthecontiguousphysicalblockintheharddiskthensuchallocationschemeisknownas contiguous allocation.
- Intheimageshownbelow, therearethreefiles inthedirectory.
- The starting block and the length of each file are mentioned in the table.We can check in the table that the contiguous blocks are assigned to each file as per its need.



File Name	Start	Length	Allocated Blocks
abc.text	0	3	0,1,2
video.mp4	4	2	4,5
jtp.docx	9	3	9,10,11

Hard Disk

Directory

- Allthesealgorithms sufferfromtheproblemofexternalfragmentation.
- As files are allocated and deleted, the free disk space is broken into little pieces. External fragmentation exists whenever free space is broken into chunks.
- It becomes a problem when the largest contiguous chunk is insufficient for a request;storageisfragmentedintoanumberofholes,noneofwhichislarge enough to store the data.
- Thisschemeeffectively**compacts**allfreespaceintoonecontiguousspace,solving the fragmentation problem.

Advantages

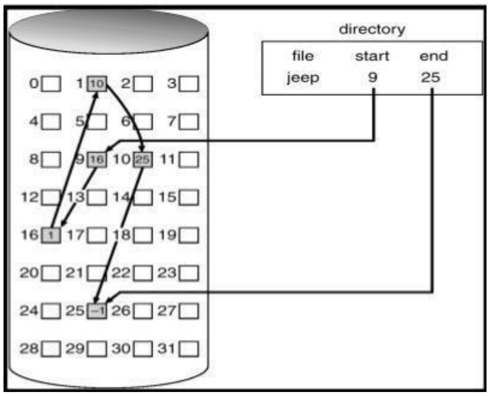
- Itissimpletoimplement.
- Wewill getExcellentreadperformance.
- SupportsRandomAccessintofiles.

Disadvantages

- Thedisk willbecome fragmented.
- Itmay bedifficult tohaveafilegrow.

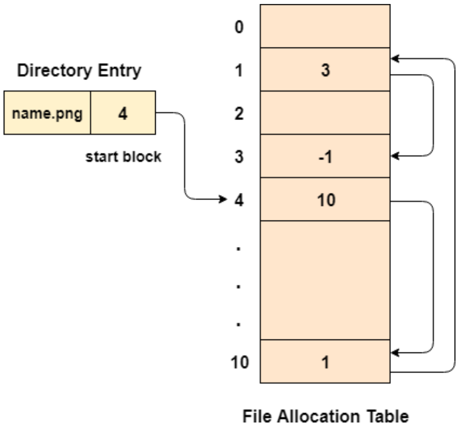
LinkedList Allocation

- LinkedListallocation solvesallproblemsofcontiguousallocation.
- Inlinkedlistallocation,eachfileisconsideredasthelinkedlistofdiskblocks.
- However, the disks blocks allocated to a particular file need not to be contiguous on the disk.
- Each disk block allocated to a file contains a pointer which points to the next disk block allocated to the same file.
- Forexample,afileoffiveblocksmightstartatblock9andcontinueatblock16,thenblock 1, then block 10, and finally block 25 (See Figure). Each block contains a pointerto the next block. These pointers are not made available to the user. Thus, if each block is512 bytesinsize,andadiskaddress(thepointer)requires4bytes,thentheusersees blocksof 508 bytes.



FileAllocation Table

- The main disadvantage of linked list allocation is that the Random access to a particular blockisnotprovided.Inordertoaccessablock,weneedtoaccessallitspreviousblocks.
- File Allocation Table overcomes this drawback of linked list allocation. In this scheme, a file allocation table is maintained, which gathers all the disk block links. The table has one entry for each disk block and is indexed by block number.
- Fileallocationtableneedstobecachedinorderto reducethenumberofheadseeks.Now theheaddoesn'tneedtotraversealltheddiskblocksinordertoaccessonesuccessiveblock.



Advantages

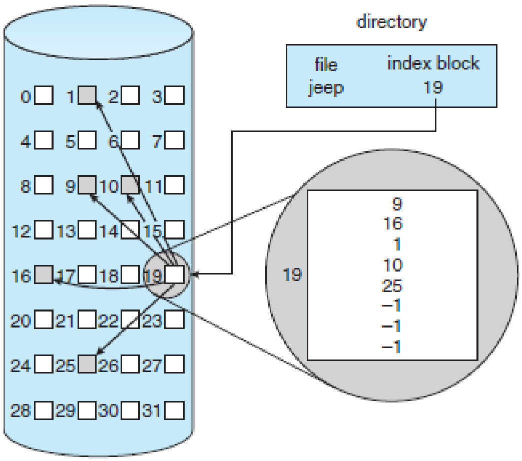
- Thereisnoexternal fragmentationwithlinkedallocation.
- Anyfreeblockcanbeutilized inorder tosatisfythefileblock requests.
- Filecan continueto growas long asthefreeblocks areavailable.
- Directory entrywill onlycontain thestartingblock address.

Disadvantages

- RandomAccessisnot provided.
- Pointersrequiresomespaceinthedisk blocks.
- Anyofthepointersinthelinkedlistmustnotbebrokenotherwisethefilewillget corrupted.
- Needtotraverseeach block.

IndexedAllocation

- Indexedallocationsolvesthisproblembybringingallthepointerstogetherintoone location: **the index block.**
- Eachfilehasitsown indexblock,whichis anarrayofdisk-block addresses.
- The ith entry in the index block points to the ith block of the file. The directory contains the address of the index block.
- Tofindandreadthe*ith*block,weusethepointerinthe*ith*index-blockentry.This scheme is similar to the paging scheme.
- Whenthefileiscreated,all pointersintheindexblockaresetto null.
- When the *ith* block is first written, a block is obtained from the free-space manager, and its address is put in the *i*th index-block entry.
- Indexed allocation supports direct access, without suffering from external fragmentation, because any free block on the disk can satisfy a request for more space.



Advantages

1. Supportsdirectaccess
2. Abaddatablockcausesthelost of onlythat block.

Disadvantages

1. Abadindexblock couldcausethe lostofentirefile.

2. Sizeof afiledepends uponthe numberofpointers,aindex block can hold.
3. Havingan indexblock forasmall fileis totallywastage.
4. Morepointeroverhead

4. FreeSpaceManagement

Since disk space is limited, we need to reuse the space from deleted files for new files, if possible. To keep track of free disk space, the system maintains a free-space list. The free-space list records all free disk blocks – those not allocated to some file or directory.

To create a file, we search the free-space list for the required amount of space, and allocate that space to the new file. This space is then removed from the free-space list. When a file is deleted, its disk space is added to the free-space list.

1. BitVector

The free-space list is implemented as a bit map or bit vector. Each block is represented by 1 bit.

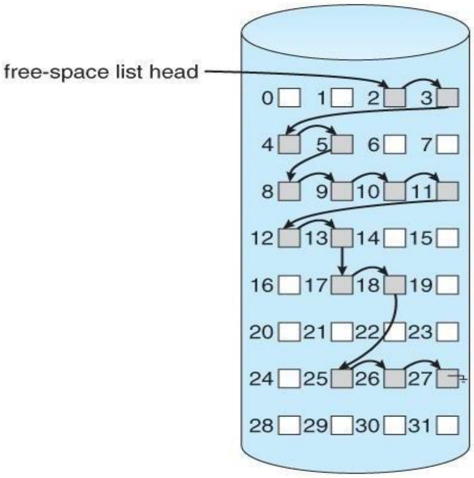
If the block is free, the bit is 1; if the block is allocated, the bit is 0. For example, Consideradiskwhereblock2,3,4,5,8,9,10,11,12,13,17,18,25,26and27arefree,and the rest of the block are allocated. The free space bit map would be 001111001111110001100000011100000 …

Themainadvantage **of**thisapproach isitsrelatively simplicityandefficiencyin finding the first free block, or n consecutive free blocks on the disk.

2. Linked List

Another approach to free-space management is to link together all the free disk blocks, keeping a pointer to the first free block in a special location on the disk and caching it in memory. This first blockcontains apointerto thenext freedisk block,and so on.

Inourexample,wewouldkeepapointertoblock2,asthefirstfreeblock.Block 2 would contain a pointer to block 3, which would point to block 4, which would point toblock5,whichwouldpointtoblock8,andsoon.However,thisschemeisnotefficient; to traverse the list, we must read each block, which requires substantial I/O time. The FAT method incorporates free-block accounting data structure. No separate method is needed.

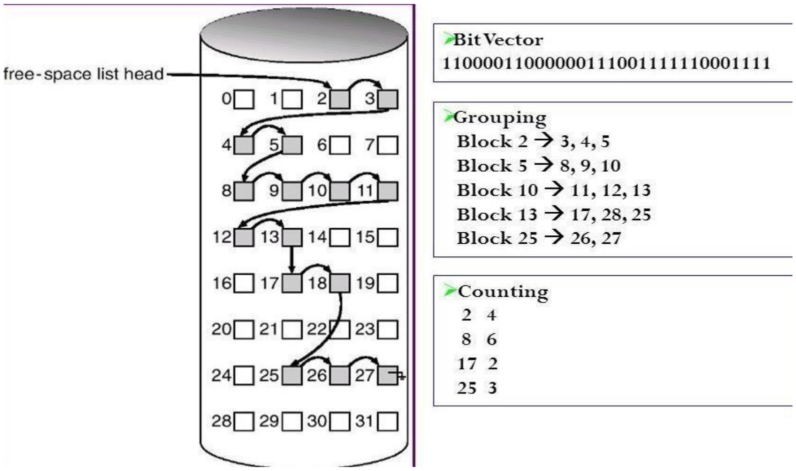


3. Grouping

Amodification ofthefree-list approach is to store theaddressesofn freeblocks in the first free block. The first n-1 of these blocks are actually free. The last block contains the addresses of another n free blocks, and so on. The importance of this implementation is that the addresses of a large number of free blocks can be found quickly.

4. Counting

We can keep the address of the first free block and the number n of free contiguousblocksthatfollowthefirstblock.Eachentryinthefree-spacelistthenconsists of a disk address and a count. Although each entry requires more space than would a simple disk address, the overall list will be shorter, as long as the count is generally greater than1.



➢Bit Vector
11000011000000111001111110001111

➢Grouping
Block 2 → 3, 4, 5
Block 5 → 8, 9, 10
Block 10 → 11, 12, 13
Block 13 → 17, 28, 25
Block 25 → 26, 27

➢Counting

2	4
8	6
17	2
25	3

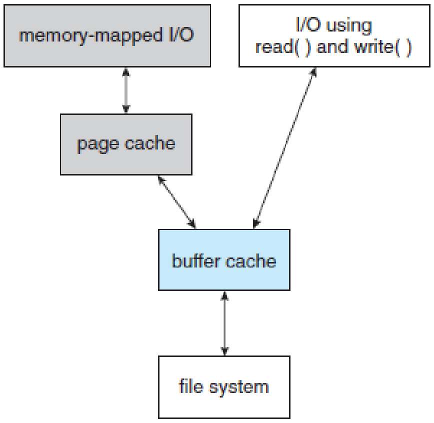
5. EfficiencyandPerformance

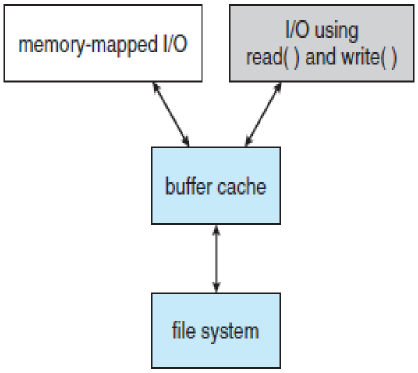
Efficiency

- The efficient use of disk space depends heavily on the disk-allocation and directory algorithms in use.
- Let’s reconsider the clustering scheme, which improves file-seek and file-transfer performance at the cost of internal fragmentation. To reduce this fragmentation, BSD UNIX varies the cluster size as a file grows. Large clusters are used where they can be filled, and small clusters are used for small files and the last cluster of a file. This
- The types of data normally kept in a file’s directory (or inode) entry also require consideration.Commonly,a“lastwritedate”isrecordedtosupplyinformationtotheuser and to determine whether the file needs to be backed up. Some systems also keep a “last access date,” so that a user can determine when the file was last read.
- The result of keeping this information is that, whenever the file is read, a field in the directory structure must be written to. That means the block must be read into memory, a sectionchanged,andtheblockwrittenbackouttodisk,becauseoperationsondisksoccur only in block (or cluster) chunks. So any time a file is opened for reading, its directory entry must be read and written as well.
- Generally, every data item associated with a file needs to be considered for its effect on efficiency and performance.

Performance

- Some systems maintain a separate section of main memory for a **buffer cache**, where blocks are kept under the assumption that they will be used again shortly. Other systems cache file data using a **page cache**.
- The **page cache** uses virtual memory techniques to cache file data as pages rather than as file-system-oriented blocks.
- Caching file data using virtual addresses is far more efficient than caching through physicaldiskblocks,asaccessesinterfacewithvirtualmemoryratherthanthefilesystem.
- Several systems—including Solaris, Linux, and Windows —use page caching to cache both process pages and file data. This is known as **unified virtual memory**.





- ❖ Thetwoalternativesforopeningandaccessingafile.Oneapproach is to use memory mapping the second is to use the standard system calls **read()** and **write()**.
- ❖ Here, the read() and write() system calls go through the buffer cache.
- ❖ The memory-mapping call, however, requires using two caches—the **page cache** and the **buffer cache.**
- ❖ A memory mapping proceeds by reading in disk blocks from the filesystemandstoringtheminthebuffercache.Becausethevirtualmemorysystemdoes not interface with the buffer cache, the contents of the file in the buffer cache must be copied into the page cache. This situation, known as **double caching**,requirescaching file-system data twice.

6. Recovery

ConsistencyChecking

- ❖ Thestoring ofcertain datastructures (e.g. directories and inodes)in memoryand thecachingofdiskoperationscanspeedupperformance,butwhathappensintheresult of a system crash? All volatile memory structures are lost, and the information stored on the hard drive may be left in an inconsistent state.
- ❖ A Consistency Checker (fsck in UNIX, chkdsk or scandisk in Windows) is often run at boot time or mount time, particularly if a filesystem was not closed down properly. Some of the problems that these tools look for include:
  - ➢ Diskblocksallocatedtofilesandalsolistedonthefreelist.
  - ➢ Diskblocksneitherallocatedtofilesnoronthefreelist.
  - ➢ Diskblocksallocatedto morethan onefile.
  - ➢ Thenumberof disk blocks allocated to afileinconsistent with the file's stated size.
  - ➢ Properly allocated files / inodes which do not appear in any directory entry.
  - ➢ Linkcountsforaninodenotmatchingthenumberofreferencestothat inode in the directory structure.
  - ➢ Twoormoreidentical filenames inthesame directory.
  - ➢ Illegally linked directories, e.g. cyclical relationships where those arenotallowed,orfiles/directoriesthatarenotaccessiblefromtherootofthe directory tree.
  - ➢ Consistency checkers will often collect questionable disk blocks into new files with names such as chk00001.dat. These files may contain valuable information that would otherwise be lost, but in most cases theycan be safely deleted, (returning those disk blocks to the free list.)

UNIX caches directory information for reads, but any changes that affect space allocationormetadata changesarewrittensynchronously,beforeanyofthecorresponding data blocks are written to.

Log-StructuredFileSystems

- ➢ Log-based transaction-oriented (a.k.a. journaling) filesystems borrowtechniquesdevelopedfordatabases,guaranteeingthatanygiventransaction either completes successfully or canbe rolled back to a safe state before the transaction commenced:

- ❖ Allmetadatachangesarewrittensequentiallytoalog.
- ❖ Aset ofchangesforperforming aspecifictask (e.g. moving afile)is atransaction.
- ❖ As changes are written to the log they are said to be committed, allowing the system to return to its work.
- ❖ In the meantime, the changes from the log are carried out on the actual filesystem, and a pointer keeps track of which changes in the log have been completed and which have not yet been completed.
- ❖ When all changes corresponding to a particular transaction have been completed, that transaction can be safely removed from the log.
- ❖ At any given time, the log will contain information pertaining to uncompleted transactions only, e.g. actions that were committed but for which the entire transaction has not yet been completed.
- ❖ Fromthelog,theremainingtransactionscanbecompleted,orifthe transaction was aborted, then the partially completed changes can be undone.

BackupandRestore

- ❖ Inordertorecoverlostdataintheeventofadiskcrash,itis important to conduct backups regularly.
- ❖ Files should be copied to some removable medium, such as magnetic tapes, CDs, DVDs, or external removable hard drives.
- ❖ A full backup copies every file on a file system.Incremental backupscopyonlyfileswhichhave changedsincesomeprevious time.
- ❖ A combination of full and incremental backups can offer a compromisebetweenfullrecoverability,thenumberandsizeofbackuptapes needed, and the number of tapes that need to be used to do a full restore.
- ❖ Atypicalbackup schedulemaythen beas follows:

Day1.Copytoabackupmediumallfilesfromthedisk.Thisiscalleda full backup.

Day2.Copytoanothermediumallfileschangedsinceday1.Thisisan incremental backup.

Day3.Copytoanothermediumall fileschangedsinceday 2.

DayN.CopytoanothermediumallfileschangedsincedayN−1.Then go back to day 1.

I/O SYSTEMS

1. I/OHardware

The role of the operating system in computer I/O is to manage and control I/O operationsand I/O devices.Adevicecommunicateswithacomputersystembysending signals over a cable or even through the air.

Port: The device communicates with the machine via a connection point (or port), for example, a serial port.

Bus:Ifoneormoredevicesuseacommonsetofwires,the connection is called a bus.

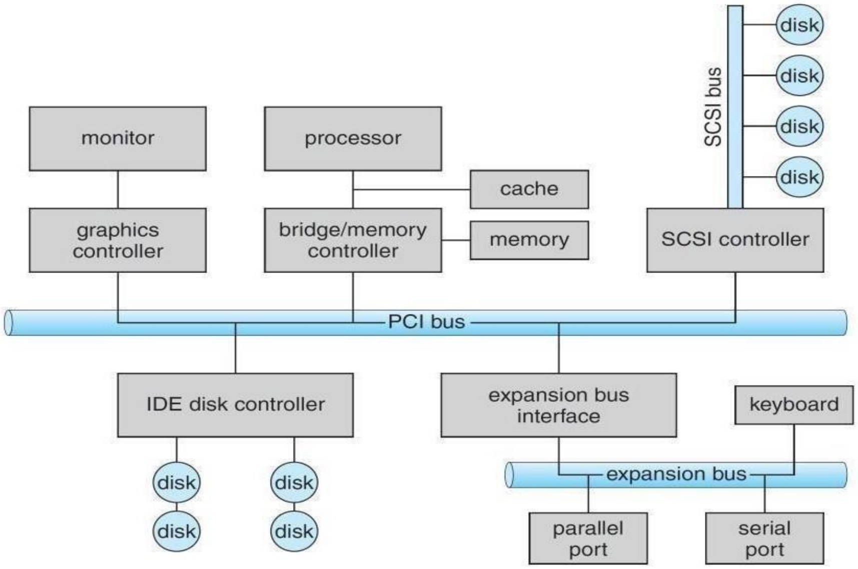
Daisychain:Device_A‘hasacablethatplugsintodevice_B‘,anddevice _B ‘has a cable that plugs into device _C ‘, and device _C ‘plugs into a port onthecomputer,thisarrangementiscalledadaisychain.Adaisychainusually operates as a bus.

PCbus structure

APCIbusthatconnectstheprocessor-memorysubsystemtothefastdevices,and an expansion bus that connects relatively slow devices such as the keyboard and serial and parallel ports. In the upper- right portion of the figure, four disks are connected together on a SCSI bus plugged into a SCSI controller.

A**controllerorhostadapter**isacollectionofelectronicsthatcanoperateaport, abus,oradevice.Aserial-portcontrollerisasimpledevicecontroller.Itisasinglechip inthecomputerthatcontrolsthesignalsonthewiresofaserialport.By contrast,aSCSI bus controller is not simple.

BecausetheSCSIprotocoliscomplex,theSCSIbuscontrollerisoftenimplemented as a separate circuit board. It typically contains a processor, microcode, and some private memory. Some devices have their own built- in controllers.



Howcantheprocessorgivecommandsanddatatoacontrollertoaccomplish anI/Otransfer?

- DirectI/Oinstructions
- Memory-mappedI/O

DirectI/Oinstructions

Use special I/O instructions that specify the transfer of a byte or word to an I/O port address.TheI/Oinstructiontriggersbuslinestoselecttheproperdeviceandtomovebitsintoor out of a device register

Memory-mappedI/O

Thedevice-controlregistersaremappedintotheaddressspaceoftheprocessor.TheCPU executes I/O requests using the standard data-transfer instructions to read and write the device-control registers.

Statusregister	Readbythehosttoindicatestatessuchas whetherthecurrentcommand hascompleted,whethera byteisavailabletobereadfrom thedata-in register,andwhethertherehasbeenadevice error.
Controlregister	Writtenby thehost tostart acommand orto changethemodeof adevice.
data-inregister	Readby thehost toget input
data-outregister	Writtenby thehost to send output

- An I/O port typically consists of four registers: status, control, data-in, and data-out registers.

1. Polling
Interactionbetweenthe hostandacontroller

- Thecontrollersetsthebusybitwhenitisbusyworking,andclearsthebusybit when it is ready to accept the next command.
- The host sets the command ready bit when a command is availablefor the controller to execute.

Coordinationbetweenthehost&thecontrolleris donebyhandshakingasfollows:

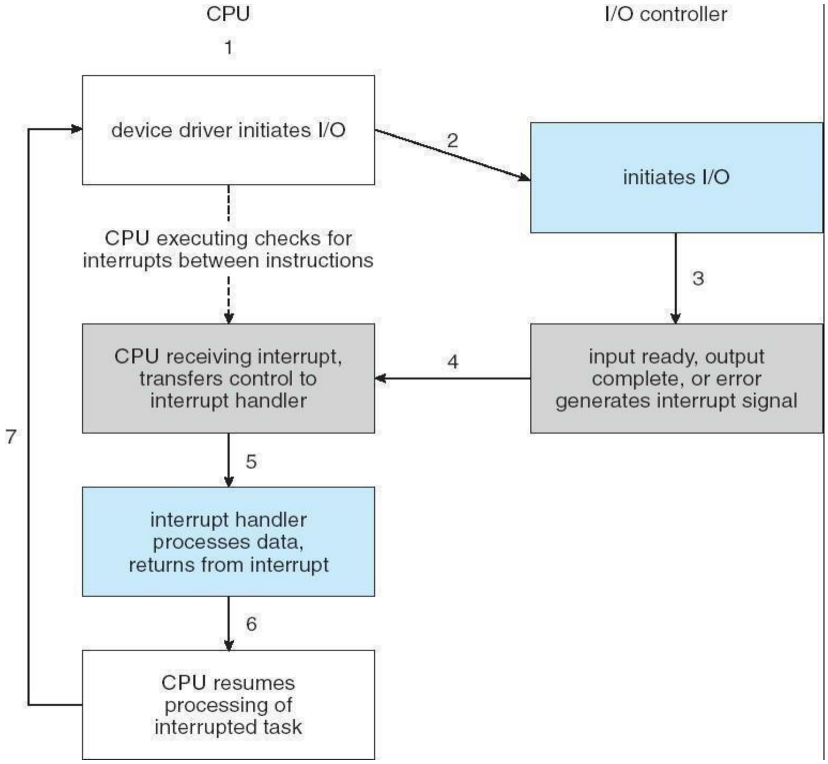
1. Thehostrepeatedly readsthebusy bituntilthat bitbecomesclear.
2. The host sets the write bit in the command register and writes a byte into the data-out register.
3. Thehostsetsthecommand-readybit.
4. Whenthecontrollernoticesthatthecommand-readybit isset,itsetsthebusy bit.
5. The controller reads the command register and sees the write command. Itreadsthedata-outregistertogetthebyte,anddoestheI/Otothedevice.
6. The controller clears the command-ready bit, clears the error bit in the status register to indicate that the device I/O succeeded, and clears the busy bit to indicate that it is finished.
7. In step 1, the host is **―busy-waiting or polling‖**: It is in a loop, reading the status register over and over until the busy bit becomes clear.

2. Interrupts

TheCPUhardwarehasawirecalledthe―interrupt-requestline‖. The basic interrupt mechanism works as follows;

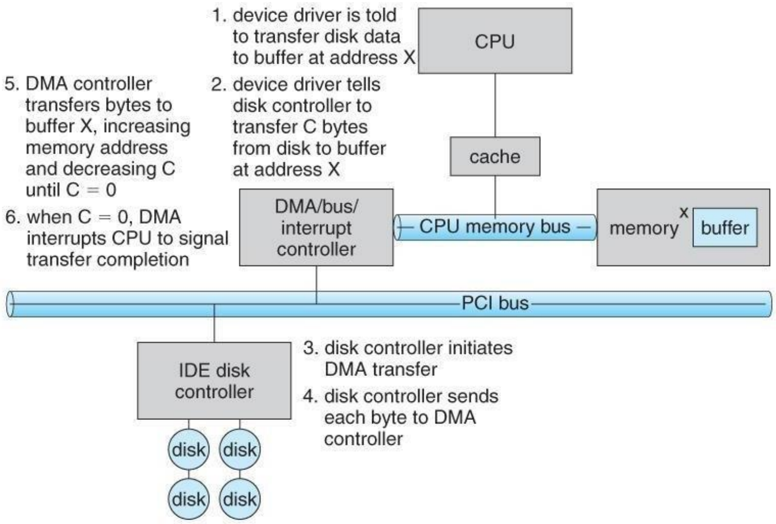
1. Device controller raises an interrupt by asserting a signal on the interrupt request line.
2. TheCPUcatchestheinterruptanddispatchestotheinterrupthandlerand
3. Thehandlerclearstheinterruptbyservicingthedevice.

- **Nonmaskable interrupt**: which is reserved for events such as unrecoverable memory errors?
- **Maskableinterrupt**:Usedbydevicecontrollerstorequestservice


CPU
I/O controller
1
device driver initiates I/O
2
initiates I/O
CPU executing checks for interrupts between instructions
3
CPU receiving interrupt, transfers control to interrupt handler
4
input ready, output complete, or error generates interrupt signal
7
5
interrupt handler processes data, returns from interrupt
6
CPU resumes processing of interrupted task

3. DirectMemoryAccess (DMA)

In general it is tough for the CPU to do the large transfers between the memory buffer&disk;becauseitisalreadyequippedwithsomeothertasks,thenthiswill create overhead. So a special-purpose processor called a direct memory-access **(DMA)** controller is used.


1. device driver is told to transfer disk data to buffer at address X
CPU
5. DMA controller transfers bytes to buffer X, increasing memory address and decreasing C until C = 0
2. device driver tells disk controller to transfer C bytes from disk to buffer at address X
cache
6. when C = 0, DMA interrupts CPU to signal transfer completion
DMA/bus/ interrupt controller
CPU memory bus
memory
X
buffer
PCI bus
IDE disk controller
3. disk controller initiates DMA transfer
4. disk controller sends each byte to DMA controller
disk
disk
disk
disk

2. ApplicationI/OInterface

I/O system calls encapsulate device behaviours in generic classes. Device-driver layer hides differences among I/O controllers from kernel

Devicesvaryonmanydimensions,asillustrated in

• **Character-streamorblock**.Acharacter-streamdevicetransfersbytesonebyone, whereas a block device transfers a block of bytes as a unit.

• **Sequentialorrandomaccess**.Asequentialdevicetransfersdatainafixed order determined by the device, whereas the user of a random-access device can instruct the device to seek to any of the available data storage locations.

• **Synchronousorasynchronous**.Asynchronousdeviceperformsdatatransferswith predictable response times, in coordination with other aspects of the system. Anasynchronous device exhibits irregular or unpredictable response times not coordinatedwithothercomputerevents.

• **Sharableordedicated**.Asharabledevicecanbeusedconcurrentlybyseveral processes or threads; a dedicated device cannot.

Speedofoperation.Devicespeedsrangefromafewbytespersecondtoafew gigabytes per second.

• **Read–write,readonly,orwriteonly**.Somedevicesperformbothinputand output, but others support only one data transfer direction.

1. BlockandCharacterDevices

Block-device: The block-device interface captures all the aspects necessary for accessing disk drives and other block-oriented devices. The device should understand the commands such as read () & write (), and if it is a random access device, it has a seek() command to specify which block to transfer next.

CharacterDevices: Akeyboardisanexampleofadevicethatisaccessedthrough a character stream interface. The basic system calls in this interface enable an application to get() or put() one character.

2. NetworkDevices

Becausetheperformanceandaddressingcharacteristics ofnetworkI/Odiffer

significantlyfromthoseofdiskI/O,mostoperatingsystemsprovideanetworkI/O interface that is different from the read0
-write()-seek()interfaceusedfordisks.

- WindowsNTprovidesoneinterfacetothenetworkinterfacecard, and a second interface to the network protocols.

- InUNIX,wefindhalf-duplexpipes,full-duplexFIFOs,full- duplex STREAMS, message queues and sockets.

3. Clocks and Timers

Mostcomputershavehardwareclocksandtimersthatprovidethreebasicfunctions:

- Givethecurrenttime
- Givetheelapsedtime
- SetatimertotriggeroperationXattimeT

Programmable interval timer: The hardware to measure elapsed time and to trigger operationsiscalledaprogrammableintervaltimer.Itcanbesettowaitacertainamountof time and then to generate an interrupt. To generate periodic interrupts, it can be set todo this operation once or to repeat.

UsesofProgrammableinterval timer:

Scheduler	Togenerateaninterrupt thatwillpre-emptaprocessat theendofits timeslice.
DiskI/Osubsystem	Toinvoketheflushingof dirtycachebufferstodiskperiodically
Network subsystem	Tocanceloperationsthoseareproceedingtooslowlybecauseof networkcongestion orfailures.

Whenthetimerinterrupts,thekernelsignalstherequester,andreloadsthe timer with the next earliest time.

Counter:Thehardware clock isconstructedfromahighfrequencycounter.

In some computers,the valueofthis counter canbereadfrom adevice register,in which the counter can be considered to be a high-resolution clock.

4. Blocking and Non-blocking I/O (or) synchronous & asynchronous: Blocking I/O: When an application issues a blocking system call;

- Theexecution ofthe application issuspended.
- Theapplication ismovedfromtheoperatingsystem'srunqueuetoawait queue.
- After the system call completes, the application is moved back to the run queue,whereitis eligibletoresume execution,at whichtime itwillreceive the values returned by the system call.

Non-blocking,I/O: Someuser-level processesneednon-blocking

I/O***Examples:***
Userinterfacethatreceiveskeyboardandmouseinputwhileprocessing and displaying data on the screen.
Videoapplication that reads frames from afile on disk while simultaneously decompressing and displaying the output on the display.

3. KernelI/O Subsystem

Kernelsprovidemanyservicesrelatedto I/O.

- One way that the I/O subsystem improves the efficiency of the computer is by scheduling I/O operations.
- Another way is by using storage space in main memory or on disk, via techniques called buffering, caching, and spooling.

1. ***I/O Scheduling:***

Todetermineagoodorderinwhich toexecutethesetof I/Orequests.Uses:

- Itcanimproveoverallsystemperformance,
- Itcansharedeviceaccessfairlyamongprocesses, and
- Itcanreducetheaverageawaitingtimefor1/0to complete.

Implementation:OSdevelopersimplementschedulingbymaintaininga —queueofrequests foreachdevice.

- WhenanapplicationissuesablockingI/Osystemcall,
- Therequest is placedon thequeueforthat device.
- The I/O scheduler rearranges the order of the queue to improve the overall system efficiency and the average response time experienced by applications.

2. ***Buffering:***

Buffer: A memory area that stores data while they are transferred between two devices or between a device and an application.

Reasonsforbuffering:

- To cope with a speed mismatch between the producer and consumer of a datastream.
- Toadaptbetweendevicesthathavedifferentdata-transfer sizes.
- TosupportcopysemanticsforapplicationI/O.

Copy semantics Suppose that an application has a buffer of data that itwishes to write to disk. It calls the write () system call, providing a pointer to the buffer and an integer specifying the number of bytes to write.

3. ***Caching***

Acacheisaregionoffastmemorythatholdscopiesofdata.Accesstothecached copy is more efficient than access to the original

Cache vs buffer: A buffer may hold the onlyexisting copy ofa data item,whereas a cache just holds a copy on faster storage of an item that resides elsewhere.

WhenthekernelreceivesafileI/Orequest,

1. Thekernelfirstaccessesthebuffercachetoseewhetherthatregionofthefile is already available in main memory.

2. If so, a physical disk I/O can be avoided or deferred. Also, disk writes are accumulated in the buffer cache for several seconds, so that large transfers are gathered to allow efficient write schedules.

4. SpoolingandDeviceReservation:

Spool:Abufferthatholdsoutputforadevice,suchasaprinter,thatcannotaccept interleaved data streams.

Aprintercanserveonlyonejobatatime,severalapplicationsmaywishtoprint their output concurrently, without having their output mixed together

TheOSprovidesacontrolinterfacethatenablesusersandsystemadministrators;

- Todisplaythe queue,
- Toremoveunwantedjobsbeforethosejobs print,
- Tosuspendprintingwhiletheprinterisserviced,andsoon.

Devicereservation-providesexclusiveaccess toa device

- Systemcallsforallocationandde-allocation
- Watchoutfordeadlock

5. Error Handling

An operating system that uses protected memory can guard against many kinds of hardware and application errors. OS can recover from disk read, device unavailable, transientwritefailuresMostreturnanerrornumberorcodewhenI/OrequestfailsSystem error logs hold problem reports

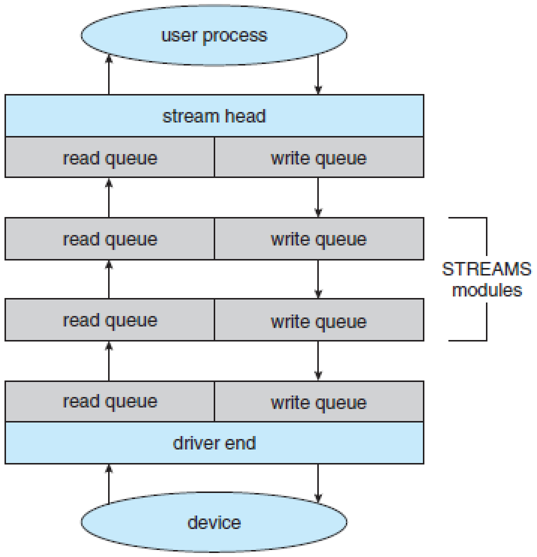
STREAMS

Streamisafull-duplexcommunicationchannelbetweenauser-levelprocess andadevicein Unix SystemVand beyondASTREAMconsists of:

- STREAMheadinterfaces withtheuserprocess
- Driverendinterfaces withthe device
- ZeroormoreSTREAMmodulesbetweenthem.

Each module contains a read queue and a write queue. Message passing is used to communicate between queues. Modules provide the functionality of STREAMS processing and they are pushed onto a stream using the ioct () system call.

Flow control: Because messages are exchanged between queues in adjacent modules, a queue in one module may overflow an adjacent queue. To prevent this from occurring, a queue may support flow control.



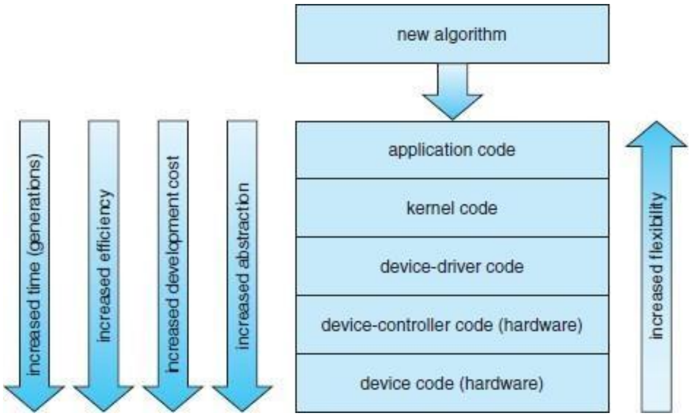
PERFORMANCE

I/Oamajorfactorin systemperformance:

- Heavy demands on CPU to execute device driver, kernel I/O code. So context switches occur due to interrupts.
- Interrupt handling is a relatively expensive task: Each interrupt causes the system to perform a state change, to execute the interrupt handler & then to restore state
- Networktrafficespeciallystressful.
- Systems use separate ―front-end processors" for terminal I/O, to reduce the interrupt burden on the main CPU.

Wecanemployseveral principlestoimprovetheefficiencyofI/O:

- Reducethenumber ofcontext switches.
- Reducethenumberoftimesthatdatamustbecopiedinmemorywhile passing between device and application.
- Reduce the frequency of interrupts by using large transfers, smart controllers &polling.
- Increase concurrency by using DMA-knowledgeable controllers or channels to offload simple data copying from the CPU.
- Move processing primitives into hardware, to allow their operation in devicecontrollers concurrent with the CPU and bus operation.
- BalanceCPU,memorysubsystem,bus,andI/Operformance,becausean overload in any one areawill cause idleness in others.



Device functionality progression.

a) **An application-level implementation**: Implement experimental I/O algorithms at the application level, because application code is flexible, and application bugs are unlikely to cause system crashes.

Itcan beinefficient;

- Becauseoftheoverheadofcontextswitches and
- Because the application cannot take advantage of internal kernel data structures and kernel functionality

b) **In-kernelimplementation**:Re-implementapplication-levelalgorithminthe kernel. This can improve the performance, but the development effort is more challenging, because an operating-system kernel is a large, complex software system. Moreover, an in-kernel implementation must be thoroughly debugged to avoid data corruption and system crashes.

c) **Ahardwareimplementation**:Thehighestperformancemaybeobtainedby a specialized implementation in hardware, either in the device or in the controller.

- ❖ Difficult and expense of making further improvements or of fixing bugs, (-) Increased development time
- ❖ Decreasedflexibility.

UNITV CASE STUDY

Linux System - Design Principles, Kernel Modules, Process Management, Scheduling, Memory Management,Input-OutputManagement,FileSystem,Inter-processCommunication;MobileOS - iOS and Android - Architecture and SDK Framework, Media Layer, Services Layer, Core OS Layer, File System.

1. LINUXSYSTEM

LinuxHistory

- Itsdevelopmentbeganin1991,whenaFinnishuniversitystudent,LinusTorvalds, began developing a small but self-contained kernel for the 80386 processor, the first true 32-bit processor in Intel's range of PC-compatible CPUs.
- Early in its development, the Linux source code was made available free— bothat no cost and with minimal distributional restrictions—on the Internet.
- The **Linux kernel** is an original piece of software developed from scratch by the Linux community.
- The**Linuxsystem**,includesamultitudeofcomponents,somewrittenfromscratch, others borrowed from other development projects, and still others created in collaboration with other teams.
- A **Linux distribution** includes all the standard components of the Linux system, plus a set of administrative tools to simplify the initial installation and subsequent upgrading of Linux and to manage installation and removal of other packages on the system.

TheLinux Kernel

- The first Linux kernel released to the public was version 0.01, dated May 14, 1991. It had no networking, ran only on 80386-compatible Intel processors and PC hardware, and had extremely limited device-driver support.
- Thenextmilestone,**Linux1.0**,wasreleasedonMarch14,1994.
- This release culminated three years of rapid development of the Linux kernel.Perhapsthesinglebiggestnewfeaturewasnetworking:1.0included support for UNIX's standard TCP/IP networking protocols such as socket interface for networking programming.
- In **March 1995**, the **1.2 kernel** was released. This release did not offer nearly the same improvement in functionality as the 1.0 release, but it did supportamuchwidervarietyofhardware,includingthenewPCIhardware bus architecture.
- In **June 1996** as Linux **version 2.0**was released. This release was given a major version-number increment because of two major new capabilities: supportformultiplearchitectures,includinga64-bitnativeAlphaport,and symmetric multiprocessing (SMP) support
- Improvements continued with the release of **Linux 2.2** in **1999**. A port to UltraSPARCsystemswasadded.Networkingwasenhancedwithmore

flexiblefirewalling,improvedroutingandtrafficmanagement,andsupport for TCP large window and selective acknowledgement.

- Linuxkernelversion3.0wasreleasedinJuly2011.

2.DESIGN PRINCIPLES

- Linuxrunson awidevarietyofplatforms, itwas originallydeveloped exclusively on PC architecture.
- Linux can run happily on a multiprocessor machine with many gigabytes of main memory and many terabytes of disk space, but it is still capable of operating usefully in under 16 MB of RAM.

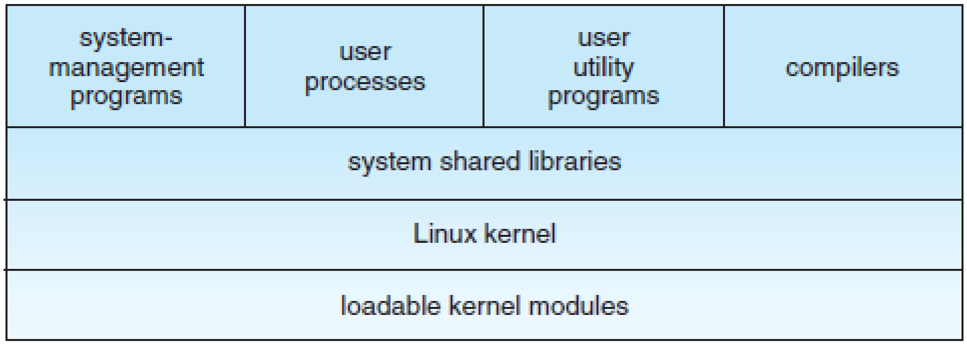
➔ComponentsofaLinux System

TheLinuxsystemis composedofthreemainbodiesof code

1. **Kernel**. Thekernel is responsible formaintaining all the important abstractions of the operating system, including such things as virtual memory and processes.

2. **Systemlibraries**.Thesystemlibrariesdefineastandardsetoffunctionsthrough which applications can interact with the kernel. These functions implement much oftheoperating-systemfunctionalitythatdoesnotneedthefullprivilegesofkernel code.ThemostimportantsystemlibraryistheClibrary,knownaslibc.Inaddition toprovidingthestandardClibrary,libcimplementstheusermodesideoftheLinux system call interface, as well as other critical system- level interfaces.

3. **System utilities**. The system utilities are programs that perform individual, specialized management tasks. Some system utilities are invoked just once to initializeandconfiguresomeaspectofthesystem.Others—knownasdaemonsin UNIX terminology—run permanently, handling such tasks as responding to incoming network connections, accepting logon requests from terminals, and updating log files.



- All the kernel code executes in the processor’s privileged mode with full access to all the physical resources of the computer.
- Linuxreferstothis**privilegedmode**askernelmode.
- UnderLinux, no usercodeis built into thekernel.
- Anyoperating-system-supportcodethatdoesnotneedtoruunin**kernel mode** is placed into the system libraries and runs in user **mode**.
- Unlike **kernel mode**, **user mode** has access only to a controlled subset of the system’s resources.

3.KERNELMODULES

- TheLinuxkernelhastheabilitytoloadandunloadarbitrarysectionsofkernelcode on demand.
- Theseloadablekernelmodulesruninprivilegedkernelmodeandasaconsequence havefull access to all the hardwarecapabilities of the machine on which they run.
- Kernelmodules areconvenientforseveral reasons.
  1. **Linux'ssourcecodeisfree**,soanybodywantingtowritekernelcodeisable to compile a modified kernel and to reboot into that new functionality.
  2. However,**recompiling,relinking,andreloading**theentirekernelisa cumbersome cycle to undertake when you aredeveloping a new driver.
  3. If you use kernel modules, you **do not have to make a new kernel** to test a newdriver—thedrivercanbecompiledonitsownandloadedintothealready running kernel.
- KernelmodulesallowaLinuxsystemto besetupwithastandardminimalkernel, without any extra device drivers built in.
- Anydevicedriversthattheuserneedscanbeeitherloadedexplicitlybythesystem atstartuporloadedautomaticallybythesystemondemandandunloadedwhennot in use.
- Forexample,amousedrivercanbeloadedwhenaUSBmouseispluggedintothe system and unloaded when the mouse is unplugged.

1. **The module-management** system allows modules to be loaded into memory and to communicate with the rest of the kernel.
2. **Themoduleloaderandunloader**,whichareuser-modeutilities,workwiththe module-management system to load a module into memory.
3. **Thedriver-registration**systemallowsmodulestotelltherestofthekernelthat a new driver has become available.
4. **A conflict-resolution mechanism** allows different device drivers to reserve hardware resources and to protect those resources from accidental use by another driver.

Module Management

- Loading a module requires more than just loading its binary contents into kernel memory.
- Linuxmaintains aninternalsymbol tablein thekernel.
- Theloading ofthe moduleis performedin**twostages**.

➔First,**themoduleloaderutility** asksthekerneltoreservea continuous areaofvirtualkernelmemoryforthemodule.Thekernelreturnstheaddress of the memory allocated, and the loader utility can use this addressto relocate the module's machine code to the correct loading address.

➔A**secondsystemcallthenpasses**themodule,plusanysymboltablethat the new module wants to export, to the kernel.

DriverRegistration

- provides a set of routines to allow drivers to be added to or removed.

- ❖ Amodulemay registermanytypesoffunctionality
- ❖ For example, a device driver might want to register two separate mechanisms for accessing the device. Registration tables include, among others, the following items:

  Device drivers. These drivers include character devices (such as printers, terminals, and mice), block devices (including all disk drives), and network interface devices.

  File systems. The file system may be anything that implements Linux's virtual file system calling routines. It might implement a format for storing files on a disk, but it might equally well be a network file system, such as NFS, or a virtual file system whose contentsaregeneratedondemand,suchasLinux's/procfilesystem.

  Network protocols. A module may implement an entire networking protocol, such as TCP or simply a new set of packet-filtering rules for a network firewall.

  Binary format. This format specifies a way of recognizing, loading, and executing a new type of executable file.

Conflict Resolution

Linuxprovidesacentralconflict-resolutionmechanismtohelparbitrateaccessto certain hardware resources. Its aims are as follows:

➔Toprevent modulesfromclashingoveraccesstohardwareresources

➔ To prevent autoprobes—device-driver probes that auto-detect device configuration—from interfering with existing device drivers

➔ To resolve conflicts among multiple drivers trying to access the same hardware—as, for example, when both the parallel printer driver and the parallel line IP (PLIP) network driver try to talk to the parallel port

4.PROCESSMANAGEMENT

Aprocess is the basic context in whichall user-requested activity is serviced within the operating system.

➔Thefork()andexec()ProcessModel

- ❖ The basic principle of UNIX process management is to separate into two stepstwooperationsthatareusuallycombinedintoone:The**creationofanewprocess** and the **running of a new program**.
- ❖ A new process is created by the fork()system call, and anew program is runafter a call to exec().
- ❖ Thesearetwodistinctlyseparatefunctions.
- ❖ Wecancreateanewprocesswithfork()withoutrunninganewprogram—thenew subprocess simply continues to execute exactly the same program, at exactly the same point, that the first (parent) process was running.

1. ProcessIdentity

➔Aprocessidentityconsistsmainlyofthefollowing items:

ProcessID(PID).Eachprocesshasaunique identifier.

Credentials. EachprocessmusthaveanassociateduserIDand one or more group IDs that determine the rights of a process to access system resources and files.

Personality: Personalitiesareprimarilyusedbyemulationlibraries to request the system calls be compatible with certain varieties of UNIX.

Namespace: Each process is associated with aspecificview of the filesystem hierarchy, calleditsnamespace.Most processes sharea common namespace and thus operate on a shared file-system hierarchy.

➔ProcessEnvironment

Aprocess'senvironmentisinheritedfromitsparentandiscomposedof **twonull-terminatedvectors**:**theargumentvector**andthe**environmentvector**.

Theargumentvectorsimplyliststhecommand-lineargumentsusedtoinvoketherunning program; it conventionally starts with the name of the program itself.

The environment vector is a list of "NAME=VALUE" pairs that associates named environmentvariableswitharbitrarytextualvalues.Theenvironmentisnotheldinkernel memory but is stored in the process's own user-mode address space as the first datum at the top of the process's stack.

➔ProcessContext

- Processcontextisthestateoftherunningprogramatanyonetime;itchanges constantly. Process context includes the following parts:

Scheduling context: The most important part of the processcontext is its scheduling context—the information that the scheduler needs to suspend and restart the process. This information includes saved copies of all the process's registers.

Accounting: The kernel maintains accounting information about the resourcescurrentlybeingconsumedbyeachprocessandthetotalresources consumed by the process in its entire lifetime so far.

File table. The file table is an array of pointers to kernel file structures representing open files.

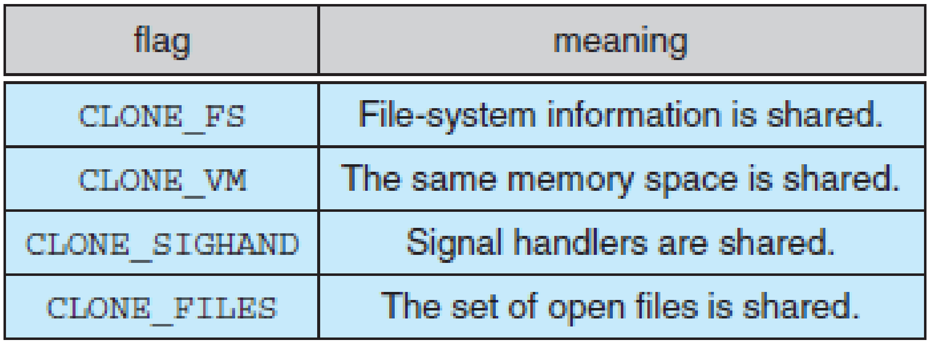
File-systemcontext: Whereasthefiletableliststheexistingopenfiles,the file-system context applies to requests to open new files. The file- system contextincludestheprocess'srootdirectory,currentworkingdirectory,and namespace.

Signal-handlertable: Thesignal-handlertabledefinestheactiontotakein response to a specific signal.

Virtual memory context : The virtual memory context describes the full contents of a process's private address space.

2. ProcessesandThreads

- Linux provides the fork() system call, which duplicates a process without loading a new executable image. Linux also provides the ability to create threads via the clone() system call.
- The clone() system call behaves identically to fork(), except that it accepts as arguments a set of flags that dictate what resources are shared between the parent and child.
- Theflagsinclude



flag	meaning
CLONE_FS	File-system information is shared.
CLONE_VM	The same memory space is shared.
CLONE_SIGHAND	Signal handlers are shared.
CLONE_FILES	The set of open files is shared.

5. SCHEDULING

- SchedulingisthejobofallocatingCPUtimetodifferenttaskswithinanoperating system.
- Linux,likeallUNIXsystems,supportspreemptive multitasking.
- Insuchasystem,theprocessschedulerdecideswhichprocessrunsandwhen.

➔ProcessScheduling

Linuxhastwoseparateprocess-schedulingalgorithms.

1. Oneis atime-sharingalgorithm forfair,preemptiveschedulingamong multiple processes.
2. The other is designed for real-time tasks, where absolute priorities are more important than fairness.

CompletelyFairScheduler(CFS).

- InCFSeachcoreoftheCPUhas itsownrunqueue.
- Each task has a so called nice value and weight assigned to it. The nice value represents how “kind” the specific task is to other tasks.
- Inotherwords,ataskwithahighnicevaluehasalowerpriorityandisthus less likely to take more of the CPUs bandwidth than a task with a lownice value.
- CFS introduced a new scheduling algorithm called **fair scheduling** that eliminates **time slices** in the traditional sense. **Instead of time slices**, all processes are **allotted a proportion of the processor’s time**. CFS calculates how long a process should run as a function of the total number of runnable processes.
- To calculate the actual length of time a process runs, CFS relies on a configurable variable called **target latency**, which is the interval of time during which every runnable task should run at least once.

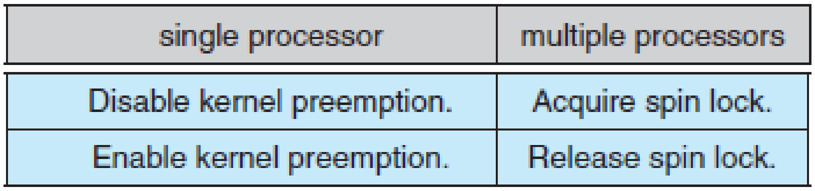
➔Real-TimeScheduling

- Linux implements the two real-time scheduling classes: first-come, first served (FCFS) and round-robin.
- Inboth cases,eachprocess hasapriorityinadditiontoitsschedulingclass.
- The scheduler always runs the process with the highest priority. Among processes of equal priority, it runs the process that has been waiting longest.
- Theonly differencebetween FCFS and round-robin scheduling isthat FCFS processes continue to run until they either exit or block, whereas a round-robin process will be preempted after a while and will be moved to the end of the scheduling queue, so round-robin processes of equal priority will automatically time-share among themselves.
- Linux’sreal-timeschedulingis**soft**—**andhard**—**real time**.

  ➔ A hard real time system guarantees that critical tasks complete on time, whereas in soft real time system, a critical real time task gets priority over other tasks and retains that priority until it completes.

➔KernelSynchronization

- Thewaythekernelschedulesitsownoperationsisfundamentallydifferent from the way it schedules processes.
- Arequestforkernel-modeexecutioncanoccurintwoways.
- A running program may request an operating-system service, either explicitly via a system call or implicitly—for example, when a page fault occurs.
- Alternatively, a device controller may deliver a hardware interrupt that causes the CPU to start executing a kernel-defined handler for that interrupt.
- The problem for the kernel is that all these tasks may try to access the same internal data structures.
- If one kernel task is in the middle of accessing some data structure when an interrupt service routine executes, then that service routine cannot access or modify the same data without risking data corruption.
- The Linux kernel provides spinlocks and semaphores (as well as reader–writer versions of these two locks) for locking in the kernel.
- Linuxusesaninterestingapproachtodisableandenablekernelpreemption. Itprovidestwosimplekernelinterfaces—**preemptdisable()**and**preempt enable().**



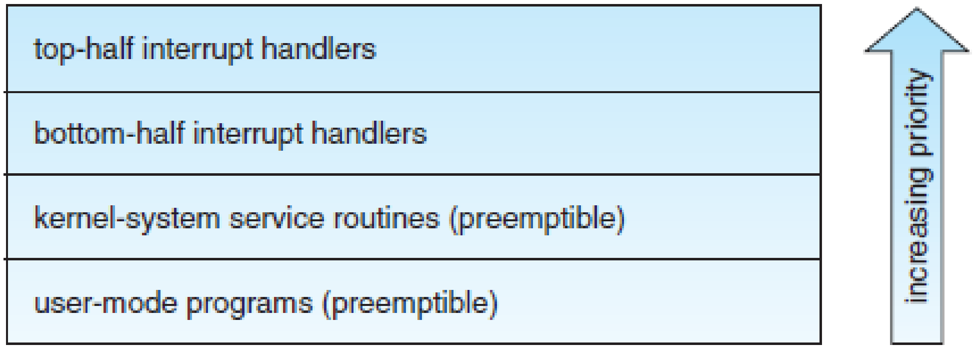
single processor	multiple processors
Disable kernel preemption.	Acquire spin lock.
Enable kernel preemption.	Release spin lock.

- Thecounteris incremented when alock is acquired and decremented when a lock is released.
- Linux implements this architecture by separating interrupt service routines into two sections: **the top half** and the **bottom half**.

  The **top half** is the standard interrupt service routine that runs with recursive interrupts disabled.

Interrupts of the same number (or line) are disabled, but other interrupts may run.
The **bottom half** of a service routine is run, with all interrupts enabled,by a miniature scheduler that ensures that bottom halves never interrupt themselves.

Interrupt protection levels.


top-half interrupt handlers
bottom-half interrupt handlers
kernel-system service routines (preemptible)
user-mode programs (preemptible)
increasing priority

➔Symmetric Multiprocessing

Linux kernel to support symmetric multiprocessor (SMP) hardware, allowing separate processes to execute in parallel on separate processors. The original implementation of SMP imposed the restriction that only one processor at a time could be executing kernel code.

6. MEMORYMANAGEMENT

MemorymanagementunderLinuxhastwocomponents.Thefirstdealswithallocatingandfreeing physical memory—pages, groupsof pages,and small blocksofRAM. Thesecondhandlesvirtual memory, which is memory-mapped into the address space of running processes.

➔ManagementofPhysicalMemory

❖ Due to specific hardware constraints, Linux separates physical memory into four different zones, or regions:

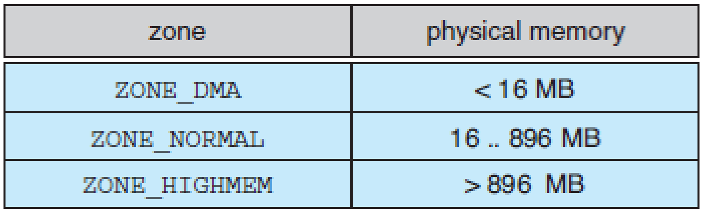
- ZONE DMA
- ZONEDMA32
- ZONENORMAL
- ZONEHIGHMEM

❖ ZONE_DMA. This zone contains pages that can undergo DMA.

❖ ZONE_DMA32. Like ZOME_DMA, this zone contains pagesthatcanundergoDMA.UnlikeZONE_DMA,thesepagesareaccessibleonlyby32- bit devices. On some architectures, this zone is a larger subset of memory.

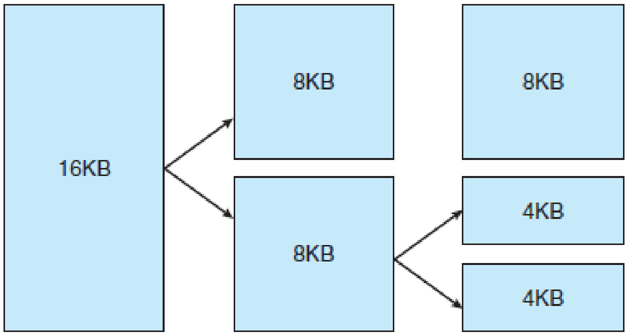
❖ ZONE_NORMAL. This zone contains normal, regularly mapped, pages.

- ZONE_HIGHMEM. This zone contains "high memory", whicharepagesnotpermanentlymappedintothekernel’saddressspace.Therelationship of zones and physical addresses on the Intel x86-32 architecture is shown Below

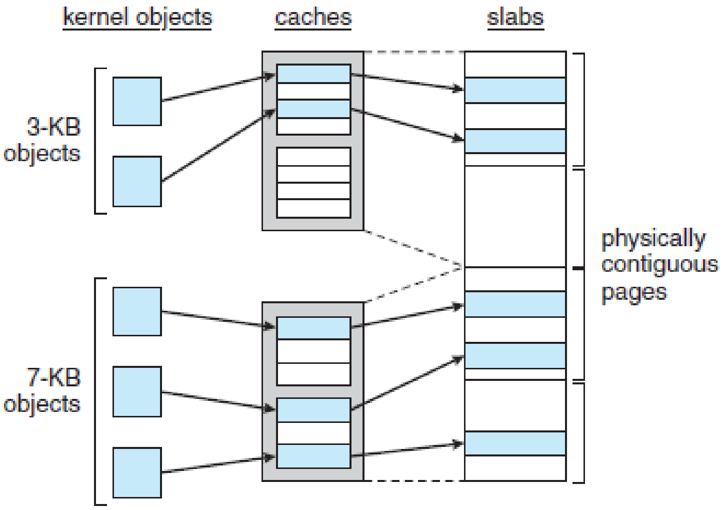


zone	physical memory
ZONE_DMA	< 16 MB
ZONE_NORMAL	16 .. 896 MB
ZONE_HIGHMEM	> 896 MB

- ThepriaryphysicaI-memorymanagerintheLinuxkernel is the **page allocator**.
- Each zone has its own allocator, which is responsible for allocating and freeing all physical pages for the zone and is capable of allocating ranges of physically contiguous pages on request.
- The allocator uses a **buddy system** to keep track of available physical pages.
- Eachallocatablememoryregionhasanadjacentpartner(or buddy).Whenevertwoallocatedpartnerregionsarefreedup,theyarecombinedtoforma larger region—a buddy heap.



- Another strategy adopted by Linux for allocating kernel memory is known as slab allocation. A slab is used for allocating memory for kerneldata structures and is made up of one or more physically contiguous pages. A cache consists of one or more slabs.



InLinux,aslabmay bein oneofthreepossible states:

1. Full.All objectsintheslabaremarkedasused.
2. Empty.Allobjectsintheslabaremarkedas free.
3. Partial.Theslabconsistsofbothusedandfree objects.

➔VirtualMemory

- TheLinuxvirtualmemorysystemisresponsibleformaintainingtheaddressspace accessible to each process.
- Itcreatespagesofvirtualmemoryondemandandmanagesloadingthosepages from disk and swapping them back out to disk as required.
- UnderLinux,thevirtualmemorymanagermaintainstwoseparateviewsofa process’s address space: as a set of **separate regions** and as a **set of pages**.

1. VirtualMemoryRegions

- Linuximplementsseveraltypesofvirtualmemoryregions.
- Onepropertythatcharacterizesvirtualmemoryisthebackingstorefortheregion, which describes where the pages for the region come from.
- Mostmemoryregionsare backedeitherby a file orbynothing.
- Aregionbacked bynothing isthesimplest typeofvirtualmemory region.
- Sucharegionrepresents demand-zeromemory:whenaprocesstriestoreadapagein such a region, it is simply given back a page of memory filled with zeros.
- Avirtualmemoryregionisalsodefinedbyitsreactiontowrites.Themappingofa region into the process’s address space can be either private or shared.

2. Lifetimeofa Virtual Address Space

- Thekernelcreates anewvirtual addressspaceintwo situations:
- whenaprocessruns anewprogramwiththeexec()systemcallandwhenanew process is created by the fork() system call.

3. Swappingand Paging

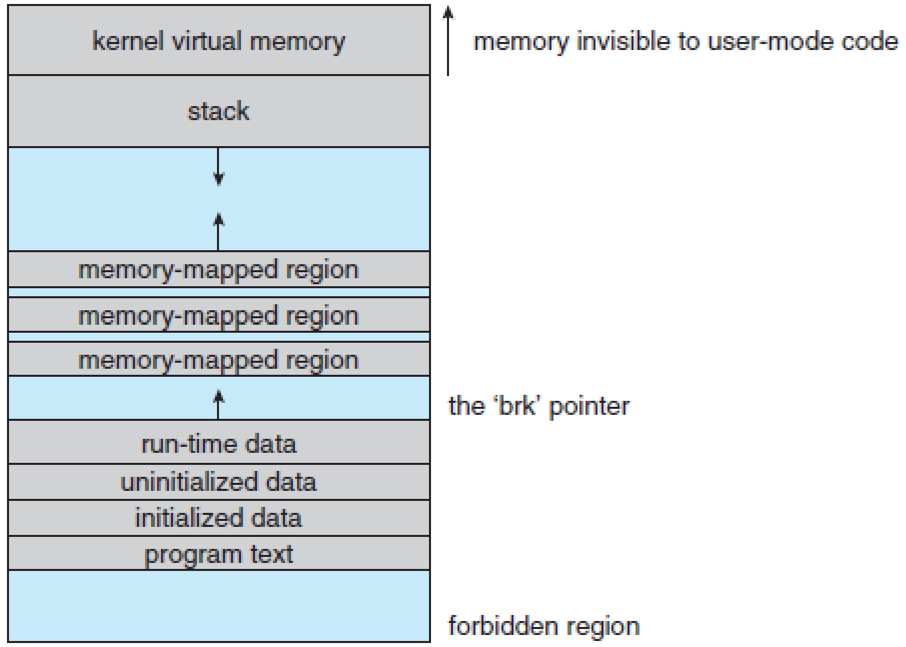
- Animportanttaskforavirtualmemorysystemistorelocatepagesofmemoryfromphysical memory out to disk when that memory is needed.
- Thepagingsystemcanbedividedintotwosections.
- First,thepolicyalgorithmdecideswhichpagestowriteouttodiskandwhentowritethem.
- Second,thepagingmechanismcarriesoutthetransferandpagesdatabackintophysical memory when they are needed again.

4. KernelVirtualMemory

- kernelvirtualmemoryareacontains tworegions.
- Thefirstisastaticareathatcontainspage-tablereferencestoeveryavailablephysical page of memory in the system, so that a simple translation from physical to virtual addresses occurs when kernel code is run.
- Theremainderofthekernel'sreservedsectionofaddressspaceisnotreservedforany specific purpose.

➔ExecutionandLoadingofUserPrograms

- TheLinuxkernel'sexecutionofuserprogramsistriggeredbyacalltotheexec()system call.
- This exec()call commands thekernel to runanewprogram within thecurrent process, completelyoverwritingthecurrentexecutioncontextwiththeinitialcontextofthenew program.
- The first job of this system service is to verify that the calling process has permission rights to the file being executed.
- NewerLinuxsystemsusethemoremodernELFformat,nowsupportedbymostcurrent UNIX implementations.



Memory layout for ELF programs.

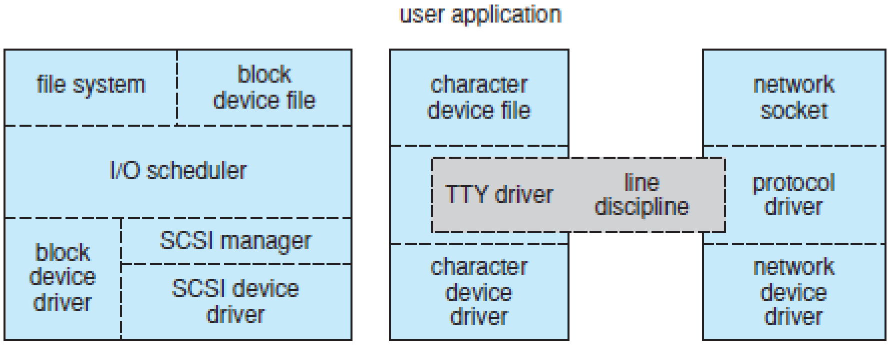
7. INPUTANDOUTPUTMANAGEMENT

- Linuxsplitsalldevicesintothreeclasses:blockdevices,characterdevices, and network devices.

➔Block Devices

- Blockdevicesprovidethemaininterfacetoall diskdevicesina system.
- Performance is particularly important for disks, and the block-device system must provide functionality to ensure that disk access is as fast as possible.
- ThisfunctionalityisachievedthroughtheschedulingofI/Ooperations.

- In the context of block devices, a block represents the unit with which the kernel performs I/O.
- Whenablockis readintomemory,itisstoredinabuffer.
- The request manager is the layer of software that manages the reading and writing of buffer contents to and from a block-device driver.
- Aseparatelist ofrequestsiskeptforeachblock-devicedriver.
- Theserequestshavebeenscheduledaccordingtoa unidirectional-elevator (C-SCAN) algorithm that exploits the order in which requests are inserted in and removed from the lists.
- Whenarequestisacceptedforprocessingbyablock-devicedriver,itisnot removed from the list.
- It is removed only after the I/O is complete, at which point the driver continues with the next request in the list, even if new requests have been inserted in the list before the active request.



Device-driver block structure.

➔CharacterDevices

- Acharacter-devicedrivercanbealmostanydevicedriverthatdoesnotoffer random access to fixed blocks of data.
- Any character-device drivers registered to the Linux kernel must also register a set of functions that implement the file I/O operations that the driver can handle.
- Thekernelperformsalmostnopreprocessingofafilereadorwriterequest to acharacterdevice. It simply passes the request to the devicein question and lets the device deal with the request.
- A line discipline is an interpreter for the information from the terminal device.
- The most common line discipline is the **tty discipline**, which glues the terminal'sdatastreamontothestandardinputandoutputstreamsofauser's runningprocesses, allowingthoseprocessestocommunicatedirectlywith the user's terminal

➔Networkdevices

- Networkdevices aredealt withdifferentlyfromblockandcharacterdevices.
- Users cannot directly transfer data to network devices. Instead, they must communicate indirectly by opening a connection to the kernel's networking subsystem.

8. INTERPROCESSCOMMUNICATION

Linuxprovidesarichenvironmentfor processestocommunicatewitheach other.

➔SynchronizationandSignals

- ❖ The standard Linux mechanism for informing a process that an event has occurred is the signal.
- ❖ Signals can be sent from any process to any other process, with restrictions on signals sent to processes owned by another user.
- ❖ The kernel also generates signals internally. For example, it can send a signalto a server process when data arrive on a network channel, to a parent process when a child terminates, or to a waiting process when a timer expires.
- ❖ Internally,theLinuxkerneldoesnotusesignalstocommunicatewithprocesses runninginkernelmode.Ifakernel-modeprocessisexpectinganeventtooccur, it will not use signals to receive notification of that event.
- ❖ Rather, communication about incoming asynchronous events within the kernel takes place through the use of scheduling states and **wait queue structures**
- ❖ Whenever a process wants to wait for some event to complete, it places itselfon a wait queue associated with that event and tells the scheduler that it is no longer eligible for execution.
- ❖ Oncetheeventhascompleted,everyprocessonthewaitqueuewillbeawoken.

➔PassingofDataamong Processes

- ❖ Linuxoffersseveralmechanismsforpassingdataamongprocesses.
- ❖ The standard UNIX pipe mechanism allows a child process to inherit a communication channel from its parent; data written to one end ofthe pipe can be read at the other.
- ❖ Under Linux, pipes appear as just another type of inode to virtual file system software, and each pipe has a pair of wait queues to synchronize the reader and writer.
- ❖ Anotherprocesscommunicationsmethod,sharedmemory,offersanextremelyfast way to communicate large or small amounts of data.
- ❖ Any data written by one process to a shared memory region can be read immediatelybyanyotherprocessthathasmappedthatregionintoitsaddressspace.

9. FILESYSTEMS

- ❖ TheLinuxkernelhandlesalltypesoffilesbyhidingtheimplementationdetailsofanysingle file type behind a layer of software, the virtual filesystem (VFS).

➔TheVirtual File System

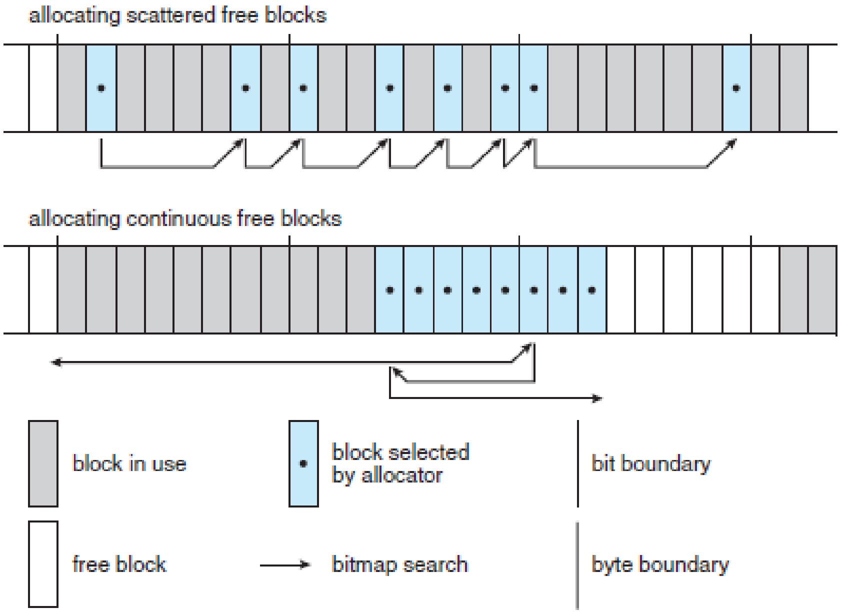
- ❖ The Linux VFS is designed around object-oriented principles. It has two components:asetofdefinitionsthatspecifywhatfile-systemobjectsareallowed to look like and a layer of software to manipulate the objects.
- ❖ TheVFSdefinesfourmainobject types:
  - ➔Aninodeobjectrepresents anindividualfile.
  - ➔Afileobjectrepresents anopenfile.
  - ➔Asuperblockobjectrepresents anentirefilesystem.
  - ➔Adentryobjectrepresentsanindividualdirectory entry.
- ❖ Foreachofthesefourobjecttypes,theVFS definesasetofoperations.

- Everyobjectof oneofthesetypescontains apointerto afunction table.
- Thefunctiontableliststheaddressesoftheactualfunctionsthatimplementthe defined operations for that object.
- Forexample,anabbreviatedAPIforsomeofthefileobject'soperations includes:

  int open(...)—Open afile.
  ssize t read(. . .) — Read from a file.
  ssize t write(. . .) — Write to a
  file.intmmap(...)—Memory-mapafile.

➔TheLinux ext3FileSystem

- Thestandardon-diskfilesystemusedbyLinuxiscalledext3,for historical reasons.
- LinuxwasoriginallyprogrammedwithaMinix-compatiblefilesystem,to ease exchanging data with the Minix development system, but that file system was severely restricted by 14-character file-name limits and a maximum file-system size of 64 MB.
- The ext3 allocation policy works as follows: As in FFS, an ext3 file systemispartitionedintomultiplesegments.Inext3,thesearecalled block groups.
- FFSusesthesimilarconceptofcylindergroups,whereeachgroup corresponds to a single cylinder of a physical disk.
- Whenallocatingafile, ext3mustfirst selecttheblockgroupforthat file.
- Fordatablocks,itattemptstoallocatethefiletotheblockgrouptowhich the file's inode has been allocated. For inode allocations, it selects the lock group in which the file's parent directory resides for nondirectory files.


allocating scattered free blocks
allocating continuous free blocks
block in use
block selected by allocator
bit boundary
free block
bitmap search
byte boundary

ext3 block-allocation policies.

➔**Journaling**

- The ext3 file system supports a popular feature called journaling, whereby modifications to the file system are written sequentially to a journal.
- Asetofoperations thatperforms aspecifictask isatransaction.
- Once a transaction is written to the journal, it is considered to be committed. Meanwhile, the journal entries relating to the transaction are replayed across the actual file system structures.
- As the changes are made, a pointer is updated to indicate which actions have completed and which are still incomplete. When an entirecommitted transaction is completed, it is removed from the journal.
- Ifthesystemcrashes, sometransactions mayremaininthejournal.
- Thosetransactionswerenevercompletedtothefilesystemeventhoughtheywere committed by the operating system, so they must be completed once the system recovers.
- The transactions can be executed from the pointer until the work is complete, and the file-system structures remain consistent.

➔TheLinuxProcessFile System

- TheLinuxprocessfilesystem,knownasthe/procfilesystem,isanexample of a file system whose contents are not actually stored anywhere but are computed on demand according to user file I/O requests.
- The/procfilesystem containsaillusionaryfilesystem.
- It does not exist on a disk. Instead, the kernel creates it in memory. It is used to provide information about the system (originally about processes, hence the name).
- Someof themoreimportantfilesanddirectoriesare explainedbelow.The /procfilesystem isdescribed inmoredetailin theprocmanual page.
- The/procfilesystemmustimplement**twothings**:a**directorystructure** andthe**filecontentswithin**.
- Toallowefficientaccesstothesevariablesfromwithinapplications,the /proc/sys subtree is made available through a special system call, sysctl(), that reads and writes the same variables in binary, rather than in text, without the overhead of the file system. sysctl() is not an extra facility; it simplyreadsthe/procdynamicentrytreetoidentifythevariablestowhich the application is referring.

MOBILEOPERATINGSYSTEMS

- A mobile operating system (OS) is software that allows smartphones, tablet PCs andother devices to run applications and programs.
- A mobile OS typically starts up when a device powers on, presenting a screen with icons ortilesthatpresentinformationandprovideapplicationaccess.Mobileoperatingsystems also manage cellular and wireless network connectivity, as well as phone access.
- Examples of mobile device operating systems include **Apple iOS**, **Google Android**, Research in Motion’s **BlackBerry OS**, Nokia’s Symbian, Hewlett-Packard’s webOS (formerly Palm OS) and Microsoft’s **Windows Phone OS**. Some, such as Microsoft’s Windows 8, function as both a traditional desktop OS and a mobile operating system.

- Most mobile operating systems are tied to specific hardware, with little flexibility. Users can jailbreak or root some devices, however, which allows them to install another mobile OS or unlock restricted applications.

10. ANDRIODVSIOS

IOS

It is Apple's mobile operating system used to run the popular iPhone, iPad, and iPod Touch devices. Formerly known as the iPhone OS, the name was changed with the introduction of the iPad. It interprets the commands of software applications ("apps") and it gives those apps access to features of the device, such as the multi-touch screen or the storage.

FeaturesofIOS

- System Fonts
- Folders
- Notificationcenter
- Accessibility
- Multitasking
- Switching Applications(application does not execute any code and may be removed from memory at any time)
- TaskCompletion (helpstoaskextratimeforcompletion oftask)
- Backgroundaudio(helpstoruninbackground)
- Voiceover IP(in casephonecallisnot in progress)
- Backgroundlocation(notifiedwhenlocationchanges)
- Push notifications

ANDROID

AndroidisasoftwarepackageandLinuxbasedoperatingsystemformobiledevicessuchastablet computersandsmartphones.ItisdevelopedbyGooglein2007andlatertheOHA(OpenHandset Alliance).BecauseGoogledevelopedAndroid,itcomeswithalotofGoogleappservicesinstalled rightoutofthebox.Gmail,GoogleCalendar,GoogleMaps,andGoogleNowareallpre-installed on most Android phones

AndroidOShasmanyfeatures,amongwhicharethefollowing:

- Enhancedinterfacewiththearrayoficonsonthemenu.Androidadaptstohighquality 2D and 3D graphics, with multi-touch support.
- Android supports multitasking, i.e. many applications will run at the same time, like in a computer. This is not possible with simple mobile phones and many other smartphones.
- Allnewmeansofconnectivityaresupport:GSM,3G,4G,Wi-Fi,Bluetooth,GPSetc.
- Androidsupportsmanylanguages,includingthosewithright-to-lefttext.
- Multimediamessagingsystem(MMS)issupported.

- Java runs great on Android. Applications for Android are developed in Java, but instead of a Java Runtime Environment, Android uses the Dalvik Executer, which is lighter on resources.
- Androidsupportsmost voiceandvideomediaformats,includingstreamingmedia.
- Additional hardware like sensors, gaming devices, other touchscreens can be integrated in Android.
- Voice and Video over IP. VoIP has many benefits, and Android manages camerasand has embedded support for seamless use of VoIP for free and cheap calls.
- On versions 2.2 and up, tethering is possible, which is the ability to use the Android device as a mobile WiFi hot spot.

Comparison chart

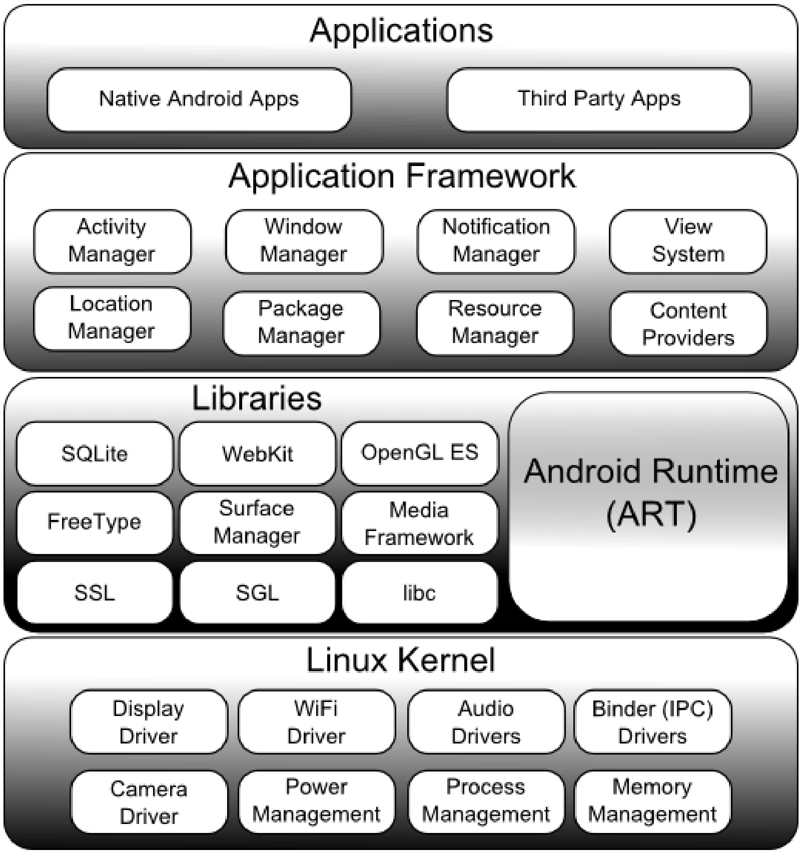
	Andriod	**iOS**
Sourcemodel	Open source	Closed,withopensourcecomponents.
OSfamily	Linux	OSX,UNIX
Initial release	September23,2008	July 29, 2007
Customizability	Alot.Canchangealmostanything.	Limitedunless jailbroken
Developer	Google,OpenHandset Alliance	AppleInc.
Widgets	Yes	No,except in NotificationCenter
Available language(s)	100+Languages	34Languages
Filetransfer	Easier than iOS. Using USB port and Android File Transfer desktop app. Photos can be transferred via USB without apps.	More difficult. Media files can be transferred using iTunes desktop app. Photos can be transferred out via USB without apps.
Availableon	Many phones and tablets.Major manufacturers are Samsung, Motorola, LG, HTC and Sony.. Nexus and Pixel line of devices is pure Android, others bundle manufacturer software.	iPod Touch, iPhone, iPad,AppleTV (2nd and 3rd generation)
Calls and messaging	Google Hangouts. 3rd party apps like Facebook Messenger, WhatsApp, Google Duo and Skype all work on	iMessage,FaceTime(withotherApple devices only). 3rd party apps like Google Hangouts, Facebook

	Androidand iOSboth.	Messenger, WhatsApp, Google Duo and Skype all work on Android and iOS both.
Internet browsing	Google Chrome (or Android Browser on older versions; other browsers are available)	MobileSafari(Otherbrowsersare available)
App store , Affordability and interface	Google Play – 1,000,000+ apps. Other appstores likeAmazon andGetjaralso distribute Android apps. (unconfirmed ".APKs")	Appleapp store– 1,000,000+ apps
Videochat	GoogleDuoand other3rd party apps	FaceTime (Apple devices only) and other 3rd party apps
Voice commands	GoogleNow, GoogleAssistant	Siri
Working state	Current	Current
Maps	GoogleMaps	Apple Maps (Google Maps also availableviaaseparate app download)
Latest stable release and Updates	Android8.0.0, Oreo(Aug21,2017)	11(Sep 19, 2017)
Alternativeapp stores and side loading	Several alternative app storesother thantheofficialGooglePlayStore.(e.g. Aptoide, Galaxy Apps)	Appleblocks3rdpartyappstores.The phone needs to be jailbroken if you want to download apps from other stores.
Batterylifeand management	Many Android phone manufacturers equip their devices with large batteries with a longer life.	Applebatteriesaregenerallynotasbig as the largest Android batteries. However, Apple is able to squeeze decent battery life via hardware/software optimizations.
Opensource	Kernel,UI,andsomestandardapps	The iOS kernel is not open source but isbasedontheopen-sourceDarwinOS.
Filemanager	Yes. (Stock Android File Manager includedondevicesrunningAndroid	Notavailable

	7.1.1)	
Photos & Videos backup	Appsavailableforautomaticbackupof photos and videos. Google Photos allows unlimited backup of photos. OneDrive, Amazon Photos and Dropbox are other alternatives.	Up to 5 GB of photos and videos can be automatically back up with iCloud. All other vendors like Google, Amazon, Dropbox, Flickr and Microsoft have auto-backup apps for both iOS and Android.
Security	Android software patches are available soonest to Nexus device users. Manufacturers tend to lag behind in pushing out these updates. So at any given time a vast majority of Android devices are not running updated fully patched software.	Most people will never encounter a problem with malware because they don't go outside the Play Store for apps.Apple'ssoftwareupdatessupport older iOS devices also.
Rooting, bootloaders, and jailbreaking	Access and complete control over your device is available and you can unlock the bootloader.	Complete control over your device is not available.
Cloud services	Native integration with Google cloud storage. 15GB free, $2/mo for 100GB, 1TB for $10. Apps available for Amazon Photos, OneDrive and Dropbox.	Native integration with iCloud. 5GB free,50GBfor$1/mo,200GBfor $3/mo, 1TB for $10/mo. Apps availableforGoogleDrive andGoogle Photos,AmazonPhotos,OneDriveand Dropbox.
Interface	TouchScreen	TouchScreen
Supported versions	Android5.0&later(Android4.4is also supported but with patches)	iOS 8 &later
Firstversion	Android1.0,Alpha	iOS 1.0

11. IOS ANDANDROIDARCHITECTUREANDSDKFRAMEWORK

1. AndroidArchitecture



AndroidSystem Architecture

TheAndroidsoftwarestackgenerallyconsistsofaLinuxkernelandacollectionofC/C++ libraries that is exposed through an application framework that provides services, and management of the applications and run time.

LinuxKernel

Android was created on the open source kernel of Linux. One main reason for choosing this kernel was that it provided proven core features on which to develop the Android operating system. The features of Linux kernel are:

1. **Security:**

TheLinuxkernel handlesthesecuritybetweentheapplicationandthesystem.

2. **MemoryManagement:**

It efficiently handles the memory management thereby providing the freedom to develop our apps.

3. **ProcessManagement:**

Itmanagestheprocess well,allocatesresourcesto processeswhenevertheyneedthem.

4. **NetworkStack:**

Iteffectivelyhandlesthenetworkcommunication.

5. **DriverModel:**

It ensuresthat theapplication works.Hardwaremanufacturers canbuild theirdrivers into the Linux build.

Libraries:

Running on the top of the kernel, the Android framework was developed with various features. It consists of various C/C++ core libraries with numerous of open source tools. Some of these are:

1. **TheAndroidruntime:**
   The Android runtime consist of core libraries of Java and ART(the Android RunTime). Older versions of Android (4.x and earlier) had Dalvik runtime.
2. **OpenGL(graphicslibrary):**
   Thiscross-language,cross-platformapplicationprograminterface(API)isusedtoproduce 2D and 3D computer graphics.
3. **WebKit:**
   This open source web browser engine provides all the functionality to display web content and to simplify page loading.
4. **Mediaframeworks:**
   Theselibrariesallowyoutoplayandrecordaudioandvideo.
5. **SecureSocketLayer(SSL):**
   TheselibrariesarethereforInternetsecurity.

AndroidRuntime:

It is the third section of the architecture. It provides one of the key components which is called Dalvik Virtual Machine. It acts like Java Virtual Machine which is designed specially forAndroid. Androiduses it'sowncustom VMdesignedto ensurethat multiple instances run efficiently on a single device.

TheDelvikVMusesthedevice'sunderlyingLinuxkerneltohandlelow-level functionality,including security,threading and memory management.

Application Framework

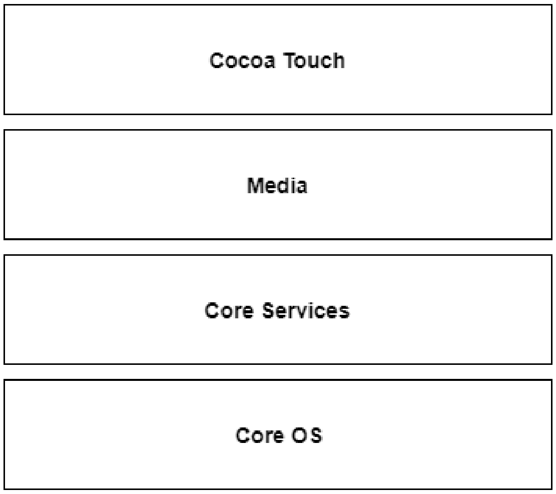
The Android team has built on a known set proven libraries, built in the background, and all of it these is exposed through Android interfaces. These interfaces warp up all the variouslibrariesandmakethemusefulfortheDeveloper.Theydon'thavetobuildanyof the functionality provided by the android. Some of these interfaces include:

1. **ActivityManager:**
   Itmanagestheactivitylifecycleandtheactivity stack.
2. **Telephony Manager:**
   It provides access to telephony services as related subscriber information, such as phone numbers.
3. **View System:**
   Itbuildstheuserinterfacebyhandlingtheviewsandlayouts.
4. **Location manager:**
   Itfindsthedevice'sgeographic location.

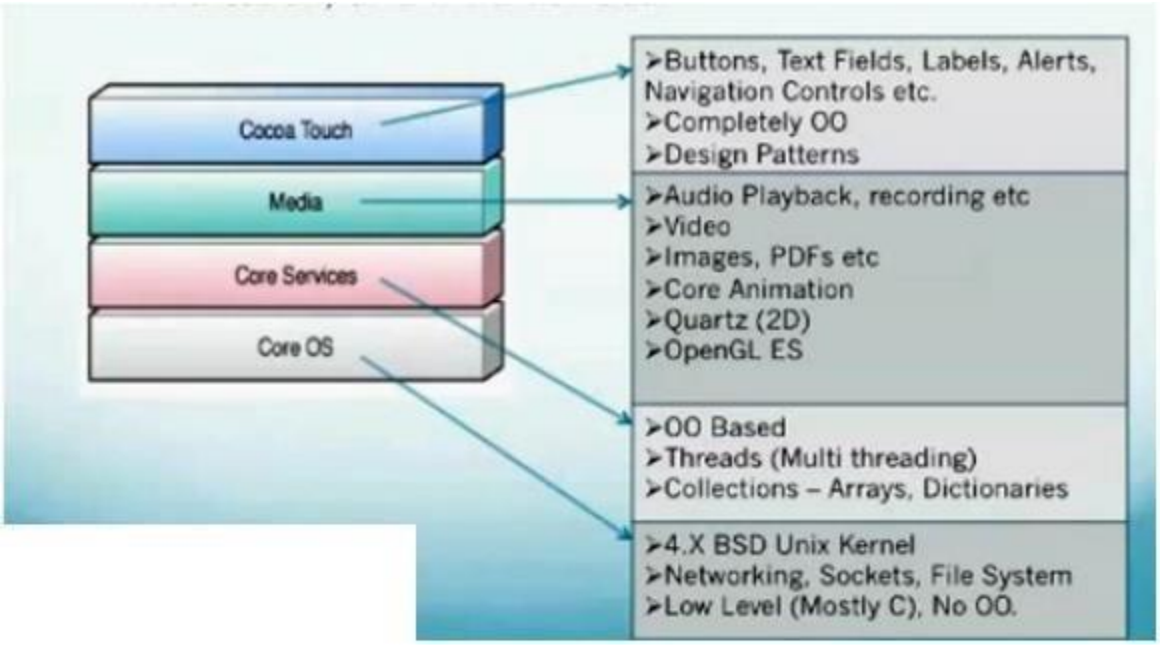
Applications:

Android applications can be found at the topmost layer. At application layer we write our application to be installed on this layer only. Examples of applications are Games, Messages, Contacts etc.

2. iOSArchitecture

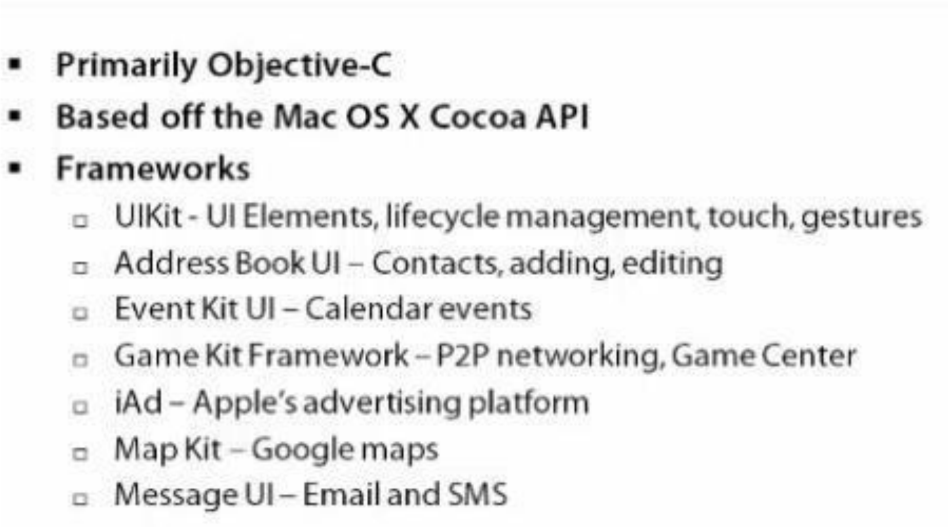


Apple iOS Architecture


Cocoa Touch
Media
Core Services
Core OS
➢Buttons, Text Fields, Labels, Alerts, Navigation Controls etc.
➢Completely OO
➢Design Patterns
➢Audio Playback, recording etc
➢Video
➢Images, PDFs etc
➢Core Animation
➢Quartz (2D)
➢OpenGL ES
➢OO Based
➢Threads (Multi threading)
➢Collections – Arrays, Dictionaries
➢4.X BSD Unix Kernel
➢Networking, Sockets, File System
➢Low Level (Mostly C), No OO.

CocoaTouch Layer

The Cocoa Touch layer sits at the top of the iOS stack and contains the frameworks thataremostcommonlyusedbyiPhoneapplicationdevelopers.CocoaTouchisprimarilywritten inObjective-C,isbasedonthestandardMacOSXCocoaAPI(asfoundonApple desktop and laptop computers) and has been extended and modified to meet the needs of the iPhone.



- **Primarily Objective-C**
- **Based off the Mac OS X Cocoa API**
- **Frameworks**
  - UIKit - UI Elements, lifecycle management, touch, gestures
  - Address Book UI – Contacts, adding, editing
  - Event Kit UI – Calendar events
  - Game Kit Framework – P2P networking, Game Center
  - iAd – Apple's advertising platform
  - Map Kit – Google maps
  - Message UI – Email and SMS

TheiOSMediaLayer

TheroleoftheMedialayeristoprovideiOSwithaudio,video,animationandgraphics capabilities. As with the other layers comprising the iOS stack, the Media layer comprises a number of frameworks that may be utilized when developing iPhone apps.

CoreOS Layer:

All theiOS technologies arebuild on thelow level features provided by theCore OS layer. These technologies include Core Bluetooth Framework, External Accessory Framework, Accelerate Framework, Security Services Framework, Local Authorisation Framework etc.

iOSCoreServices

TheiOSCoreServiceslayerprovidesmuchofthefoundationonwhichthepreviously referenced layers are built

*******************EndofAndriod andiOSArchitecture Framework*************

12. THEiOS MEDIA LAYER

- The role of the Media layer is to provide iOS with audio, video, animation and graphics capabilities.
- As with the other layers comprising the iOS stack, the Media layer comprises a numberof frameworks that may be utilized when developing iPhone apps.

CoreVideoFramework(CoreVideo.framework)

A new framework introduced with iOS 4 to provide buffering support for the Core Media framework.Thismaybeutilizedbyapplicationdevelopersitistypicallynotnecessarytousethis framework.

CoreTextFramework(CoreText.framework)

- TheiOSCoreTextframeworkisaC-basedAPIdesignedtoeasethehandlingofadvanced text layout and font rendering requirements.

ImageI/OFramework (ImageIO.framework)

- TheImageIOframework,thepurposeofwhichistofacilitatetheimportingandexporting of image data and image metadata, was introduced in iOS 4.
- The framework supports a wide range of image formats including PNG, JPEG, TIFF and GIF.

AssetsLibraryFramework(AssetsLibrary.framework)

- TheAssetsLibraryprovidesamechanismforlocatingandretrievingvideoandphotofiles located on the iPhone device.
- Inadditiontoaccessingexistingimagesandvideos,thisframeworkalsoallowsnewphotos and videos to be saved to the standard device photo album.

CoreGraphicsFramework (CoreGraphics.framework)

- The iOS Core Graphics Framework (otherwise known as the Quartz 2D API) provides a lightweight two dimensional rendering engine.
- FeaturesofthisframeworkincludePDFdocumentcreationandpresentation,vectorbased drawing,transparentlayers,pathbaseddrawing,anti-aliasedrendering,colormanipulation and management, image rendering and gradients.
- Thosefamiliar with the Quartz2D APIrunningon MacOS X will bepleased to learn that the implementation of this API is the same on iOS.

QuartzCoreFramework(QuartzCore.framework)

- The purpose of the Quartz Core framework is to provide animation capabilities on the iPhone.
- It provides the foundation for the majority of the visual effects and animation used by the UIKit framework and provides an Objective-C based programming interface for creation of specialized animation within iPhone apps.

OpenGLESframework (OpenGLES.framework)

- For many years the industry standard for high performance 2D and 3D graphics drawing has been OpenGL.
- Originally developed by the now defunct Silicon Graphics, Inc (SGI) during the 1990s in the form of GL, the open version of this technology (OpenGL) is now under the care of a non-profit consortium comprising a number of major companies including Apple, Inc., Intel, Motorola and ARM Holdings.

- OpenGL for Embedded Systems (ES) is a lightweight version of the full OpenGL specification designed specifically for smaller devices such as the iPhone,iOS 3 or later supports both OpenGL ES 1.1 and 2.0 on certain iPhone models (such as the iPhone 3GS and iPhone 4). Earlier versions of iOS and older device models support only OpenGL ES version 1.1.

iOSAudio Support

- iOS is capable of supporting audio in AAC, Apple Lossless (ALAC), A-law, IMA/ADPCM, Linear PCM, µ-law, DVI/Intel IMA ADPCM, Microsoft GSM 6.10 and AES3-2003 formats through the support provided by the following frameworks.

AVFoundationframework (AVFoundation.framework)

- An Objective-C based framework designed to allow the playback, recording and management of audio content.

Core Audio Frameworks (CoreAudio.framework, AudioToolbox.framework and AudioUnit.framework)

- TheframeworksthatcompriseCoreAudioforiOSdefinesupportedaudiotypes,playback and recording of audio files and streams and also provide access to the device's built-in audio processing units.

OpenAudioLibrary(OpenAL)

- OpenALisacrossplatformtechnologyusedtoprovidehigh-quality,3Daudioeffects(also referred to as positional audio).
- Positionalaudiocanbeusedinavarietyofapplicationsthoughistypicallyusingtoprovide sound effects in games.

MediaPlayerframework (MediaPlayer.framework)

- The iOS Media Player framework is able to play video in .mov, .mp4, .m4v, and .3gp formats at a variety of compression standards, resolutions and frame rates.

CoreMidiFramework(CoreMIDI.framework)

- IntroducediniOS4,theCoreMIDIframeworkprovidesanAPIforapplicationstointeract with MIDI compliant devices such as synthesizers and keyboards viathe iPhone's dock connector.

******************************EndofiOSMediaLayer*****************************

13. THEiOSCORESERVICESLAYER

- The iOS Core Services layer provides much of the foundation on which the previously referenced layers are built and consists of the following frameworks.

AddressBookframework (AddressBook.framework)

- TheAddressBookframeworkprovidesprogrammaticaccesstotheiPhoneAddressBook contact database allowing applications to retrieve and modify contact entries.

CFNetworkFramework(CFNetwork.framework)

- The CFNetwork framework provides a C-based interface to the TCP/IP networking protocol stack and low level access to BSD sockets.
- ThisenablesapplicationcodetobewrittenthatworkswithHTTP,FTPandDomainName servers and to establish secure and encrypted connections using Secure Sockets Layer (SSL) or Transport Layer Security (TLS).

CoreDataFramework (CoreData.framework)

- This framework is provided to ease the creation of data modeling and storage in Model-View-Controller (MVC) based applications.
- Use of the Core Data framework significantly reduces the amount of code that needs tobewrittentoperformcommontaskswhenworkingwithstructureddatainanapplication.

CoreFoundationFramework(CoreFoundation.framework)

- The Core Foundation is a C-based Framework that provides basic functionality such as data types, string manipulation, raw block data management, URL manipulation, threads and run loops, date and times, basic XML manipulation and port and socket communication.AdditionalXML capabilitiesbeyondthoseincluded withthisframework are provided via the libXML2 library.
- Though this is a C-based interface, most of the capabilities of the Core Foundation framework are also available with Objective-C wrappers via the Foundation Framework.

CoreMediaFramework(CoreMedia.framework)

- TheCoreMedia frameworkisthe lower level foundationuponwhich the AV Foundation layer is built.
- While most audio and video tasks can, and indeed should, be performed using the higher level AV Foundation framework, access is also provided for situations where lower level control is required by the iOS application developer.

CoreTelephonyFramework(CoreTelephony.framework)

- The iOS Core Telephony framework is provided to allow applications to interrogate the device for information about the current cell phone service provider and to receive notification of telephony related events.

EventKitFramework(EventKit.framework)

- An API designed to provide applications with access to the calendar and alarms on the device.

FoundationFramework(Foundation.framework)

- TheFoundationframeworkisthestandardObjective-Cframeworkthatwillbefamiliarto those that have programmed in Objective-C on other platforms (most likely Mac OS X).

- ❖ Essentially, this consists of Objective-C wrappers around much of the C-based Core Foundation Framework.

CoreLocationFramework (CoreLocation.framework)

- ❖ The Core Location framework allows you to obtain the current geographical location of thedevice(latitudeandlongitude)andcompassreadingsfromwithyourownapplications.
- ❖ The method used by the device to provide coordinates will depend on the data availableatthetimetheinformationisrequestedandthehardwaresupportprovidedbytheparticular iPhone model on which the app is running (GPS and compass are only featuredon recent models).
- ❖ This will eitherbebased on GPS readings, Wi-Fi networkdataor cell towertriangulation (or some combination of the three).

MobileCoreServicesFramework (MobileCoreServices.framework)

- ❖ The iOS Mobile Core Services framework provides the foundation for Apple’s Uniform Type Identifiers (UTI) mechanism, a system for specifying and identifying data types.
- ❖ A vast range of predefined identifiers have been defined by Apple including such diverse data types as text, RTF, HTML, JavaScript, PowerPoint .ppt files, PhotoShop images and MP3 files.

StoreKitFramework(StoreKit.framework)

- ❖ The purpose of the Store Kit framework is to facilitate commerce transactions between your application and the
- ❖ AppleAppStore.Priortoversion3.0ofiOS,itwasonlypossibletochargeacustomerfor an app at the point that they purchased it from the App Store. iOS 3.0 introduced the conceptofthe“inapppurchase”wherebytheusercanbegiventheoptionmakeadditional payments from within the application.
- ❖ This might, for example, involve implementing a subscription model for an application, purchasing additional functionality oreven buying afastercarforyou to drivein aracing game.

SQLitelibrary

- ❖ Allows for a lightweight, SQL based database to be created and manipulated from within your iPhone application.

SystemConfigurationFramework(SystemConfiguration.framework)

- ❖ The System Configuration framework allows applications to access the network configuration settings of the device to establish information about the “reachability” of thedevice(forexample whetherWi-Fiorcellconnectivityisactiveandwhetherandhow traffic can be routed to a server).

QuickLookFramework(QuickLook.framework)

- ❖ One of the many new additions included in iOS 4, the Quick Look framework provides a usefulmechanismfordisplayingpreviewsofthecontentsoffilestypesloadedontothe device (typically viaaninternetor networkconnection)for whichtheapplicationdoes not already provide support.

- Fileformattypessupportedbythisframeworkincludei Work,MicrosoftOfficedocument, RichTextFormat,AdobePDF, Imagefiles,public.textfilesandcommaseparated(CSV).

*****************************EndofCoreServicesLayer*********************

14. TheiOSCoreOSLayer

- TheCoreOSLayeroccupiesthebottompositionoftheiOSstackand,assuch,sitsdirectly on top of the device hardware.
- Thelayerprovidesavarietyofservicesincludinglowlevelnetworking,accesstoexternal accessories and the usual fundamental operating system services suchas memory management, file system handling and threads.

AccelerateFramework (Accelerate.framework)

- IntroducedwithiOS4,theAccelerateFrameworkprovidesahardwareoptimizedC-based API for performing complex and large number math, vector, digital signal processing (DSP) and image processing tasks and calculations.

ExternalAccessoryframework(ExternalAccessory.framework)

- Provides the ability to interrogate and communicate with external accessories connected physically to the iPhone via the 30-pin dock connector or wirelessly via Bluetooth.

SecurityFramework(Security.framework)

- TheiOS Security framework provides all the security interfaces you would expect to find onadevicethatcanconnecttoexternalnetworksincludingcertificates,publicandprivate keys, trust policies, keychains, encryption, digests and Hash-based Message Authentication Code (HMAC).

System(LibSystem)

- Aswehavepreviouslymentioned,theiOSisbuiltuponaUNIX-like foundation.
- TheSystemcomponentoftheCoreOSLayerprovidesmuchthesamefunctionalityasany otherUNIXlikeoperatingsystem.Thislayerincludestheoperatingsystemkernel (based on the Mach kernel developed by Carnegie Mellon University) and device drivers.
- The kernel is the foundation on which the entire iOS is built and provides the low level interface to the underlying hardware.
- Amongst other things the kernel is responsible for memory allocation, process lifecycle management, input/output, inter-process communication, thread management, low level networking, file system access and thread management.
- As an app developer your access to the System interfaces is restricted for security and stability reasons. Those interfaces that are available to you are contained in a C-based library called LibSystem.

**********************EndodCoreOSLayer****************************

15. FILESYSTEMBASICS

- A*filesystem*handlesthepersistentstorageofdatafiles,apps,andthefilesassociated with the operating system itself. Therefore, the file system is one of the fundamental resources used by all processes.

- APFS is the default file system in macOS, iOS, watchOS, and tvOS. APFS replaces HFS+asthedefaultfilesystemforiOS10.3andlater,andmacOSHighSierraandlater macOS additionally supports a variety of other formats, as described in Supported FileSystems.

- Thefilesystemusesdirectoriestocreateahierarchical organization

- Before you begin writing code that interacts with the file system, you should first understandalittleabouttheorganizationoffilesystemandtherulesthatapplytoyour code.

- Asidefromthebasictenetthatyoucannotwritefilestodirectoriesforwhichyoudonot have appropriate security privileges, apps are also expected to be good citizens and put files in appropriate places.

- Precisely where you put files depends on the platform, but the overarching goal is to makesurethattheuser'sfilesremaineasilydiscoverableandthatthefilesyourcodeuses internally are kept out of the user's way.

AbouttheiOS FileSystem

- TheiOSfilesystemisgearedtowardapps running ontheirown.Tokeepthesystem simple,usersofiOSdevicesdonothavedirectaccesstothefilesystemandappsare expected to follow this convention.

iOSStandardDirectories:WhereFilesReside?

- Forsecuritypurposes,aniOSapp'sinteractionswiththefilesystemarelimitedtothe directories inside the app's sandbox directory.

- During installation of a new app, the installer creates a number of container directoriesfor the app inside the sandbox directory.

- Eachcontainerdirectoryhasaspecific role.

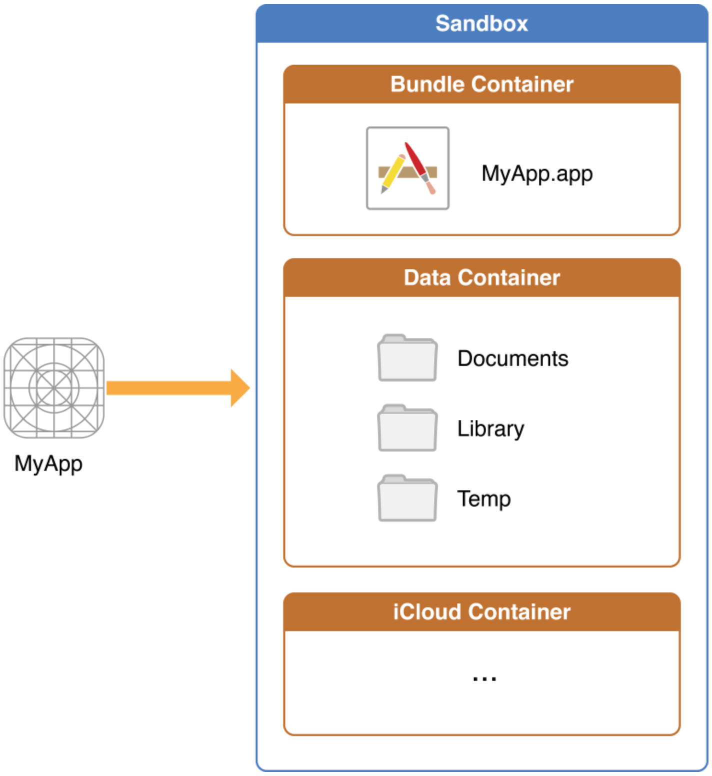
- The bundle container directory holds the app's bundle, whereas the data container directory holds data for both the app and the user.

- Thedatacontainerdirectoryisfurtherdividedintoanumberofsubdirectoriesthatthe app can use to sort and organize its data.

- Theappmayalsorequestaccesstoadditionalcontainerdirectories—forexample,the iCloud container—at runtime.

- Thesecontainerdirectoriesconstitutetheapp'sprimaryviewofthefilesystem. The Figureshows a representation of the sandbox directory for an app.

- AniOSappoperating within itsownsandbox directory



- Anappisgenerallyprohibitedfromaccessingorcreatingfilesoutsideitscontainer directories.

- Oneexceptiontothisruleiswhenanappusespublicsysteminterfacestoaccessthings such as the user's contacts or music.

- Inthosecases,thesystemframeworksusehelperappstohandleanyfile-related operations needed to read from or modify the appropriate data stores.

- Tablelistssomeofthemoreimportantsubdirectoriesinsidethesandboxdirectoryand describes their intended usage.
- Thistablealsodescribesanyadditionalaccessrestrictionsforeachsubdirectoryand points out whether the directory's contents are backed up by iTunes and iCloud.

Table	CommonlyuseddirectoriesofaniOSapp
Directory	**Description**

AppName`.app`	Thisistheapp’sbundle.Thisdirectorycontainstheappandallofits resources. You cannot write to this directory. To prevent tampering, the bundle directory is signed at installation time. Writing to this directory changes the signature and prevents your app from launching. You can, however, gainread-onlyaccesstoanyresourcesstoredintheappsbundle.Formore information, see the *Resource Programming Guide* The contents of this directory are not backed up by iTunes or iCloud. However,iTunesdoesperformaninitialsyncofanyappspurchasedfrom the App Store.
`Documents/`	Use this directory to store user-generated content. The contents of this directorycanbemadeavailabletotheuserthroughfilesharing;therefore, his directory should only contain files that you may wish to expose to the user. Thecontents ofthis directory arebacked up byiTunes and iCloud.
`Documents/Inbox`	Use this directory to access files that your app was asked to open by outsideentities.Specifically,theMailprogramplacesemailattachments associated with your app in this directory. Document interaction controllers may also place files in it. Your app can read and delete files in this directory but cannot create new filesorwritetoexistingfiles.Iftheusertriestoeditafileinthisdirectory, your app must silently move it out of the directory before making any changes. Thecontents ofthis directoryarebacked up byiTunes and iCloud.
`Library/`	Thisisthetop-leveldirectoryforanyfilesthatarenotuserdatafiles.You typically put files in one of several standard subdirectories. iOS apps commonly use the `Application Support` and `Caches` subdirectories; however, you can create custom subdirectories. Usethe`Library`subdirectoriesforanyfilesyoudon’twantexposedto the user. Your app should not use these directories for user data files. The contents of the `Library` directory (with the exception of the`Caches`subdirectory)arebackedupbyiTunesandiCloud. For additional information about the Library directory and its commonly usedsubdirectories,seeTheLibraryDirectoryStoresApp-SpecificFiles.
`tmp/`	Use this directory to write temporary files that do not need to persist between launches of your app. Your app should remove files from this directorywhentheyarenolongerneeded;however,thesystemmaypurge this directory when your app is not running.

	Thecontentsofthisdirectoryarenotbacked upbyiTunesor iCloud.

AniOSappmaycreateadditionaldirectoriesintheDocuments,Library,andtmpdirectories. You might do this to better organize the files in those locations.

WhereYouShould PutYourApp’sFiles

- Topreventthesyncingandbackupprocessesonios devicesfromtakingalongtime,be selective about where you place files. Apps that store large files can slow down the process of backing up to iTunes or iCloud.

- These apps can also consume a large amount of a user's available storage, which may encouragetheusertodeletetheappordisablebackupofthatapp'sdatatoiCloud.With this in mind, you should store app data according to the following guidelines:

- **Put user data in Documents/.** User data generally includes any files you might want to expose to the user—anything you might want the user to create, import, delete or edit. For a drawing app, user data includes any graphic files the user might create. For a text editor,itincludesthetextfiles.Videoandaudioappsmayevenincludefilesthattheuser has downloaded to watch or listen to later.

- **Putapp-createdsupportfilesin theLibrary/Applicationsupport/**directory. In general,thisdirectoryincludesfilesthattheappusestorunbutthatshouldremainhidden fromtheuser.Thisdirectorycanalsoincludedata files,configurationfiles, templatesand modified versions of resources loaded from the app bundle.

- **RememberthatfilesinDocuments/andApplicationSupport/**arebackedupby default. You can exclude files from the backup by calling using the**NSURLIsExcludedFromBackupKeykey**.Anyfilethatcanbere-createdor downloadedmustbeexcludedfromthebackup.Thisisparticularlyimportantforlarge media files. If your application downloads video or audio files, makesure they arenot included in the backup.

- **Puttemporarydatain thetmp/**directory.Temporarydatacomprisesany datathatyou do not need to persist for an extended period of time. Remember to delete those files whenyouaredonewiththemsothattheydonotcontinuetoconsumespaceontheuser’s device. The system will periodically purge these files when your app is not running; therefore, you cannot rely on these files persisting after your app terminates.

- **Put data cache files in the Library/Caches/** directory. Cache data can be used for any data that needs to persist longer than temporary data, but not as long as a support file. Generally speaking, the application does not requirecache data to operate properly, but it can use cache data to improve performance.

- Examples of cache data include(but arenot limitedto)database cache files and transient, downloadable content. Note that the system may delete the **Caches/** directory to free up disk space, so your app must be able to re-create or download these files as needed.

PARTA

1. **Whatarethecomponents ofaLinux System?**
   Linuxsystemcomposedofthreemainmodules.They are:
   i) Kernel
   ii) System libraries
   iii) System utilities
2. **Whatarethemain supports fortheLinux modules?**
   TheModulesupport under Linux has three components. They are:
   i) Modulemanagement
   ii) Driverregistration
   iii) Conflictresolution mechanism.
3. **Defineshell.**
   A shell is a program that provides the traditional, text-only user interface for Linux and otherUnix-likeoperatingsystems.Itsprimaryfunctionistoreadcommandsthataretyped into the console.
4. **Whatismeantbykernel inLinux system?**
   Kernelisresponsibleformaintainingalltheimportantabstractionsoftheoperating system including such things as virtual memory and processes.
5. **Whatis meantbysystemLibraries?**
   Systemlibrariesdefineastandardsetoffunctionsthroughwhichapplicationscaninteract withthekernelandthatimplementmuchoftheoperating-systemfunctionalitythatdoesn't need the full privileges of kernel code.
6. **WhatismeantbysystemUtilities?**
   System Utilities are system programs that perform individual, specialized management tasks. Some of the system utilities may be invoked just to initialize and configure some aspect of the system and others may run permanently, handling such tasks as responding toincomingnetworkconnections,acceptinglogonrequestsfromterminalsorupdatinglog files.
7. **Whatdoyoumeantbyprocess?**
   Processisthebasiccontextwithininwhichalluser-requestedactivityisservicedwithin the OS.
8. **Whatismeantby Process-ID**
   A PID is an acronym for process identification number on a Linux or Unix-like operating system. A PID is automatically assigned to each process when it is created. A process is nothing but running instance of a program and each process has a unique PID on a Unix-like system.
9. **Whatismeantby personality?**
   Processpersonalityareprimarilyusedbyemulationlibrariestorequestthatsystemcall be compatible with certain version of UNIX
10. **Whatismeantbybuffercache?**
    Itisthekernel'smaincacheforblocked-orienteddevicessuchasdiskdriversandisthe main mechanism through which I/O to these devices is performed.
11. **Whatisthedisadvantageofstaticlinking?**
    Themaindisadvantageofstaticlinkingisthateveryprogramgeneratedmustcontain copies of exactly the same common system library functions.

12. **Whatis thefunction ofmodule management?**
    Themodulemanagement allowsmodulestobeloadedintomemoryandtotalktotherest of the kernel.
13. **Whatis thefunctionofdriverregistration?**
    Driverregistrationallowsmodulestotelltherestofthekernelthatanewdriverhas become available.
14. **Whatisthefunction ofconflictresolutionmechanism?**
    Thismechanismallowsdifferentdevicedriverstoreservehardwareresourcesandto protect those resources from accidental use by another driver.
15. **Whatis meantbydevicedrivers?**
    Devicedriversincludei) characterdevicessuchasprinters,terminalsii)Blockdevices (including all disk drives) and network interface devices.
16. **Whatisacharacterdevice?**
    A device driver which does not offer random access to fixed blocks of data. A character device driver must register a set of functions which implement the driver's various file I/O operations.
17. **Whatis MobileOS?**
    A mobile operating system (mobile OS) is an OS built exclusively for a mobile device, such as a smartphone, personal digital assistant (PDA), tablet or other embedded mobile OS.
18. **Whatis iOS?**
    iOS is a mobile operating system created and developed by Apple Inc. exclusively for its hardware. It is the operating system that presently powers many of the company's mobile devices,includingtheiPhone,iPad,andiPodTouch.Itisthesecondmost popularmobile operating system globally after Android.
19. **Listtheservicesavailablein iOS.**
    i) CocoaTouch
    ii) Medialayer
    iii) Servicelayer
    iv) CoreOS layer
20. **ListthefeaturesofiOS.**
    i) System fonts
    ii) Folders
    iii) Notificationcenter
    iv) Accessibility
    v) Multitasking
    vi) Switching Applications
    vii) Taskcompletion
    viii) Backgroundaudio
    ix) VoiceoverIP
    x) BackgroundLocation
    xi) Push notification
21. **Listtheadvantages of iOS**
    - Bestgamingexperience.
    - Avast numberofapplications.

- Suitsforbusiness and gaming.
- ExcellentUIandfluidresponsive.
- Thelatest versionhastwonotification menus.
- Excellentsecurity.
- Multitasking.
- Jailbreakingforcustomization.
- Wearablesaregettinglaunched.
- Feelis awesome.
- Excellentformedia entertainment.
- Multi-languagesupport.
- ApplePay Support.
- Quicksettings inthenotificationbar.

22. ListthedisadvantagesofiOS

- Ithas areviewprocess,whendeveloperswantto publishanapptheyneedtosend ittoAppleforreviewthattakesaround7daysandittakesevenmoreinsomecases.
- Applicationsareverylargewhencomparedtoothermobileplatforms
- UsingiOS arecostly Apps andno widget support
- Batteryperformanceisverypooron 3G
- Repaircosts areverypiracy
- Notflexibleonly supportsiOS devices

23. Listtheadvantages of Android

- AndroidIsMoreCustomizableCanchangealmost anything.
- InAndroid,anynewpublicationcanbedoneeasilyandwithoutany review process
- UseaDifferent Messaging Appfor SMS
- AndroidOffersanOpenPlatform
- EasyaccesstotheAndroidAppMarket
- Cost Effective
- Upcomingversionshaveasupport tosaveRAWimages
- BuiltinBetaTestingandstagedrollout

24. Listthedisadvantagesof Android

- Needinternetconnection
- Advertising
- Wastefulmemory
- Manyapplicationcontainviruses

25. HowareiOS andAndroidsimilar? Howaretheydifferent?

Similarities: Botharebasedonexistingkernel.Bothhavearchitecturethatusessoftware stacks. Both provide framework for developers

Difference: iOS is closed-source and Android is open source. iOS applications are developed in objective C, Android in java. Android uses a virtual machine, and iOS executes code natively.

26. **DescribesomechallengesofdesigningOSformobiledevicecomparedwith designing OS for traditional PC's**

- LessstoragecapacitymeanstheOSmustbemanagememory
- Lessprocessingpowerplusfewerprocessorsmeantheoperatingsystemcarefully apportion
- Processorstoapplications

Part-B

1. DrawaneatsketchofoverviewofiOSarchitectureandexplainindetail.(13)
2. DiscussprocessmanagementandschedulinginLINUX. (13)
3. IllustratesomeexistingSDKarchitectureimplementationframeworks.(13)
4. DescribeaboutthenetworkstructureofLINUXsystem.(13)
5. Explainindetailthedesignprinciples,kernelmodulesinLINUXsystem.(7+6)
6. Demonstratethefunctionsofthekernel,serviceandcommandlayersofOS.(13)
7. GeneralizetheimportanceofmemorymanagementinOperatingsystem.(13)
8. Explainindetailaboutfilesystemmanagementdonein LINUX.(13)
9. SummarizeInterProcessCommunicationwithsuitableexample.(13)
10. Analyze:
    a. mobileOS(5)ii)desktopOS(4)iii)multi-userOS(4)
11. CompareandcontrastAndriodOSandIOS.(13)
12. ExplainindetailaboutLinuxarchitecture. (13)
13. Comparethefunctionsofmedialayer,servicelayerandcoreOSlayer.
14. ExplainthebasicconceptsoftheLinux system
15. 2.Explainaboutkernelmodules
16. 3.ExplainindetailabouttheprocessmanagementinLinux
17. 4.ExplainindetailabouttheschedulinginLinux
18. 5.ExplaintheiOSarchitectureandvariouslayersavailableiniOS
19. Discussaboutvariousservicesinthemedialayer
20. DiscussaboutvariousservicesintheiOScoreOSlayer
    DiscussaboutvariousservicesintheiO

    21.SserviceOSlayer