MODULE: 5

CONTENTS:

❖ **FileSystem**

- Fileconcept;
- Accessmethods;
- Directorystructure;
- Filesystemmounting;
- Filesharing;
- Protection:

❖ **ImplementationofFileSystem**

- Filesystemstructure;
- Filesystemimplementation;
- Directoryimplementation;
- Allocationmethods;
- Freespacemanagement.

❖ **SecondaryStorageStructures**

- Massstoragestructures
- Diskstructure
- Diskattachments
- Diskscheduling
- Diskmanagement
- swapspacemanagement

❖ **Protection**

- Goalsofprotection
- Principlesofprotection
- Domainofprotection
- Accessmatrix
- Accesscontrol
- Revocationofaccessrights
- Capability-Basedsystems

MODULE4
FILE SYSTEM

FILECONCEPT

FILE:

- *A* file is a named collection of related information that is recorded on secondarystorage.
- The information in a file is defined by its creator. Many different types of informationmay bestoredinafilesourceprograms,objectprograms, executableprograms,numericdata,text,payrollrecords,graphicimages,soundrecordings, andsoon.

*A*filehasacertaindefinedwhichdependsonitstype.

- *Atext*fileisasequenceofcharactersorganizedinto lines.
- *Asource*fileisasequenceofsubroutinesandfunctions,eachofwhichisfurtherorganizedasdeclarationsfollowedbyexecutablestatements.
- An*object*fileisasequenceofbytesorganizedintoblocksunderstandablebythesystem's linker.
- An*executable*fileisaseriesofcodesectionsthattheloadercanbringintomemoryandexecute.

FileAttributes

- Afileisnamed,fortheconvenienceofitshumanusers,andisreferredtobyitsname.Anameisusuallyastringofcharacters,suchas*example*.c
- Whenafileisnamed,itbecomesindependentoftheprocess,theuser,andeventhesystemthatcreatedit.

Afile'sattributesvaryfromoneoperatingsystemtoanotherbuttypicallyconsistofthese:

- **Name**:Thesymbolicfilenameistheonlyinformationkeptinhuman readableform.
- **Identifier:**Thisunique tag,usuallyanumber, identifiesthe file withinthefilesystem;itisthenon-human-readablenameforthefile.
- **Type:**Thisinformationisneededforsystems thatsupportdifferenttypesoffiles.
- **Location:** This information is a pointer to a device and to the location of the file onthatdevice.
- **Size:**Thecurrentsizeofthefile(inbytes,words,orblocks)andpossiblythemaximumallowedsizeareincludedinthisattribute.
- **Protection:**Access-controlinformationdetermineswhocandoreading,writing,executing,and so on.
- **Time, date, and user identification:** This information may be kept for creation, lastmodification, and last use. These data can be useful for protection, security, and

usagemonitoring.

The information about all files is kept in the directory structure, which also resides onsecondary storage. Typically, a directory entry consists of the file's name and its uniqueidentifier.Theidentifierinturnlocatestheotherfileattributes.

FileOperations

A file is an abstract data type. To define a file properly, we need to consider the operationsthatcanbeperformedonfiles.

1. **Creatingafile:**Twostepsarenecessarytocreateafile,
   a) Spaceinthefilesystemmustbefoundforthefile.
   b) Anentryforthenewfile mustbe madeinthedirectory.
2. **Writing a file:**To write a file, we make a system call specifying both the name of thefile and the information to be written to the file. Given the name ofthe file, thesystem searches the directory to find the file's location. The system must keep a writepointer to the location in the file where the next write is to take place. The writepointermustbeupdatedwheneverawriteoccurs.
3. **Reading a file:**To read from a file, we use a system call that specifies the name of thefileandwherethenextblockofthefileshouldbeput.Again,thedirectory issearched for the associated entry, and the system needs to keep a read pointer to thelocation in the file where the next read is to take place. Once the read has taken place,the read pointer isupdated. Because a process is usually either reading from orwriting to a file, the current operation location can be kept as a per-process currentfile-positionpointer.
4. **Repositioning within a file:**The directory is searched for the appropriate entry, andthe current-file-position pointer is repositioned to a given value. Repositioning withinafileneednotinvolveanyactualI/0.Thisfileoperationisalsoknownasfilesseek.
5. **Deleting a file:**To delete a file, search the directory for the named file. Having foundthe associated directory entry, then release all file space, so that it can be reused byotherfiles,anderasethedirectoryentry.
6. **Truncating a file:**The user may want to erase the contents of a file but keep itsattributes. Rather than forcing the user to delete the file and then recreate it, thisfunction allows all attributes to remain unchanged but lets the file be reset to lengthzeroand its file space released.

- Othercommonoperationsincludeappendingnewinformationtotheendofanexistingfile andrenaming anexistingfile.
- Most of the file operations mentioned involve searching the directory for the entryassociated with thenamed file.
- To avoid this constant searching, many systems require that an open () system call bemadebeforeafileisfirstusedactively.
- Theoperatingsystemkeepsasmalltable, calledthe**openfile table**containinginformationaboutallopenfiles.Whenafileoperationisrequested,thefileisspecifiedviaanindexintothistable,sonosearchingisrequired.

- The implementation of the open() and close() operations is more complicated in anenvironmentwhereseveralprocessesmayopenthefilesimultaneously
- Theoperatingsystemusestwolevelsof internaltables:
  1. Aper-processtable
  2. Asystem-widetable

Theper-processtable:

- Tracksallfilesthataprocesshasopen. Storedinthis tableisinformationregardingtheuseofthefilebytheprocess.
- Eachentryintheper-processtableinturnpointstoasystem-wideopen-filetable.

Thesystem-widetable

- contains process-independent information, such as the location of the file on disk,access dates, and file size. Once a file has been opened by one process, the system-widetableincludesanentryforthefile.

Severalpiecesofinformationareassociated withanopenfile.

1. **File pointer:** On systems that do not include a file offset as part of the read() andwrite() system calls, the system must track the last read write location as a current-file-position pointer. This pointer is unique to each process operating on the file andthereforemustbekeptseparatefromtheon-diskfileattributes.
2. **File-open count:** As files are closed, the operating system must reuse its open-filetable entries, or it could run out of space in the table. Because multiple processes mayhave opened a file, the system must wait for the last file to close before removing theopen-file table entry. The file-open counter tracks the number of opens and closes andreacheszero onthelastclose.Thesystemcanthenremovetheentry.
3. **Disk location of thefile:**Most file operations requirethe system tomodify datawithin the file. The information needed to locate the file on disk is kept in memory sothatthesystemdoesnothavetoreaditfromdiskforeachoperation.
4. **Access rights:** Each process opens a file inan accessmode. This information isstored on the per-process table so the operating system can allow or deny subsequentI/0requests.

FileTypes

- The operating system should recognize and support file types. If an operating systemrecognizesthe typeofafile,itcanthenoperateonthefileinreasonableways.
- A common technique for implementing file types is to include the type as part of thefilename.Thenameissplitintotwoparts-**anameandanextension**,usuallyseparatedbyaperiodcharacter
- Thesystemusestheextensiontoindicatethetypeofthefileandthetypeofoperationsthatcanbedoneonthatfile.

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, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rtf, doc	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	ps, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm, mp3, avi	binary file containing audio or A/V information

FileStructure

- File types also can be used to indicate the internal structure of the file. For instance sourceand object files have structures that match the expectations of the programs that read them.Certain files must conform to a required structure that is understood by the operatingsystem.
  For example: the operating system requires that an executable file have a specific structureso that it can determine where in memory to load the file and what the location of the firstinstructionis.
- The operating system support multiple file structures: the resulting size of the operatingsystem also increases. If the operating system defines five different file structures, it needstocontainthecodetosupportthesefilestructures.
- It is necessary to define every file as one of the file types supported by the operatingsystem.Whennewapplicationsrequireinformationstructuredinwaysnotsupportedby

theoperatingsystem,severeproblemsmayresult.

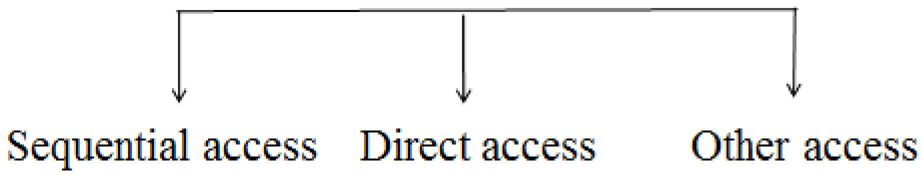
- Example: The Macintosh operating system supports a minimal number of file structures. Itexpectsfiles to contain two parts: aresource fork and datafork.
  - **Theresourcefork**containsinformationofinteresttotheuser.
  - **Thedatafork**containsprogramcodeordata

InternalFileStructure

- Internally, locating an offset within a file can be complicated for the operating system.Disk systems typically have a well-defined block size determined by the sizeofasector.
- All disk I/0 is performed in units of one block (physical record), and all blocks are thesame size. It is unlikely that the physical record size will exactly match the length of thedesiredlogicalrecord.
- Logical records may even vary in length. Packing a number of logical records intophysicalblocksisacommonsolutiontothisproblem.

ACCESSMETHODS

- Files store information. When it is used, this information must be accessed and readintocomputermemory. Theinformationin the filecanbeaccessedin severalways.
- Someofthecommon methodsare:



1. Sequentialmethods

- Thesimplestaccessmethodissequentialmethods.Informationinthefileisprocessedin order,onerecordaftertheother.
- Readsandwritesmakeupthebulkoftheoperationsonafile.
- Areadoperation(next-reads)readsthenextportionofthefileandautomaticallyadvancesafilepointer ,whichtrackstheI/Olocation
- Thewriteoperation(writenext)appendstotheendofthefileandadvancestotheendofthenewlywrittenmaterial.
- Afilecanberesettothebeginningandonsomesystems,aprogrammaybeabletoskipforwardorbackwardnrecordsforsomeintegern-perhapsonlyforn=1.

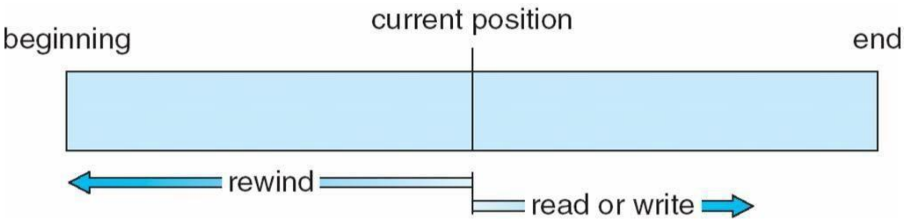


Figure:Sequential-accessfile.

2. DirectAccess

- A file is made up of fixed length logical records that allow programs to read andwriterecordsrapidlyinnoparticularorder.
- The direct-accessmethod is based on a disk model of a file, since disks allowrandom access to any file block. For direct access, the file is viewed as a numberedsequenceofblocksorrecords.
- Example: if we may read block 14, then read block 53, and then write block 7. Thereareno restrictionson the orderofreadingorwritingfora direct-accessfile.
- Direct-accessfilesareofgreatuseforimmediateaccesstolargeamountsofinformationsuchasData bases,wheresearchingbecomeseasyandfast.
- For the direct-access method, the file operations must be modified to include theblock number as a parameter. Thus, we have read n, where n is the block number,ratherthanreadnext,and·writenratherthanwritenext.
- An alternative approach is to retain read next and write next, as with sequentialaccess, and to add an operation position file to n, where n is the block number. Then,toaffectareadn,wewouldpositiontonandthenreadnext.

sequential access	implementation for direct access
reset	*cp* = 0;
read next	*read cp*; *cp* = *cp* + 1;
write next	*write cp*; *cp* = *cp* + 1;

Figure:Simulationofsequentialaccessonadirect-accessfile.

3. OtherAccessMethods:

- Other access methods can be built on top of a direct-access method. These methodsgenerallyinvolvetheconstructionofanindexforthefile.
- The **Index**, is like an index in the back of a book contains pointers to the variousblocks. To find a record in the file, we first search the index and then use the pointertoaccessthefiledirectlyandtofindthedesiredrecord.

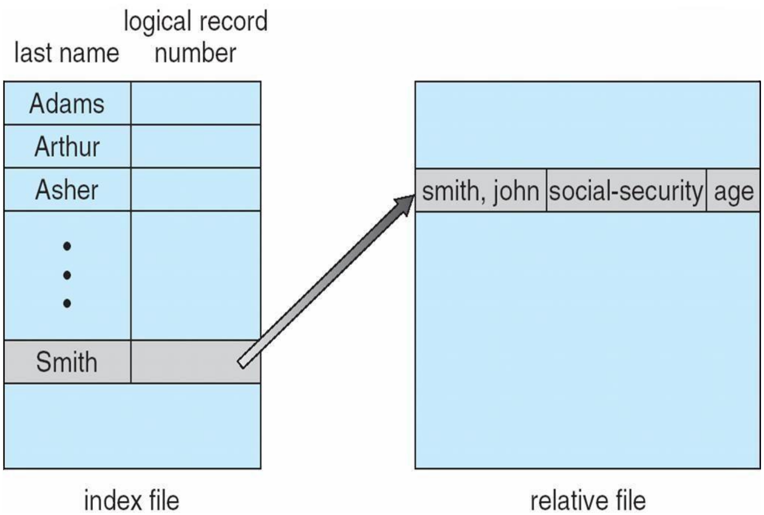


Figure:Exampleofindexandrelativefiles

DIRECTORYANDDISKSTRUCTURE

- Files are stored onrandom-access storagedevices, includinghard disks, opticaldisks,andsolidstate(memory-based) disks.
- Astoragedevicecanbeusedinitsentiretyforafilesystem.Itcanalsobesubdividedforfiner-grained control
- Diskcanbesubdividedintopartitions.EachdisksorpartitionscanbeRAIDprotectedagainstfailure.
- Partitionsalsoknownasminidisksorslices.Entitycontainingfilesystemknownasa volume. Each volume that contains a file system must also contain informationabout the files in the system. This information is kept in entries in a **device directory**or**volumetableofcontents.**

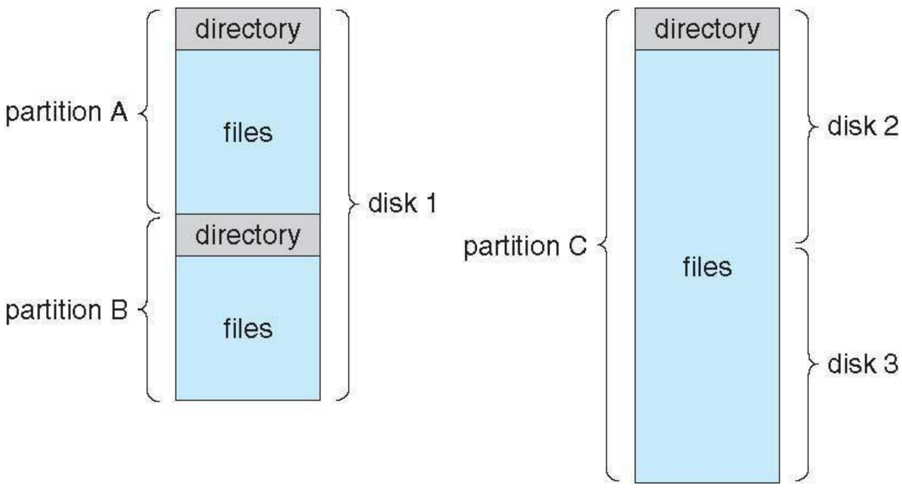


Figure:A Typical File-systemOrganization

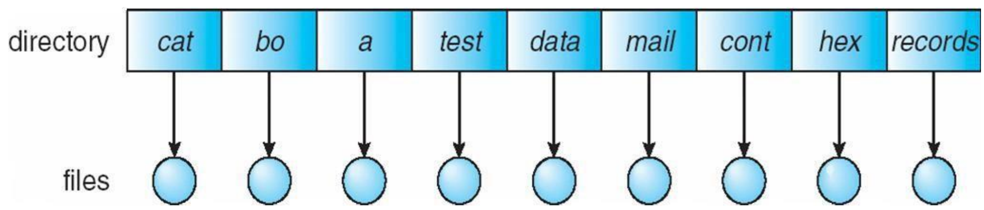
DirectoryOverview

- The directory can be viewed as a symbol table that translates file names into theirdirectory entries. A directory contains information about the files including attributeslocationandownership.Toconsideraparticulardirectorystructure,certainoperation sonthedirectoryhavetobeconsidered:
  - **Search for a file**: Directory structure is searched for a particular file in directory.Files have symbolic names and similar names may indicate a relationship betweenfiles. Using this similarity it will be easy to find all whose name matches a particularpattern.
  - **Createafile**:Newfilesneededtobecreated andaddedtothedirectory.
  - **Delete a file:**When a file is no longer needed, then it is able to remove it fromthedirectory.
  - **List a directory**: It is able to list the files in a directory and the contents of thedirectoryentryforeachfileinthelist.
  - **Rename a file**: Because the name of a file represents its contents to its users, It ispossibletochangethenamewhenthecontentsoruseofthefilechanges.Renamingafilema yalsoallowitsposition withinthedirectorystructuretobechanged.
  - **Traverse the file system**: User may wish to access every directory and every filewithin a directory structure. To provide reliability the contents and structure of theentirefile systemis saved at regularintervals.

- Themostcommon schemesfordefiningthelogicalstructureofadirectoryaredescribedbelow
  1. Single-levelDirectory
  2. Two-Level Directory
  3. Tree-StructuredDirectories
  4. Acyclic-GraphDirectories
  5. GeneralGraph Directory

1. Single-levelDirectory

- Thesimplestdirectorystructureisthesingle-leveldirectory.Allfilesarecontainedinthesamedirectory,whichiseasytosupportandundersta nd

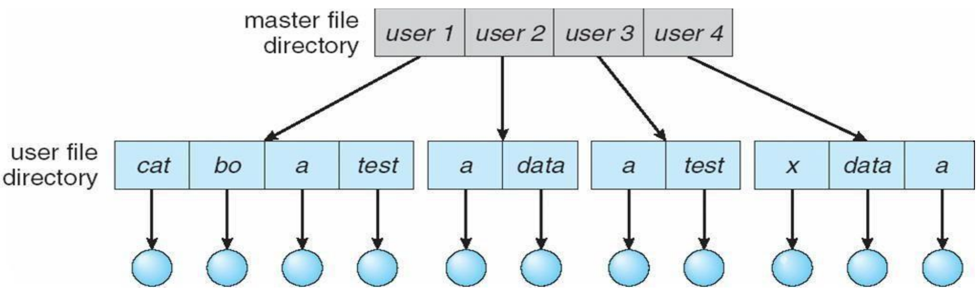


- Asingle-leveldirectoryhassignificantlimitations,whenthenumberoffilesincreasesorwhenthesystemh asmorethanoneuser.

- As directory structure is single, uniqueness of file name has to be maintained, which isdifficultwhentherearemultiple users.
- Even a single user on a single-level directory may find it difficult to remember the namesofallthefilesasthenumberoffilesincreases.
- It is not uncommon for a user to have hundreds of files on one computer system and anequal number of additional files on another system. Keeping track of so many files is adauntingtask.

2. Two-LevelDirectory

- In the two-level directory structure, each user has its own **user file directory** (UFD).TheUFDshavesimilarstructures,buteachlistsonlythefilesofasingle user.
- When a user refers to a particular file, only his own UFD is searched. Different usersmay have files with the same name, as long as all the file names within each UFD areunique.
- To create a file for a user, the operating system searches only that user's UFD toascertain whether another file of that name exists. To delete a file, the operating systemconfinesitssearchtothelocalUFDthus;itcannotaccidentallydeleteanotheruser'sfilet hathasthesame name.
- When a user job starts or a user logs in, the system's Master file directory (MFD) issearched. The MFD is indexed by user name or account number, and each entry pointstotheUFDforthatuser.

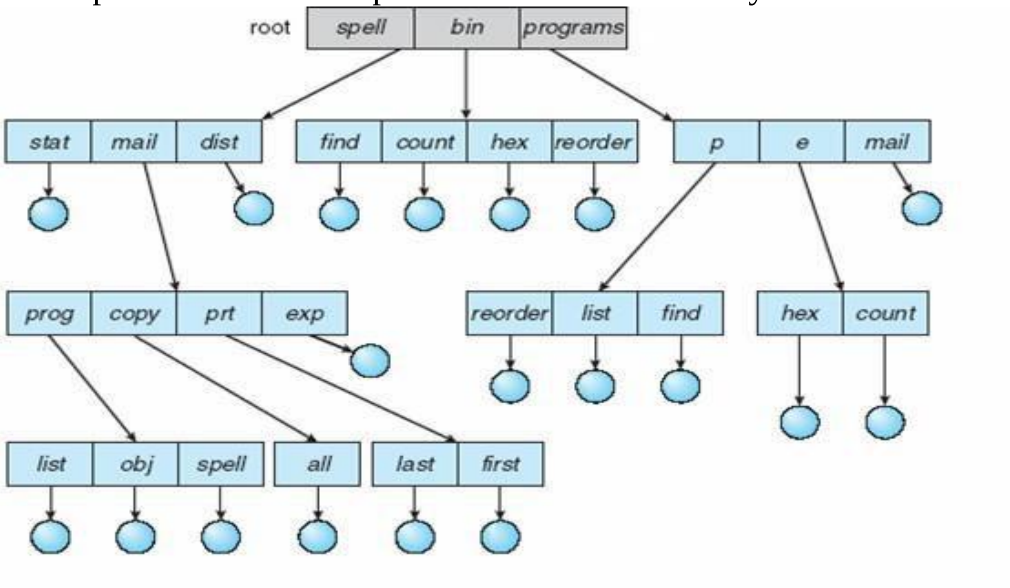


- **Advantage:**
  - Nofilename-collision amongdifferentusers.
  - Efficientsearching.
- **Disadvantage**
  - Usersareisolatedfromoneanotherandcan'tcooperateonthesametask.

3. TreeStructuredDirectories

- Atreeisthemostcommondirectorystructure.
- Thetreehasarootdirectory,andeveryfileinthesystemhasauniquepathname.
- Adirectorycontainsasetoffilesorsubdirectories.Adirectoryissimplyanotherfile,butitistreated inaspecialway.
- Alldirectorieshavethesameinternalformat.Onebitineachdirectoryentrydefinestheentryasafil e(0)orasasubdirectory(1).Specialsystemcallsareusedtocreateand deletedirectories.

- **Twotypesofpath-names:**
  1. Absolutepath-name:beginsattheroot.
  2. Relativepath-name: definesapathfromthecurrentdirectory.



Howtodeletedirectory?

1. Todeletean emptydirectory:Just deletethedirectory.
2. Todeleteanon-emptydirectory:
   - First,deleteallfilesin the directory.
   - Ifanysubdirectoriesexist,thisproceduremust beappliedrecursivelytothem.

Advantage:

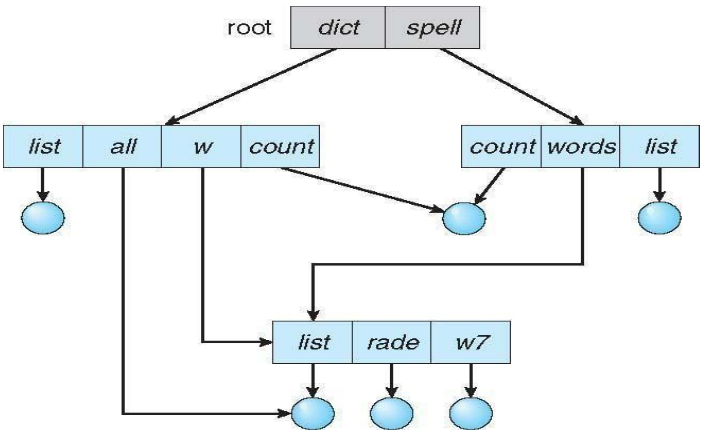
- Userscanbeallowedtoaccessthefilesof otherusers.

Disadvantages:

- Apathtoafilecan belongerthana pathin atwo-level directory.
- Prohibitsthesharingoffiles(or directories).

4. AcyclicGraphDirectories

- The common subdirectory should be shared. A shared directory or file will exist in thefile system in two or more places at once. A tree structure prohibits the sharing of filesordirectories.
- An acyclic graph is a graph with no cycles. It allows directories to share subdirectoriesandfiles.



- The same file or subdirectory may be in two different directories. The acyclic graphisanatural generalizationofthetree-structureddirectoryscheme.

Twomethodstoimplementshared-files(orsubdirectories):

1. Createanewdirectory-entrycalledalink.Alinkisapointertoanotherfile(orsubdirectory).
2. Duplicateallinformationaboutshared-filesinbothsharingdirectories.

Twoproblems:

1. Afilemayhavemultipleabsolutepath-names.
2. Deletion mayleavedangling-pointerstothenon-existentfile.

Solutiontodeletionproblem:

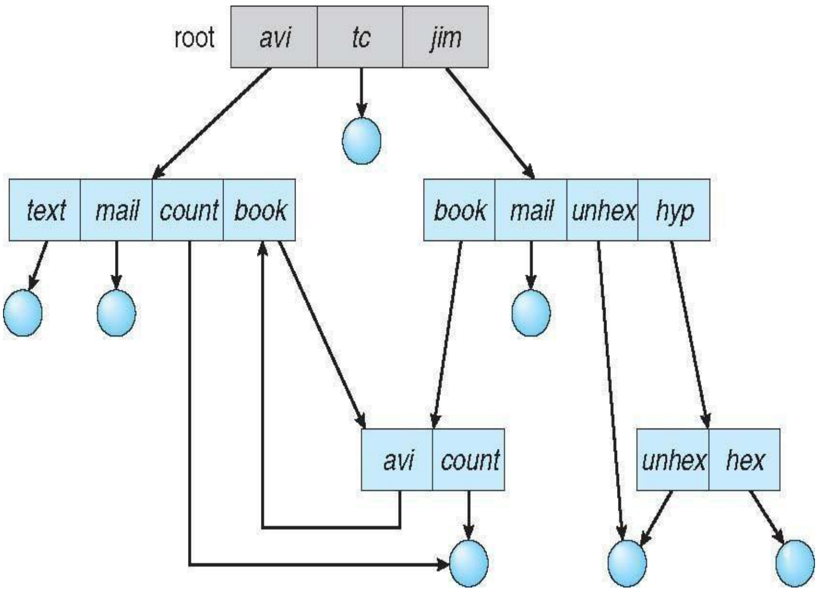
1. Useback-pointers:Preservethefileuntilallreferencestoitaredeleted.
2. Withsymboliclinks,removeonly thelink,notthefile.Ifthefileitselfisdeleted,thelinkcanberemoved.

5. GeneralGraphDirectory

- **Problem:**Iftherearecycles,wewanttoavoidsearchingcomponentstwice.
- **Solution:**Limittheno. ofdirectoriesaccessed inasearch.
- **Problem:** With cycles, the reference-count may be non-zero even when it is no longerpossible to refer to a directory (or file). (A value of 0 in the reference count means thatthereareno morereferencestothefileordirectory,andthefilecanbedeleted).
- **Solution**: Garbage-collection scheme can be used to determine when the last referencehasbeendeleted.

Garbagecollectioninvolves

- Firstpass traversesthe entirefile-systemand markseverythingthatcanbeaccessed.
- Asecondpasscollectseverythingthatisnotmarkedontoalist offree-space



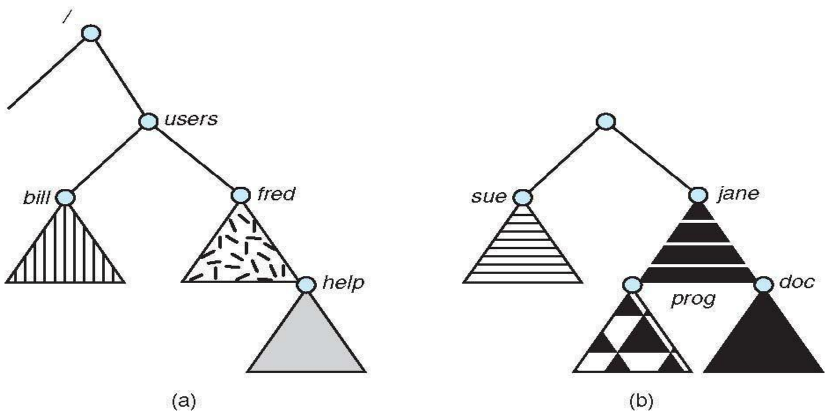
FILESYSTEMMOUNTING

- Afilemustbe*opened*beforeitisused,afilesystemmustbe*mounted*beforeitcanbeavailabletopr ocesseson the system
- **MountPoint:**Thelocationwithinthefilestructurewherethefilesystemistobeattached.

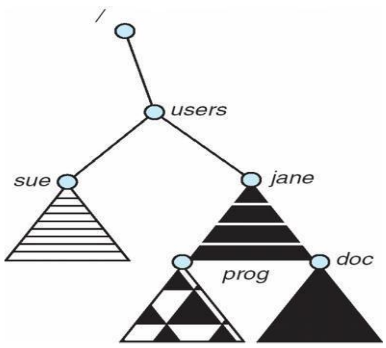
The mountingprocedure:

- Theoperatingsystemisgiventhe nameofthe deviceandthemountpoint.
- The operating system verifies that the devicecontains a valid file system. It doesso by asking the device driver to read the device directory and verifying that thedirectoryhastheexpectedformat
- The operating system notes in its directory structure that a file systemis mountedatthe specifiedmountpoint.

Toillustratefilemounting,considerthefilesystemshowninfigure.Thetrianglesrepresentsub-treesofdirectoriesthatareofinterest



- Figure(a)showsanexistingfilesystem,
- whileFigure1(b)showsanun-mountedvolumeresidingon*/device/dsk.*Atthispoint,onlythefilesontheexistingfilesystemcan beaccessed.



- Abovefigureshowstheeffectsofmountingthevolumeresidingon*/device/dsk*over */users.*
- Ifthevolumeisun-mounted,thefilesystemisrestoredtothesituationdepictedinfirstFigure.

FILESHARING

- Sharingoffileson multi-usersystemsisdesirable.
- Sharing maybedonethroughaprotectionscheme.
- Ondistributedsystems, filesmaybesharedacrossanetwork.
- NetworkFile-system(NFS)isacommondistributedfile-sharingmethod.

MultipleUsers

File-sharingcanbedonein2ways:

1. Thesystemcan allowausertoaccessthefiles ofother usersbydefaultor
2. Thesystemmayrequirethatauserspecificallygrantaccess.

Toimplementfile-sharing,thesystemmustmaintainmorefile-&directory-attributesthanonasingle-usersystem.
Mostsystemsuse conceptsoffileownerandgroup.

1. Owner

- Theuserwhomaychangeattributes&grantaccessandhasthemostcontroloverthefile(ordirectory).
- Mostsystemsimplementownerattributes bymanagingalistof user-namesanduser IDs

2. Group

- Thegroupattribute definesasubsetofuserswhocanshareaccesstothefile.
- Groupfunctionalitycanbeimplementedasasystem-widelistofgroup-namesandgroupIDs.
- Exactlywhichoperationscanbeexecutedbygroup-membersandotherusersisdefinablebythe file'sowner.
- TheownerandgroupIDsoffilesarestoredwiththeotherfile-attributesandcanbeusedtoallow/denyrequestedoperations.

RemoteFileSystems

It allows a computer to mount one or more file-systems from one or more remote-machines.Thereare threemethods:

1. Manuallytransferringfilesbetweenmachinesviaprogramslikeftp.
2. AutomaticallyDFS(Distributedfile-system):remotedirectoriesarevisiblefromalocalmachine.
3. Semi-automaticallyviawww(WorldWideWeb):Abrowserisneededtogainaccesstotheremotefiles,andseparateoperations(awrapperforftp)areusedtotransferfiles.

ftpisusedforbothanonymousandauthenticatedaccess.**Anonymousaccess**allowsausertotransferfiles withouthavinganaccountontheremotesystem.

ClientServerModel

- Allowsclientstomountremotefile-systemsfromservers.
- Themachinecontainingthefilesiscalledthe**server**.Themachineseekingaccesstothefilesisca lledthe**client**.
- Aservercanservemultipleclients,andaclientcanusemultipleservers.
- Theserverspecifieswhichresources(files)areavailabletowhichclients.
- Aclientcanbespecifiedbyanetwork-namesuchasanIPaddress.

Disadvantage:

- Clientidentificationismore difficult.
- InUNIXanditsNFS(networkfile-system),authenticationtakesplaceviatheclientnetworkinginformationbydefault.
- Oncetheremotefile-systemismounted,file-operationrequestsaresenttotheserverviatheDFSprotocol.

DistributedInformationSystems

- Providesunifiedaccesstotheinformation neededforremotecomputing.
- TheDNS(domain namesystem)provideshostname-to-networkaddresstranslations.
- Otherdistributedinfo.systemsprovideusername/passwordspacefor adistributedfacility

FailureModes

- Localfile-systemscanfailforavarietyofreasonssuchasfailureofdisk(containingthefile-system),corruptionofdirectory-structure&cable failure.
- Remotefile-systemshavemorefailuremodesbecauseofthecomplexityofnetwork-systems.
- Thenetworkcanbeinterruptedbetween2hosts.Suchinterruptionscanresultfromhardwarefailu re,poorhardwareconfiguration ornetworkingimplementationissues.
- DFSprotocolsallowdelayingoffile-systemoperationstoremote-hosts,withthehopethatthe remote-hostwillbecomeavailableagain.
- Toimplementfailure-recovery,somekindofstateinformationmaybemaintainedonboththeclientandtheserver.

ConsistencySemantics

- Theserepresentanimportantcriterionofevaluatingfile-systemsthatsupportsfile-sharing.Thesespecifyhowmultipleusersofasystemaretoaccessashared-filesimultaneously.
- In particular, they specify when modifications of data by one user will be observed byotherusers.
- Thesesemanticsare typicallyimplementedas codewiththefile-system.

- Thesearedirectlyrelatedtothe process-synchronizationalgorithms.
- AsuccessfulimplementationofcomplexsharingsemanticscanbefoundintheAndrewfile-system (AFS).

UNIXSemantics

- UNIXfile-system(UFS)usesthefollowingconsistencysemantics:
    1. Writestoanopen-filebyauserarevisibleimmediatelytootheruserswhohavethisfile opened.
    2. Onemodeofsharingallowsuserstosharethepointerofcurrentlocationintoafile.Thus,theadvancingofthepointerbyoneuseraffectsallsharing users.
- Afileisassociatedwithasinglephysicalimagethatisaccessedasanexclusiveresource.
- Contentionforthesingleimagecausesdelaysinuserprocesses.

SessionSemantics

TheAFSusesthefollowingconsistencysemantics:

1. Writestoanopenfilebyauserarenotvisibleimmediatelytootheruserssthathavethesamefileopen.
2. Onceafileisclosed,thechangesmadetoitarevisibleonlyinsessionsstartinglater.Alreadyopeninstancesofthefiledonotreflectthesechanges.

- Afilemaybeassociatedtemporarilywithseveral(possiblydifferent)imagesatthesametime.
- consequently,multipleusersareallowedtoperformbothreadandwriteaccessesconcurrentlyon their images ofthe file,withoutdelay.
- Almostnoconstraints areenforcedonscheduling accesses.

ImmutableSharedFilesSemantics

- Onceafileisdeclaredassharedbyitscreator,itcannotbemodified.
- Animmutablefilehas2keyproperties:
    1. File-name maynot bereused
    2. File-contentsmaynotbe altered.
- Thus,the nameofan immutablefilesignifiesthatthecontents of thefilearefixed.
- Theimplementationofthesesemanticsinadistributedsystemissimple,becausethesharingisdisciplined

PROTECTION

- Wheninformationisstoredinacomputersystem,wewanttokeepitsafefromphysicaldamage(reliability)andimproperaccess(protection).
- Reliabilityisgenerallyprovidedbyduplicate copiesoffiles.
- Forasmallsingle-usersystem,wemightprovideprotectionbyphysicallyremovingthefloppydisksandlockingthe minadeskdrawer.
- Fileowner/creatorshouldbeabletocontrolwhatcanbedoneandby whom.

TypesofAccess

- Systemsthat donot permit accessto the filesofotherusers do not need protection.Thisistooextreme,socontrolled-access isneeded.
- Followingoperationsmaybecontrolled:
  1. *Read:*Readfromthe file.
  2. *Write:* Writeorrewritethefile.
  3. *Execute:*Loadthefileinto memoryandexecuteit.
  4. *Append*: Writenew informationattheend ofthe file.
  5. *Delete:*Deletethefile andtreeitsspaceforpossiblereuse.
  6. *List:*Listthenameandattributes ofthefile.

AccessControl

- Commonapproachtoprotectionproblemisto makeaccessdependent onidentityofuser.
- FilescanbeassociatedwithanACL(access-controllist)whichspecifiesusernameandtypesofaccessforeachuser.

Problems:

1. Constructingalistcanbetedious.
2. Directory-entrynowneedstobeofvariable-size,resultinginmorecomplicatedspacemanagement.

Solution:

- TheseproblemscanberesolvedbycombiningACLswithan‘owner,group,universe’accesscontrol scheme
- Toreducethelength oftheACL, manysystems recognize3classificationsof users:
  1. *Owner:*Theuserwho createdthefileistheowner.
  2. *Group:*Asetofuserswhoaresharingthefileandneedsimilaraccessisagroup.
  3. *Universe:*Allother usersinthesystemconstitutetheuniverse.

OtherProtectionApproaches

- Apasswordcanbeassociatedwitheachfile.
- **Disadvantages:**
    1. Theno.ofpasswordsyouneedtoremembermaybecomelarge.
    2. If only one password is used for all the files, then all files are accessible if it isdiscovered.
    3. Commonly,onlyonepasswordisassociatedwithalloftheuser'sfiles,soprotectionis all-ornothing.

- Inamultileveldirectory-structure,weneedtoprovideamechanismfordirectoryprotection.
- Thedirectoryoperationsthatmust beprotectedaredifferentfromtheFile-operations:
    1. Controlcreation&deletionoffilesinadirectory.
    2. Controlwhethera user candeterminetheexistenceof afileinadirectory.

IMPLEMENTATIONOFFILESYSTEM

FILESYSTEMSTRUCTURE

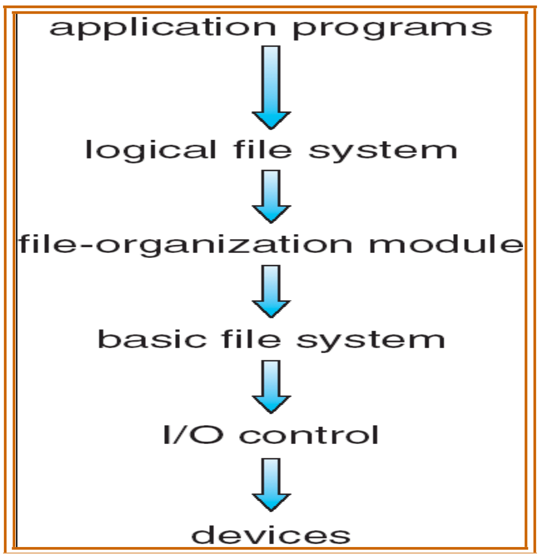
- Disksprovidethe bulk of secondary-storage onwhichafile-systemis maintained.

Thediskis asuitable mediumforstoring multiplefiles.

- Thisisbecauseof two characteristics
    1. A disk can be rewritten in place; it is possible to read a block from the disk, modifytheblock,andwriteitbackintothesameplace.
    2. A disk can access directly any block of information it contains. Thus, it is simple toaccessany fileeithersequentially orrandomly,andswitchingfromonefiletoanotherrequiresonlymovingtheread-writeheadsandwaitingforthedisktorotate.
- To improve I/O efficiency, I/O transfers between memory and disk are performed in unitsof blocks. Each block has one or more sectors. Depending on the disk drive, sector-sizevariesfrom32bytesto 4096bytes.Theusualsizeis512bytes.
- File-systems provide efficient and convenient access to the disk by allowing data to bestored,located,andretrieved easily
- Designproblemsoffile-systems:
    1. Defininghowthefile-systemshouldlooktotheuser.
    2. Creatingalgorithms&data-structurestomapthelogicalfile-systemontothephysicalsecondary-storagedevices.

LayeredFileSystems:

- The file-system itself is generally composed of many different levels. Every level indesignusesfeaturesoflowerlevelstocreatenewfeaturesforusebyhigherlevels.



- File system provide efficient and convenient access to the disk by allowing data to bestored,located,andretrievedeasily.

Afilesystemposestwoquitedifferentdesignproblems.

1. The first problem is defining how the file system should look to the user. This taskinvolves defining a file and its attributes, the operations allowed on a file,and thedirectorystructurefororganizingfiles.
2. The second problem is creating algorithms and data structures to map the logical filesystemontothephysicalsecondary-storagedevices.

The file system itself is generally composed of many different levels. The structure shown inFigure is an example of a layered design. Each level in the design uses the features of lowerlevelstocreatenewfeaturesforusebyhigherlevels.

- The lowest level, **the *I/O control***, consists of **device drivers** and interrupts handlers totransferinformationbetweenthemain memoryandthedisksystem.

A device driver can be thought of as a translator. Its input consists of high-level commands suchas"retrieveblock123."
Its output consists of low level, hardware-specific instructions that are used by the hardwarecontroller, which interfaces theI/0device to the rest ofthe system.

- ThedevicedriverusuallywritesspecificbitpatternstospeciallocationsintheI/0controller's memory to tell the controller which device location to act on and what actionstotake.
- The **basic file system** needs only to issue generic commands to the appropriate devicedriver to read and write physical blocks on the disk. Each physical block is identified byitsnumericdiskaddress(forexample,drive1,cylinder73,track2,sector10).

This layer also manages the memory buffers and caches that hold various file-system, directory,anddata blocks.
Ablockinthebufferisallocatedbeforethetransferofadiskblockcanoccur.Whenthebufferis full, the buffer manager must find more buffer memory or free up buffer space to allow arequestedI/Otocomplete.

Fileorganization

- Module knows about files and their logical blocks, as well as physical blocks. By knowingthe type of file allocation used and the location of the file, the file- organization modulecan translate logical block addresses to physical block addresses for the basicfile systemtotransfer.
- Each file's logical blocks are numbered from 0 (or 1) through *N*. Since the physical blockscontaining the data usually do not match the logical numbers, a translation is needed tolocateeachblock.
- Thefile-organizationmodulealsoincludesthefree-spacemanager,whichtracksunallocatedblocksandprovidestheseblockstothefile-organizationmodulewhenrequested.

Logicalfilesystem

- Manages metadata information. Metadata includes all of the file- system structure exceptthe actual *data* (or contents of the files). The logical file system manages the directorystructure to provide the file organization module with the information the latter needs,givenasymbolicfilename.Itmaintainsfilestructurevia**file-controlblocks(FCB)**.
- FCB contains information about the file, including ownership, permissions, and location ofthefilecontents.

FileSystemImplementation

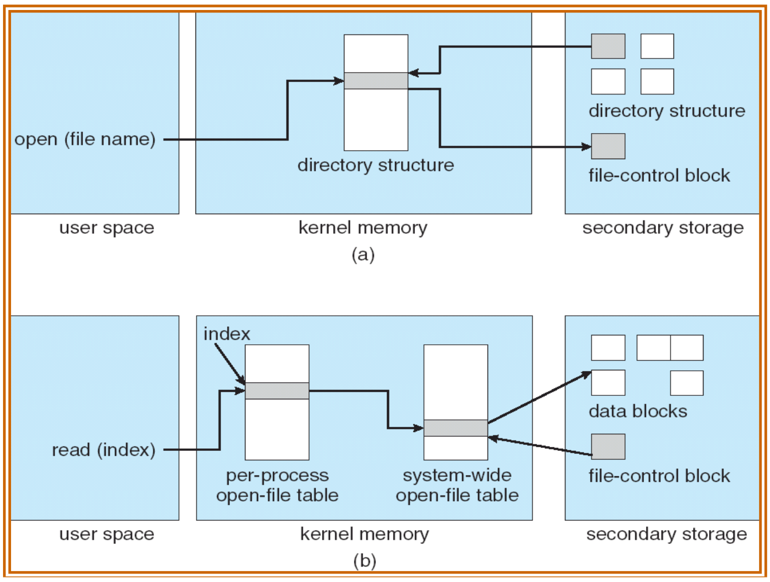
On disk, the file system may contain information about how to boot an operating system storedthere, the total number of blocks, the number and location of free blocks, the directory structure,andindividual files.

- **Boot Control Block**: On disk, the file system may contain information about how to bootan operating system stored there, the total number of blocks, the number and location offreeblocks,thedirectorystructure,andindividualfiles.
- **Volume Control Block** : (per volume) contains volume (or partition) details, such as thenumber of blocks in the partition, the size of the blocks, a free-block count and free-blockpointersandafree-FCBcountandFCBpointers.
- **Adirectory structure**(perfilesystem)is usedtoorganizethefiles.
- **A per-file FCB** contains many details about the file. It has a unique identifier number toallowassociationwithadirectoryentry.

Thein-memoryinformationisusedforbothfile-systemmanagementandperformanceimprovementviacaching.Thedataareloadedatmounttime,updatedduringfile-systemoperations,anddiscardedatdismount.Severaltypesofstructuresmaybeincluded.

- Anin-memorymounttablecontainsinformationabouteachmountedvolume.
- Anin-memorydirectory-structurecacheholdsthedirectoryinformationofrecentlyaccesseddirectories.

- ThesystemwideopenfiletablecontainsacopyoftheFCBofeachopenfile,aswellasotherinformation.
- Theper-processopenfiletablecontainsapointertotheappropriateentry inthesystem-wideopen-filetable,aswellasotherinformation.



- Buffersholdfile-systemblocks whentheyarebeingreadfromdiskorwrittentodisk.

Stepsforcreatingafile:

1) Anapplicationprogramcallsthelogicalfilesystem,whichknowstheformatofthedirectory structures
2) Thelogicalfilesystemallocatesanewfilecontrolblock(FCB)
   - IfallFCBsarecreatedatfile-systemcreationtime,anFCBisallocatedfromthefreelist
3) Thelogicalfilesystem then
   - Readstheappropriatedirectoryintomemory
   - Updatesthe directorywiththenewfilename andFCB
   - Writesthedirectorybacktothedisk

UNIX treats a directory exactly the same as a file by means of a type field in the i nodeWindows NT implements separate system calls for files and directories and treats directories asentitiesseparate fromfiles.

Stepsforopeningafile:

1) Thefunctionfirstsearchesthesystem-wideopen-filetabletoseeifthefileisalreadyinusebyanotherprocess
   - Ifitis,aper-processopen-filetableentryiscreatedpointingtotheexistingsystem-wideopen-filetable
   - Thisalgorithmcanhavesubstantialoverhead;consequently,partsofthedirectorystructureareusuallycachedinmemorytospeed operations

2) Oncethefileisfound,theFCBis copiedintoa system-wideopen-file tableinmemory
   - Thistablealsotracksthenumberof processesthathavethefileopen
3) Next,anentryismadeintheper-processopen-filetable,withapointertotheentryinthesystem-wideopen-filetable
4) Thefunctionthenreturnsapointer/indextotheappropriateentryintheper-processfile-systemtable
   - Allsubsequentfileoperationsarethenperformedviathis pointer
   - UNIXreferstothispointerasthe filedescriptor
   - Windowsreferstoitasthefilehandle Stepsforclosingafile:
1) Theper-processtableentryisremoved
2) Thesystem-wideentry'sopencountisdecremented
3) Whenallprocessesthathaveopenedthefileeventuallycloseit

Any updated metadata is copied back to the disk-based directory structure. The system-wide open-filetableentryisremoved

file permissions
file dates (create, access, write)
file owner, group, ACL
file size
file data blocks or pointers to file data blocks

PartitionsandMounting

- Each partition can be either "raw," containing no file system, or"cooked," containing afilesystem.
- Rawdiskisused wherenofilesystemisappropriate.
- UNIX swap space can use a raw partition, for example, as it uses its own format on diskanddoesnot usea filesystem.
- Boot information can be stored in a separate partition. Again, it has its own format,because at boot time the system does not have the file-system code loaded and thereforecannotinterpret thefile-systemformat.
- Bootinformationisasequentialseriesofblocks,loadedasanimageintomemory.
- Execution of the image starts at a predefined location, such as the first byte. This bootloader in turn knows about the file-system structure to be able to find and load the kernelandstartitexecuting.
- Therootpartitionwhichcontainstheoperating-systemkernelandsometimesothersystemfiles,ismountedatboottime.

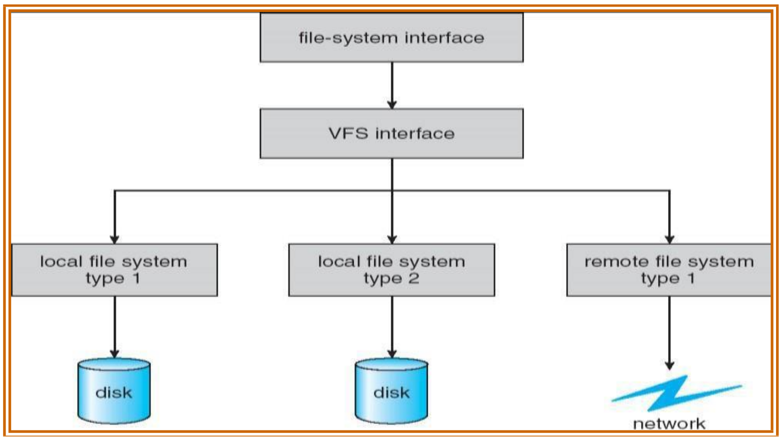
- Othervolumescanbeautomaticallymountedatbootormanuallymountedlater,dependingonthe operatingsystem.
- Aftersuccessfulmountoperation,theoperatingsystemverifiesthatthedevicecontainsavalidfile system.
- Itisdonebyaskingthedevicedrivertoreadthedevicedirectoryandverifyingthatthedirectoryhasth eexpectedformat.
- Iftheformatisinvalid,thepartitionmusthaveitsconsistency checkedandpossiblycorrected,eitherwithorwithoutuserintervention.
- Finally,theoperatingsystemnotesinitsin-memorymounttablethatafilesystemismounted,alongwiththetypeofthefilesystem.
- The<u>rootpartition</u>is mountedatboot time
  - Itcontainsthe operating-systemkernelandpossiblyothersystemfiles
- <u>Othervolumes</u>canbeautomaticallymountedatboottimeormanuallymountedlater
- Aspartofasuccessfulmountoperation,theoperatingsystem<u>verifies</u>thatthestoragedevicecontainsa <u>validfile system</u>
  - It asksthedevicedrivertoreadthedevicedirectoryandverifythatthedirectoryhastheexpected format
  - Iftheformatisinvalid,thepartitionmusthaveits<u>consistencychecked</u>andpossiblycorrected
  - Finally,theoperatingsystemnotesinitsin-memory mounttablestructurethata<u>filesystemismounted</u>alongwiththetypeofthefilesystem

VirtualfileSystems

Thefile-systemimplementationconsistsofthreemajorlayers,asdepictedschematicallyinFigure. The first layer is the file-system interface, based on the open(), read(), write(), and close()callsandonfiledescriptors.

The second layer is called the virtual file system (vfs) layer. The VFS layer serves two importantfunctions:

- It separates file-system-generic operations from their implementation by defining a cleanVFS interface. Several implementations for the VFS interface may coexist on the samemachine,allowingtransparentaccesstodifferenttypesoffilesystems mountedlocally.
- It provides a mechanism for uniquely representing a file throughout a network. The VFS isbased on a file-representation structure, called a vnode that contains a numerical designatorfor a network-wide unique file. This network-wide uniqueness is required for support ofnetworkfilesystems.
- Thekernelmaintainsonevnodestructurefor eachactivenode.
- Thus,theVFSdistinguisheslocalfilesfromremoteones,andlocalfilesarefurtherdistinguishedaccordingtotheirfile-systemtypes.



DirectoryImplementation

Selectionofdirectoryallocationanddirectorymanagementalgorithmssignificantlyaffectstheefficiency, performance,andreliabilityofthefilesystem

OneApproach:Directindexingofa linearlist

- Consistsofalistoffilenameswithpointerstothedata blocks
- Simpletoprogram
- Time-consumingtosearchbecauseitisalinearsearch.
- Sortingthelistallowsforabinarysearch;however,thismaycomplicatecreatinganddeletingfiles
- Tocreateanewfile,wemustfirstsearchthedirectorytobesurethatnoexistingfilehasthesamename.
- Addanew entryattheendofthe directory.
- Todeleteafile,wesearchthedirectoryforthenamedfileandthenreleasethespaceallocatedtoit.
- Toreusethedirectoryentry,wecandooneofseveralthings.Marktheentryasunused.
- Analternativeistocopythelastentryinthedirectoryintothefreedlocationandtodecreasethelength ofthedirectory.
- Directoryinformationisusedfrequently,and userswillnoticeifaccesstoitis slow.

AnotherApproach:Listindexingviaahashfunction

- Takesavaluecomputedfromthefilenameandreturnsapointertothefilenameinthelinearlist
- Greatlyreducesthedirectorysearchtime
- **Canresultincollisions**–situationswheretwofilenameshashtothesamelocation
- Ahashtableareitsgenerallyfixedsizeandthedependenceofthehashfunctiononthatsize.(i.e.,fixed numberofentries).
- Eachhashentrycanbealinkedlistinsteadofanindividualvalue,andwecanresolvecollisions byadding the new entryto thelinkedlist.

ALLOCATIONMETHODS

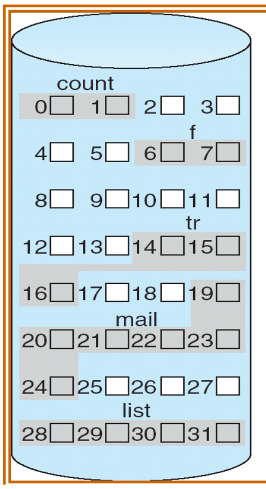
Allocation methods address the problem of allocating space to files so that disk space is utilizedeffectivelyandfilescanbeaccessedquickly.

Three methodsexistforallocatingdiskspace

- **Contiguousallocation**
- **Linkedallocation**
- **Indexedallocation**

Contiguousallocation:

- Requiresthateachfile occupyasetofcontiguousblocks onthedisk
- Accessing a file is easy – only need the starting location (block #) and length (number ofblocks)
- Contiguous allocation of a file is defined by the disk address and length (in block units) ofthe first block. If the file is *n* blocks long and starts at location *b*, then it occupies blocks *b*,*b* + 1, *b* + 2, ... ,*b* + *n* - 1. The directory entry for each file indicates the address of thestartingblockandthelengthofthearea allocatedforthisfile.
- Accessingafilethathasbeenallocatedcontiguouslyiseasy.Forsequentialaccess,thefile system remembers the disk address of the last block referenced and when necessary,reads the next block. For direct access to block *i* of a file that starts at block *b*, we canimmediately access block *b* + *i*. Thus, both sequential and direct access can be supportedbycontiguousallocation.



file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Disadvantages:

1. Finding space for a new file is difficult. The system chosen to manage free spacedetermines how this task is accomplished. Any management system can be used, butsomeareslowerthanothers.
2. Satisfying a request of size *n* from a list of free holes is a problem. First fit and bestfitarethemostcommonstrategiesusedtoselectafreehole fromthesetofavailable

holes.

3. Theabovealgorithmssufferfromtheproblem ofexternalfragmentation.
   - Asfilesareallocatedanddeleted,thefreediskspaceisbrokenintopieces.
   - Externalfragmentation existswheneverfreespaceisbrokenintochunks.
   - It becomes a problem when the largest contiguous chunk is insufficient for arequest; storage is fragmented into a number of holes, none of which is largeenoughtostore thedata.
   - Dependingonthetotalamountofdiskstorageandtheaveragefilesize,externalfragmentationmaybeaminor oramajorproblem.

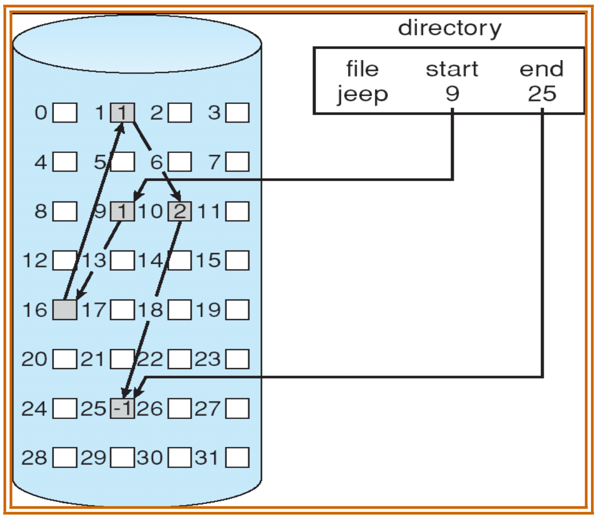
LinkedAllocation:

- Solvestheproblemsofcontiguous allocation
- Eachfileisalinked list ofdiskblocks:blocksmaybescatteredanywhereonthedisk
- Thedirectorycontainsapointertothefirstandlastblocksofafile
- Creatinganewfile requiresonlycreationofanew entryinthedirectory
- Writingto afilecausesthefree-spacemanagementsystemtofindafreeblock

- ➢ Thisnewblockis writtentoandislinked totheendofthefile
- ➢ Readingfromafilerequiresonlyreadingblocksbyfollowingthepointersfromblocktoblock.

Advantages

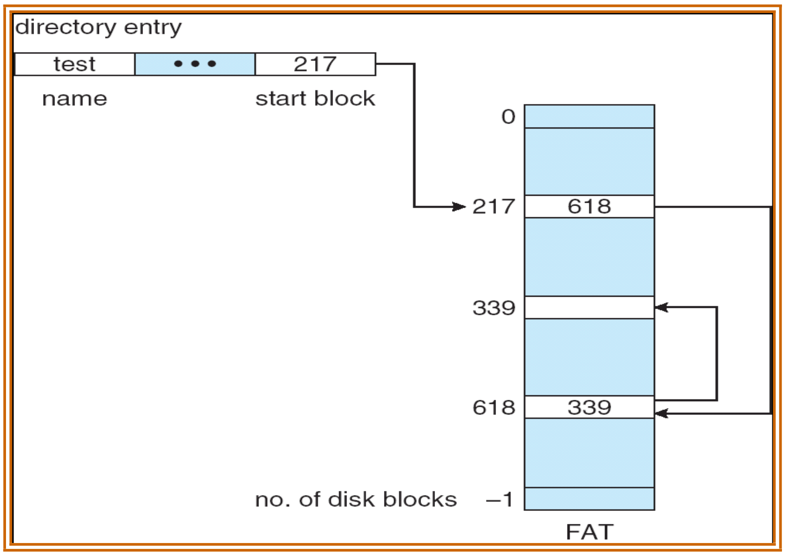
- Thereisno externalfragmentation
- Anyfreeblocksonthe freelistcanbeusedtosatisfyarequestfordiskspace
- Thesizeofafileneed notbedeclaredwhenthefileiscreated
- Afilecancontinuetogrowaslongasfreeblocksareavailable
- Itisnevernecessarytocompactdiskspaceforthesakeoflinked allocation(however,fileaccessefficiencymayrequireit)



- Each file is a linked list of disk blocks; the disk blocks may be scattered anywhere on thedisk.Thedirectorycontainsapointertothefirstandlastblocksofthefile.
- For example, a file of five blocks might start at block 9 and continue at block 16, thenblock 1, then block 10, and finally block 25. Each block contains a pointer to the nextblock. These pointers are not made available to the user. A disk address (the pointer)requires4bytesinthedisk.
- To**create**anewfile,wesimplycreateanewentryilethedirectory.Withlinkedallocation, each directory entry has a pointer to the first disk block of the file. This pointeris initialized to *nil* (the end-of-list pointer value) to signify an empty file. The size field isalsosetto0.
- A **write** to the file causes the free-space management system to filed a free block, and thisnewblockiswrittentoandislinkedtothe endofthefile.
- To **read** a file, we simply read blocks by following the pointersfromblock to block.There is no external fragmentation with linked allocation, and any free block on the free-spacelistcanbeusedtosatisfyarequest.Thesizeofafileneednotbedeclaredwhenthatfileiscreate d.
- A file can continue to grow as long as free blocks are available. Consequently, it is nevernecessarytocompactdiskspace.
- **Disadvantages**:
    1. The major problem is that it can be used effectively only for sequential-accessfiles. To filed the i th block of a file, we must start at the beginning of that fileandfollowthepointersuntilwegettotheithblock.
    2. Space required for the pointers. Solution is clusters. Collect blocks into multiplesandallocateclustersrather thanblocks
    3. Reliability - the files are linked together by pointers scattered all over the diskandifapointerwerelostordamagedthenallthelinksarelost.

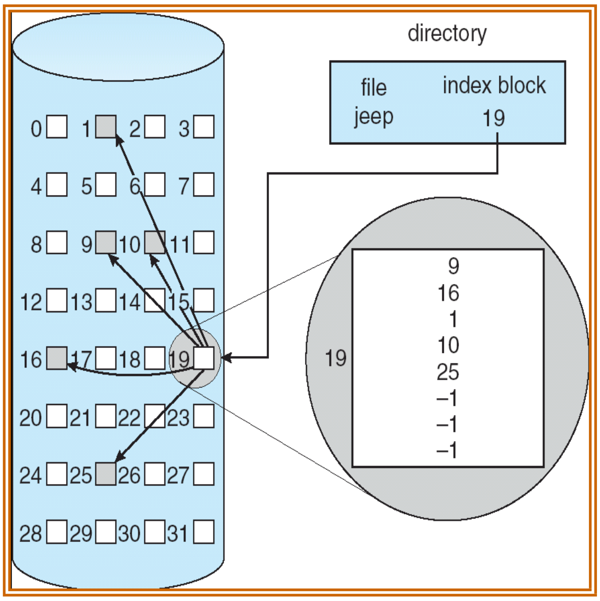
FileAllocationTable:

- Asectionofdiskatthebeginningofeachvolumeissetasidetocontainthetable.Thetablehasoneent ryforeachdiskblockandisindexedbyblocknumber.
- TheFATisusedinmuchthesamewayasalinkedlist.Thedirectoryentrycontainstheblocknumbe rofthefirstblockofthefile.
- Thetableentryindexedbythat blocknumbercontainstheblock numberofthe next blockinthefile.
- Thechaincontinuesuntilitreachesthelastblock,whichhasaspecialend-of-filevalueasthetableentry.
- Anunusedblockisindicatedbyatablevalueof 0.
- ConsideraFATwitha fileconsistingof diskblocks217, 618,and339.



Indexedallocation:

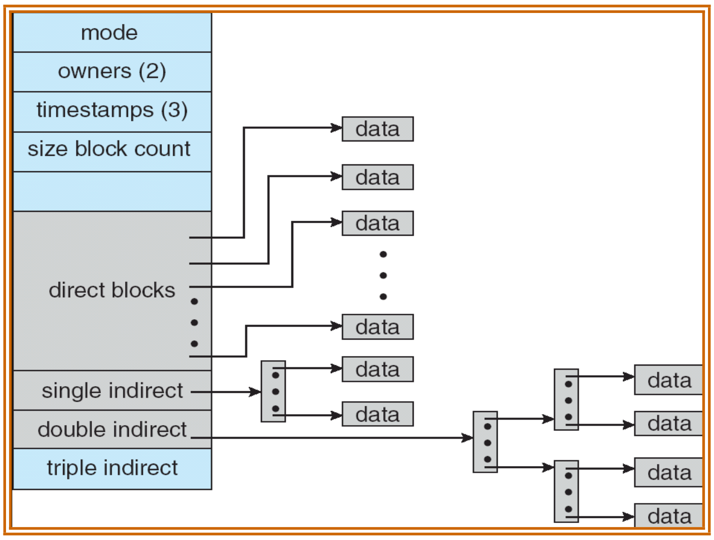
- Bringsallthepointerstogetherintoonelocationcalledindexblock.
- Eachfilehasitsownindexblock,whichisanarrayofdisk-blockaddresses.
- The *ith* entry in the index block points to the*ith* block of the file. The directory containsthe address of the index block. To find and read the *ith* block, we use the pointer in the *ith*index-block entry.
- When the file is created, all pointers in the index block are set to *nil*. When the ith block isfirstwritten,ablockisobtainedfromthefree-spacemanageranditsaddressisputintheithindex-block entry.
- Indexed allocation supports direct access, without suffering from external fragmentation,becauseanyfreeblockonthediskcansatisfyarequestformorespace.
- Disadvantages:
  - Suffersfromsomeofthesameperformanceproblemsas linkedallocation
  - Indexblockscanbecachedinmemory;however,datablocksmaybespreadalloverthediskvolume.
  - Indexedallocationdoessufferfromwastedspace.
  - Thepointeroverheadoftheindexblockisgenerallygreaterthanthepointeroverheadoflinkedallocation.



If the index block is too small, however, it will not be able to hold enough pointers for a large file,and a mechanism will have to be available to deal with this issue. Mechanisms for this purposeincludethe following:

a) **Linked scheme.** An index block is normally one disk block. Thus, it can be read and writtendirectly byitself.Toallowforlargefiles,wecanlinktogetherseveralindexblocks.Forexample, an index block might contain a small header giving the name of the file and a set of thefirst 100 disk-block addresses. The next address (the last word in the index block) is *nil* (for asmallfile)orisapointertoanotherindexblock(foralargefile).

b) **Multilevel index.** A variant of linked representation uses a first-level index block to point to aset of second-level index blocks, which in turn point to the file blocks. To access a block, theoperating system uses the first-level index to find a second-level index block and then uses thatblock to find the desired data block. This approach could be continued to a third or fourth level,dependingonthedesiredmaximumfilesize



c) **Combined scheme.** For eg. 15 pointers of the index block is maintained in the file's i node.The first 12 of these pointers point to **direct blocks**; that is, they contain addresses of blocks thatcontain data of the file. Thus, the data for small files (of no more than 12 blocks) do not need aseparate index block.If the block size is 4 KB,then up to 48KB of data can be accesseddirectly. The next three pointers point to indirect blocks. The first points to a **single indirectblock**, which is an index block containing not data but the addresses of blocks that do containdata. The second points to a **double indirect block**, which contains the address of a block thatcontains the addresses of blocks that contain pointers to the actual data blocks. The last pointercontainstheaddress ofa **tripleindirectblock**.

Performance

- Contiguousallocationrequiresonlyoneaccesstogetadiskblock.Sincewecaneasilykeep the initial address of the file in memory, we can calculate immediately the diskaddressofthe *ith*blockandreaditdirectly.
- For linked allocation, we can also keep the address of the next block in memory andread it directly. This method is fine for sequential access. Linked allocation shouldnotbeusedforanapplicationrequiringdirectaccess.
- Indexed allocation is more complex. If the index block is already in memory, then theaccess can bemadedirectly.However,keeping the index block inmemory requiresconsiderable space. If this memory space is not available, then we may have to read firsttheindexblockandthenthedesireddatablock.

FreeSpaceManagement

Thespacecreatedafterdeletingthefilescanbereused.Anotherimportantaspectofdiskmanagement is keeping track of free space in memory. The list which keeps track of free space inmemory is called the free-space list. To create a file, search the free-space list for the requiredamount 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. The free-space list, isimplementedindifferentwaysasexplainedbelow.

a) BitVector

- Fastalgorithmsexistforquicklyfindingcontiguousblocksofagivensize
- One simple approach is to use a ***bit vector***, in which each bit represents a disk block,setto1iffreeor0ifallocated.

Forexample,consideradiskwhereblocks2,3,4,5,8,9,10,11,12,13,17and18arefree,andtherestoftheblock sareallocated.Thefree-spacebitmapwouldbe
0011110011111100011

- Easytoimplementandalsoveryefficientinfindingthefirstfreeblockor‘n’ consecutivefreeblocks onthedisk.

- Thedown sideisthata 40GBdiskrequiresover5MBjusttostorethebitmap.

b) LinkedList

a. Alinkedlistcanalsobeusedtokeeptrack of allfreeblocks.
b. Traversing the list and/or finding a contiguous block of a given size are not easy, butfortunately are not frequently needed operations. Generally the system just adds andremovessingleblocksfromthebeginning ofthelist.
c. TheFATtablekeepstrackofthefreelistasjustone morelinked listonthetable.

c) Grouping

a. A variation on linked list free lists. It stores the addresses of n free blocks in the firstfreeblock.Thefirstn-1blocksareactuallyfree.Thelastblockcontainstheaddressesofanothernfree blocks, andsoon.
b. Theaddress of alargenumberoffreeblocks canbefoundquickly.

d) Counting

a. When therearemultiple contiguous blocks of freespacethen the system cankeeptrackofthestartingaddressofthegroupandthenumberofcontiguousfreeblocks.
b. Ratherthankeepingallistofnfreediskaddresses,wecankeeptheaddressoffirstfreeblockandthenumberoffreecontiguousblocksthatfollowthefirstblock.
c. Thustheoverallspaceisshortened.Itissimilartotheextentmethodofallocatingblocks.

e) SpaceMaps

a. Sun's ZFS file system was designed for huge numbers and sizes of files, directories,andeven file systems.
b. The resulting data structures could be inefficient if not implemented carefully.Forexample, freeing up a 1 GB file on a 1 TB file system could involve updatingthousandsof blocksof free listbitmapsif the filewasspreadacrossthedisk.
c. ZFSusesacombinationoftechniques,startingwithdividingthediskupinto(hundreds of)***Metaslabs***ofamanageablesize,eachhavingtheirownspacemap.
d. Free blocks are managed using the counting technique, but rather than write theinformation to a table, it is recorded in a log-structured transaction record. Adjacentfreeblocksarealsocoalescedintoalargersinglefreeblock.
e. Anin-memoryspacemapisconstructedusingabalancedtreedatastructure,constructedfromthe log data.
f. The combination of the in-memory tree and the on-disk log provide for very fast andefficientmanagementoftheseverylargefilesandfreeblocks.

SECONDARYSTORAGESTRUCTURES

OVERVIEWOFMASS-STORAGESTRUCTURE

Weblink-https://youtu.be/ZjMwUhapSEM

MagneticDisks

- Magneticdisksprovidethebulkofsecondarystorageformoderncomputersystems.
- Eachdiskplatterhasaflatcircularshape,likeaCD.Commonplatterdiametersrangefrom1.8to5.25inches.
- Thetwosurfacesofaplatterarecoveredwithamagneticmaterial.Theinformationstoredbyrecordingitmagneticallyontheplatters.

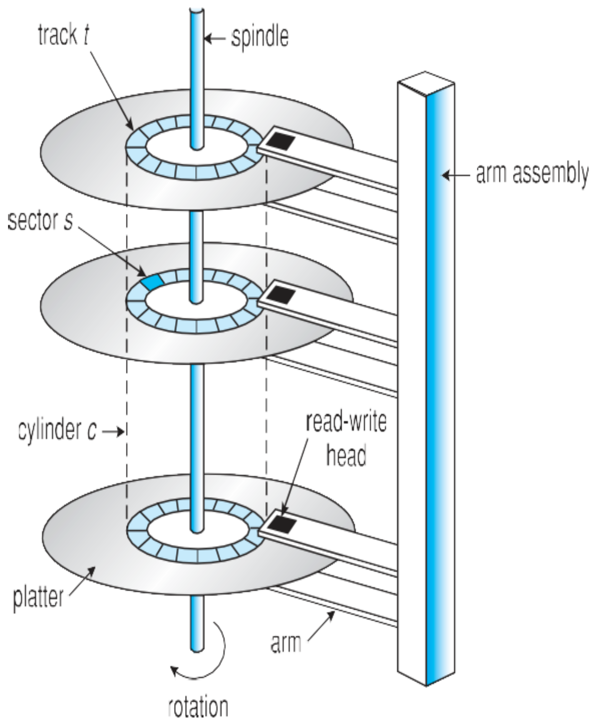


Figure:Moving-headdiskmechanism

- The surface ofa platter is logically dividedinto circular tracks,which are subdividedinto sectors. Sector is the basic unit of storage. The set of tracks that are at one armposition makesupacylinder.
- Thenumberofcylindersinthediskdriveequalsthenumberoftracksineachplatter.
- There may be thousands of concentric cylinders in a disk drive, and each track maycontainhundreds ofsectors.

- **Seek Time:-**Seek time is the time required to move the disk arm to the requiredtrack.
- **Rotational Latency (Rotational Delay):-**Rotational latency is the time taken forthedisktorotatesothattherequiredsectorcomesunderther/whead.
- **Positioning time orrandom access time**is the summation of seek time androtationaldelay.
- **Disk Bandwidth:-**Disk bandwidth is the total number of bytes transferred dividedby total timebetweenthe first request for service and the completion of lasttransfer.
- **Transferrate**istherateatwhichdataflowbetweenthedriveandthecomputer.

As the disk head flies on an extremely thin cushion of air, the head will make contact with thedisk surface. Although the disk platters are coated with a thin protective layer, sometimes theheadwilldamagethemagneticsurface.Thisaccidentiscalleda**headcrash**.

MagneticTapes

- Magnetic tape is a secondary-storage medium. It is a permanent memory and can holdlargequantitiesofdata.
- The time taken to access data (access time) is large compared with that of magnetic disk,becauseheredataisaccessedsequentially.
- When the n^{th} data has to be read, the tape starts moving from first and reaches the n^{th}position and then data is read from n^{th} position. It is not possible to directly move to thenth position. So tapesareusedmainly forbackup,forstorageofinfrequently usedinformation.
- A tape is kept in a spool and is wound or rewound past a read-write head. Moving to thecorrect spot on a tape can take minutes, but once positioned, tape drives can write data atspeedscomparabletodiskdrives.

DISKSTRUCTURE

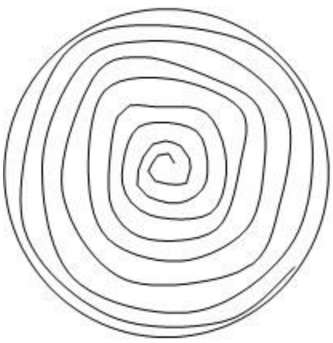
- Moderndiskdrivesareaddressedasalargeone-dimensionalarray.Theone-dimensionalarrayoflogicalblocksis mappedontothesectorsofthe disksequentially.
- Sector 0 is the first sector of the first track on the outermost cylinder. The mappingproceeds in order through that track, then through the rest of the tracks in that cylinder,andthenthroughtherestof thecylindersfromoutermosttoinnermost.

Thediskstructure(architecture)canbeoftwotypes–

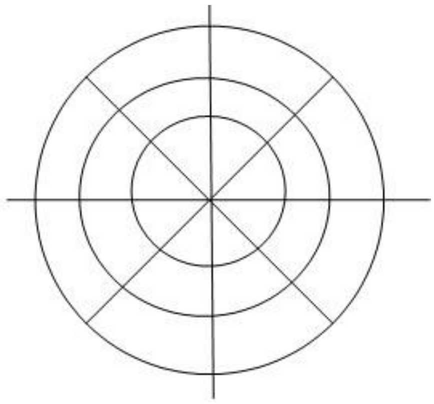
1. ConstantLinearVelocity(CLV)
2. ConstantAngularVelocity(CAV)

1. **CLV**– The density of bits per track is uniform. The farther a track is from the center ofthe disk, the greater its length, so the more sectors it can hold. As we move from outerzones to inner zones, the number of sectors per track decreases. This architecture is usedinCD-ROMandDVD-ROM.

2. **CAV**– There is same number of sectors in each track. The sectors are densely packed inthe inner tracks. The density of bits decreases from inner tracks to outer tracks to keepthedatarateconstant.



CLV



CAV

DISKATTACHMENT

Computers canaccess dataintwo ways.

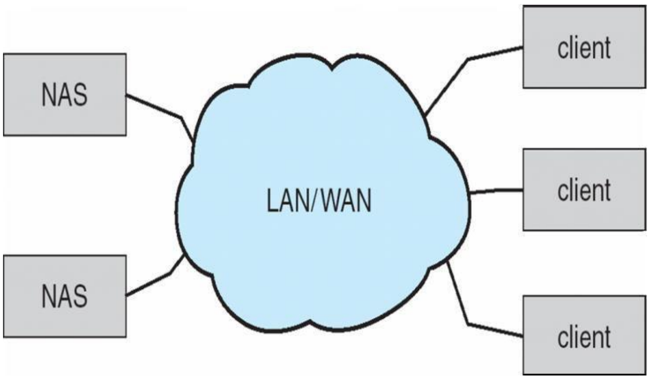
1. viaI/Oports(orhost-attachedstorage)
2. viaaremote hostin adistributedfilesystem(ornetwork-attached storage)

1. Host-AttachedStorage:

- Host-attachedstorageisstorageaccessedthroughlocalI/Oports.
- Example: the typical desktop PC uses an I/O bus architecture called IDE or ATA.ThisarchitecturesupportsamaximumoftwodrivesperI/Obus.
- The other cabling systems are – SATA (Serially Attached Technology Attachment),SCSI(SmallComputerSystemInterface)andfiberchannel(FC).
- SCSI is a bus architecture. Its physical medium is usually a ribbon cable. FC is ahigh- speed serial architecture that can operate over optical fiber or over a four-conductor copper cable. An improved version of this architecture is the basis ofstorage-areanetworks(SANs).

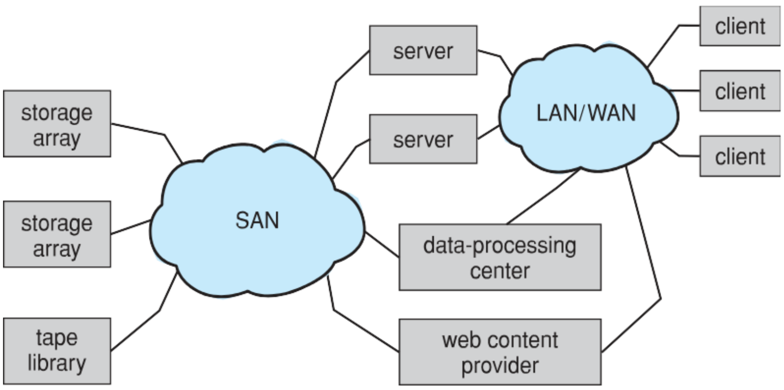
2. Network-AttachedStorage

- A network-attached storage (NAS) device is a special-purpose storage system that isaccessedremotelyovera networkas shown inthe figure.
- Clients access network-attached storage via a remote-procedure-call interface. Theremote procedure calls (RPCs) are carried via TCP or UDP over an IP networkusuallythesamelocal-area network (LAN)carries alldatatrafficto theclients.
- Network-attachedstorageprovidesaconvenientwayforallthecomputersonaLANtoshareapoolofstoragefiles.



3. StorageAreaNetwork(SAN)

- Astorage-areanetwork(SAN)isaprivatenetworkconnectingserversandstorageunits.
- ThepowerofaSANliesinitsflexibility.Multiplehostsandmultiplestoragearrayscanattach tothesameSAN,andstoragecanbedynamicallyallocatedto hosts.
- ASANswitchallowsorprohibitsaccessbetweenthehostsandthestorage.FiberChanelisthe mostcommonSANinterconnect.



DISKSCHEDULING

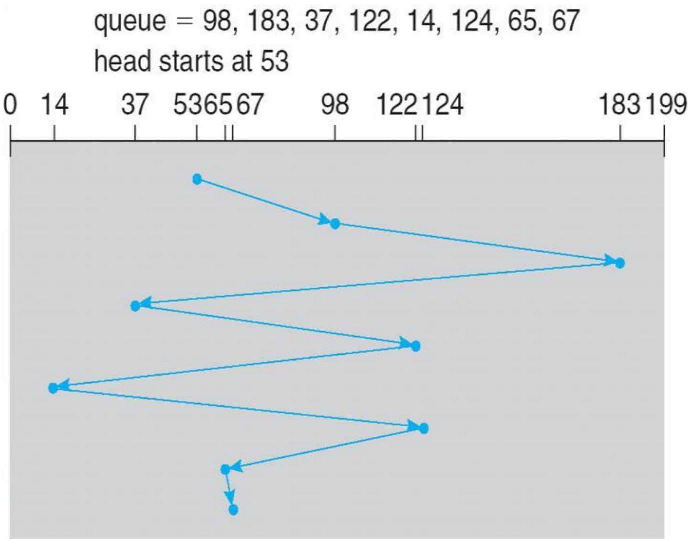
Differenttypesofdiskschedulingalgorithms areasfollows:

1. FCFS(FirstComeFirstServe)
2. SSTF(ShortestSeekTime First)
3. SCAN(Elevator)
4. C-SCAN
5. LOOK
6. C-LOOK

1. FCFSschedulingalgorithm:

This is the simplest form of disk scheduling algorithm. This services the request in the orderthey are received. This algorithm is fair but do not provide fastest service. It takes no specialcaretominimize theoverall seektime.

Eg:- consider a disk queue with request for i/o to blocks on cylinders. 98, 183, 37, 122, 14,124,65,67



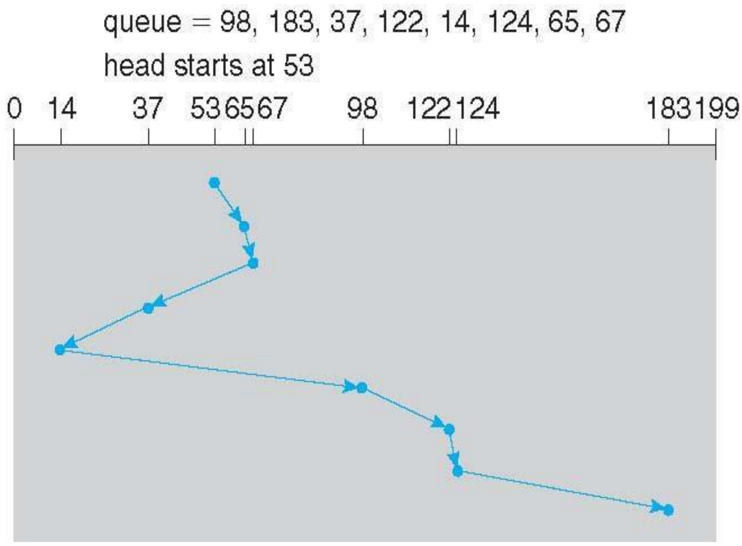
If the disk head is initially at 53, it will first move from 53 to 98 then to 183 and then to 37,122,14,124,65,67foratotalheadmovementof640cylinders.Thewildswingfrom122to14andthenbackto124illustratestheproblemwiththisschedule.

Weblink:https://youtu.be/hSaPhBtU_BA

2. SSTF(Shortest Seek TimeFirst)algorithm:

Thisselectstherequestwithminimumseektimefromthecurrentheadposition.SSTFchoosesthepending requestclosest tothe current head position.

Eg:- consider a disk queue with request for i/o to blocks on cylinders. 98, 183, 37, 122, 14,124,65,67.

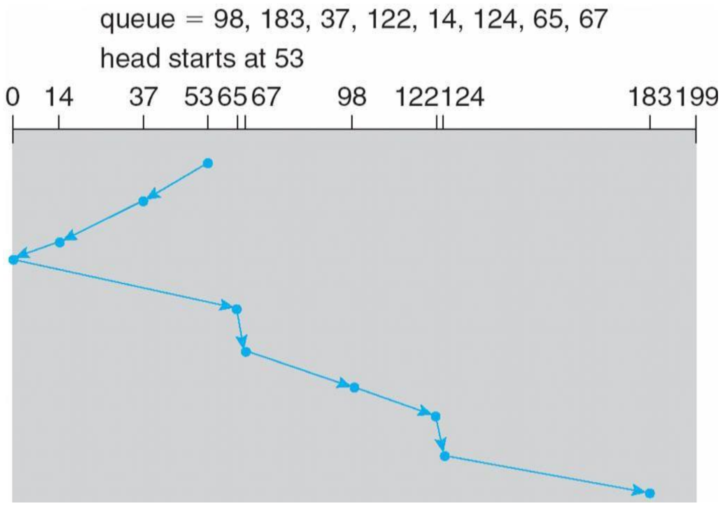


If the disk head is initially at 53, the closest is at cylinder 65, then 67, then 37 is closer than98 to 67. So it services 37, continuing we service 14, 98, 122, 124 and finally 183. The totalhead movement is only 236 cylinders. SSTF is a substantial improvement over FCFS, it isnotoptimal.

3. **<u>SCANalgorithm:</u>**

In this the disk arm starts moving towards one end, servicing the request as it reaches eachcylinder until it gets to the other end of the disk. At the other end, the direction of the headmovement is reversed and servicing continues. The initial direction is chosen dependinguponthedirectionofthehead.

Eg:- consider a disk queue with request for i/o to blocks on cylinders. 98, 183, 37, 122, 14,124,65,67



Ifthediskheadisinitiallyat53andiftheheadismovingtowardstheoutertrack,itservices65,67,98,122,124and183.Atcylinder199thearmwillreverseandwillmove

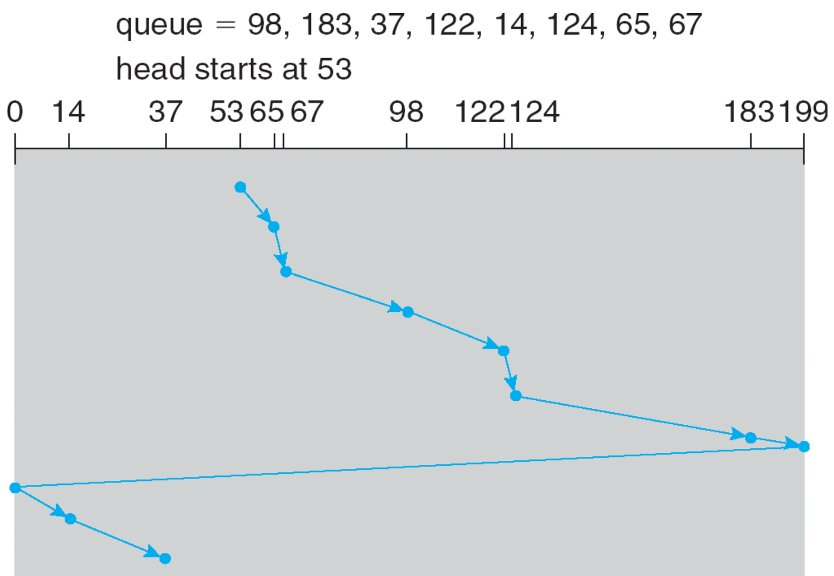
towardstheotherendofthediskservicing37andthen14.TheSCANisalsocalledaselevatoralgorithm

4. **C-SCAN(Circularscan)algorithm**:

C-SCAN isavariantof SCANdesignedtoprovideamoreuniformwaittime.

Like SCAN, C-SCANmoves the head from end of the disk to the other servicing therequest along the way. When the head reaches the other end, it immediately returns to thebeginningofthedisk,withoutservicinganyrequestonthereturn.

Eg:- consider a disk queue with request for i/o to blocks on cylinders. 98, 183, 37, 122, 14,124,65,67.



Ifthe disk head is initially at 53 and if the head ismoving towards the outer track, itservices 65, 67, 98, 122, 124 and 183. At cylinder 199 the arm will reverse and will moveimmediately towards the other end of the disk, then changes the direction of head and serves14andthen37.

Note: If the disk head is initially at 53 and if the head is moving towards track 0, it services37 and 14 first. At cylinder 0 the arm will reverse and will move immediately towards theotherendofthediskservicing65,67,98,122,124and183.

5. **LookSchedulingalgorithm:**

Look and C-Look scheduling are different version ofSCAN andC-SCAN respectively.Here the arm goes only as far as the final request in each direction. Then it reverses, withoutgoing all the way to the end of the disk. The Look and C-Look scheduling look for a requestbeforecontinuingtomoveinagivendirection.

Eg:- consider a disk queue with request for i/o to blocks on cylinders. 98, 183, 37, 122, 14,124,65,67.

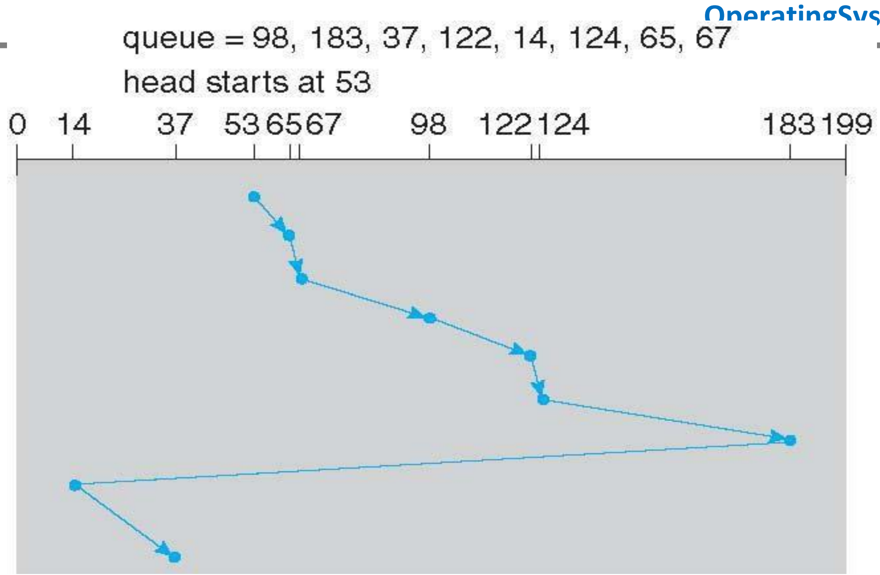


Figure:C-LOOK disk scheduling.

Ifthediskheadisinitiallyat53andiftheheadismovingtowardstheoutertrack,itservices 65, 67, 98, 122, 124 and 183. At the final request 183, the arm will reverse and willmovetowardsthefirstrequest14andthenserves37.

SELECTIONOFADISK-SCHEDULING ALGORITHM

- SSTFiscommonlyusedanditincreasesperformanceoverFCFS.
- SCANandC-SCANalgorithmisbetterforaheavyloadondisk.SCANandC-SCANhavelessstarvationproblem.
- SSTFor Lookisareasonablechoiceforadefaultalgorithm.
- Selection of disk scheduling algorithm is influenced by the file allocation method, ifcontiguous file allocation is chosen, then FCFS is best suitable, because the files arestoredincontiguousblocksandtherewill belimitedheadmovementsrequired.
- A linked or indexed file may include blocks that are widely scattered on the disk,resultingingreaterheadmovement.
- The location of directories and index blocks is also important. Since every file must beopened to be used, and opening a file requires searching the directory structure, thedirectorieswillbeaccessedfrequently.
- Suppose that a directory entry is on the first cylinder and a file's data are on the finalcylinder. The disk head has to move the entire width of the disk. If the directory entrywere on the middle cylinder, the head would have to move, at most, one-half the width.Caching the directories and index blocks in main memory can also help to reduce thedisk-armmovement,particularlyforreadrequests.

DISKMANAGEMENT

DiskFormatting

- The process of dividing the disk into sectors and filling the disk with a special datastructure is called low-level formatting. Sector is the smallest unit of area that is read /written by the disk controller. The data structure for a sector typically consists of aheader, a data area (usually 512 bytes in size) and a trailer. Theheader and trailercontain information used by the disk controller, such as a sector number and an error-correctingcode (ECC).
- When the controller writes a sector of data during normal I/O, the ECC is updated with avalue calculated from all the bytes in the data area. When a sector is read, the ECC isrecalculated and is compared with the stored value. If the stored and calculated numbersaredifferent,thismismatchindicatesthatthedataareaofthesectorhasbecomecorrupted and thatthedisk sectormaybebad.
- Most hard disks are low-level- formatted at the factory as a part of the manufacturingprocess. This formatting enables the manufacturer to test the disk and to initialize themappingfromlogicalblocknumberstodefect-freesectorsonthedisk.
- When the disk controller is instructed for low-level-formatting of the disk, the size ofdata block of all sector sit can also be told how many bytes of data space to leavebetween the header and trailer of all sectors. It is of sizes, such as 256, 512, and 1,024bytes. Formatting a disk with a larger sector size means that fewer sectors can fit on eachtrack;butit alsomeans that fewerheadersandtrailers arewritten on each track andmorespaceisavailableforuserdata.

Theoperatingsystem needstorecorditsowndatastructuresonthedisk.Itdoessointwostepsi.e.,Partitionandlogicalformatting.

1. **Partition**– is to partition the disk into one or more groups of cylinders. The operatingsystem can treat each partition as though it were a separate disk. For instance, onepartition can hold a copy of the operating system's executable code, while another holdsuserfiles.
2. **Logicalformatting (or creation ofa file system)**- Now, the operating system storesthe initial file-system data structures onto the disk. These data structures may includemapsoffreeandallocatedspace(aFATormodes)andaninitial emptydirectory.

Toincreaseefficiency,mostfilesystemsgroupblockstogetherintolargerchunks,frequentlycalled**clusters.**

BootBlock

Whenacomputerisswitchedonorrebooted,itmusthaveaninitialprogramtorun.Thisiscalledthe bootstrapprogram.

Thebootstrapprogram –

- InitializestheCPUregisters,devicecontrollers,mainmemory,andthenstartstheoperatingsystem.
- Locatesandloadstheoperatingsystemfromthedisk
- Jumpsto beginningthe operating-systemexecution.

The bootstrap is stored in read-only memory (ROM). Since ROM is read only, it cannot beinfectedbyacomputervirus.Theproblemisthatchangingthisbootstrapcoderequireschanging the ROM, hardware chips. So most systems store a tiny bootstrap loader program inthe boot ROM whose only job is to bring in a full bootstrap program from disk. The fullbootstrap program can be changed easily: A new version is simply written onto the disk. Thefull bootstrap program is stored in "the boot blocks" at a fixed location on the disk. A disk thathasabootpartitioniscalledabootdiskorsystemdisk.

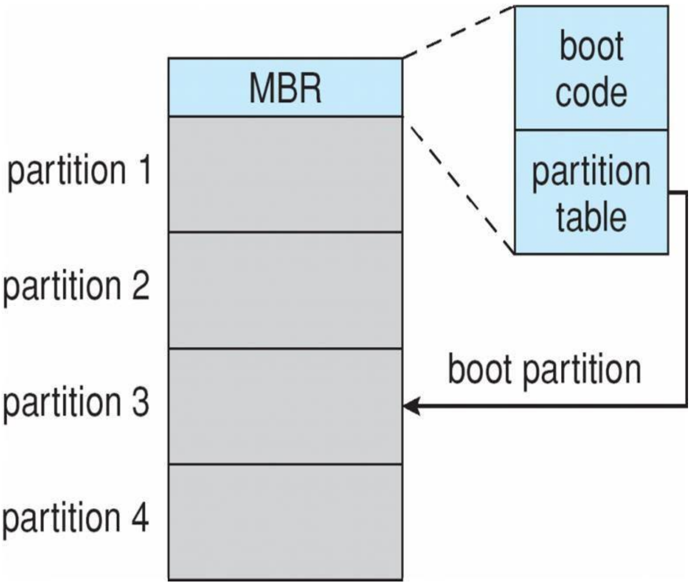


Figure:BootingfromdiskinWindows2000.

The Windows 2000 system places its boot code in the first sector on the hard disk (master bootrecord, or MBR). The code directs the system to read the boot code from, the MBR. In additionto containing boot code, the MBR contains a table listing the partitions for the hard disk and aflagindicatingwhichpartitionthesystemistobebooted from.

BadBlocks

Disksarepronetofailureofsectorsduetothefastmovementofr/whead.Sometimesthewholedisk willbe changed.Such group of sectorsthatare defectivearecalledas**badblocks**.

Differentwaystoovercomebadblocksare-

- Somebadblocksarehandledmanually,eg.InMS-DOS.
- Somecontrollersreplaceeachbadsectorlogicallywithoneofthesparesectors(extrasectors).Th eschemesusedaresectorsparing orforwardingandsectorslipping.

In MS-DOS format command, scans the disk to find bad blocks. If format finds a bad block, itwrites a special value into the corresponding FAT entry to tell the allocation routines not to usethatblock.

In SCSI disks, bad blocks are found during the low-level formatting at the factory and isupdatedoverthelifeofthedisk.Low-levelformattingalsosetsasidesparesectorsnotvisibleto the operating system. The controller can be told to replace each bad sector logically with oneofthesparesectors.Thisschemeisknownas**sectorsparingorforwarding**.

Atypicalbad-sectortransactionmightbeasfollows:

- Theoperatingsystemtriestoreadlogicalblock87.
- Thecontrollerfindsthatthesector isbad.Itreportsthisfinding totheoperatingsystem.
- Thenexttimethesystemisrebooted,aspecial,commandisruntotelltheSCSIcontrollerto replace the bad sectorwith a spare.
- Afterthat,wheneverthesystemrequestslogicalblock87,therequestistranslatedintothereplace mentsector's(spare)addressbythe controller.

Somecontrollersreplacebadblocksbysectorslipping.

Example: Suppose that logical block 17 becomes defective and the first available spare followssector 202. Then, sector slipping remaps all the sectors from 17 to 202, moving them all downone spot. That is, sector 202 is copied into the spare, then sector 201 into 202, and then 200 into201, and so on, until sector 18 is copied into sector 19. Slipping the sectors in this way frees upthespace ofsector18,sosector17canbemappedtoit.

SWAP-SPACEMANAGEMENT

- Swap-spacemanagementisanotherlow-level taskof theoperatingsystem.
- Swappingoccurswhentheamountofphysicalmemoryreachesacriticallylowpointandprocess esaremovedfrommemoryto swapspace tofreeavailablememory.

Swap-SpaceUse

- The amount of swap space needed on a system can vary depending on the amount ofphysical memory, the amount of virtual memory it is backing, and the way in which thevirtual memoryisused.Itcanrangefromafewmegabytesofdiskspacetogigabytes.
- The swap space can overestimate or underestimated. It is safer to overestimate than tounderestimate the amount ofswap spacerequired.Ifa system runsout ofswap spacedue to underestimation of space, it may be forced to abort processes or may crashentirely. Overestimation wastes disk space that could otherwise be used for files, but itdoesnootherharm.

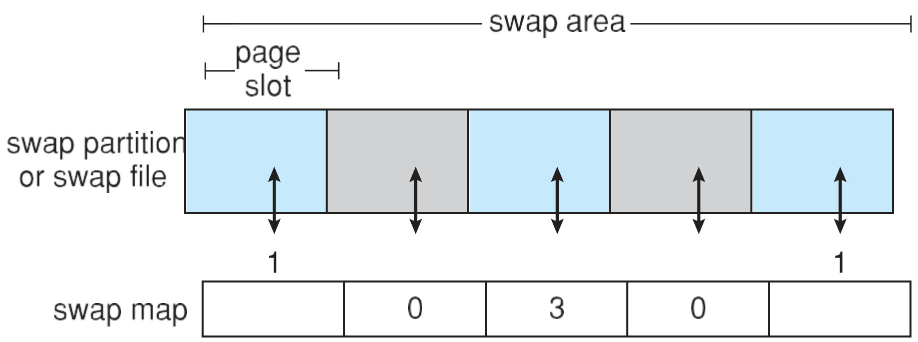
Swap-SpaceLocation

- A swap space can reside in one of two places: It can be carved out of the normal filesystem, or it can be in a separate disk partition. If the swap space is simply a large filewithin the file system, normal file-system routines can be used to create it, name it, andallocateitsspace.
- External fragmentation can greatly increase swapping times by forcing multiple seeksduring reading or writing of a process image. We can improve performance by cachingtheblocklocationinformationin physicalmemory.
- Alternatively, swap space can be created in a separate raw partition. A separate swap-spacestoragemanagerisusedtoallocateanddeallocatetheblocksfromtherawpartition.

Swap-SpaceManagement:AnExample

- Solaris allocates swap space only when a page is forced out of physical memory, ratherthanwhen thevirtualmemorypageisfirstcreated.
- Linux is similar to Solaris in that swap space is only used for anonymous memory or forregions of memory shared by several processes. Linux allows one or more swap areas tobeestablished.
- A swap area may be in either a swap file on a regular file system or a raw swap partition.Each swap area consists of a series of 4-KB page slots, which are used to hold swappedpages.Associatedwitheachswapareaisaswapmap—anarrayofintegercounters,eachcorrespondingtoa page slot in theswaparea.

- Ifthevalueofacounteris0,thecorrespondingpageslotisavailable.Valuesgreaterthan 0 indicate that the page slot is occupied by a swapped page.The value ofthecounterindicatesthenumberofmappingstotheswappedpage;forexample,avalueof3indicatesthatthe swappedpageis mappedtothreedifferentprocesses.
- Thedatastructuresfor swappingonLinuxsystemsareshownin belowfigure.



PROTECTION

GOALS OFPROTECTION

- Protection is a mechanism for controlling the access of programs, processes, or users tothe resources defined by a computer system. Protection ensures that only processes thathave gained proper authorization from the operating system can operate on the files,memorysegments,CPU,andotherresourcesofasystem.
- Protectionisrequiredtopreventmischievous,intentionalviolationofanaccessrestrictionbyauser.

PRINCIPLESOFPROTECTION

- A key, time-tested guiding principle for protection is the ‘principle of least privilege’. Itdictatesthatprograms,users,andevensystemsbegivenjustenoughprivilegestoperformtheirtasks.
- Anoperatingsystemprovidesmechanismstoenableprivilegeswhentheyareneededandto disable them when theyare not needed.

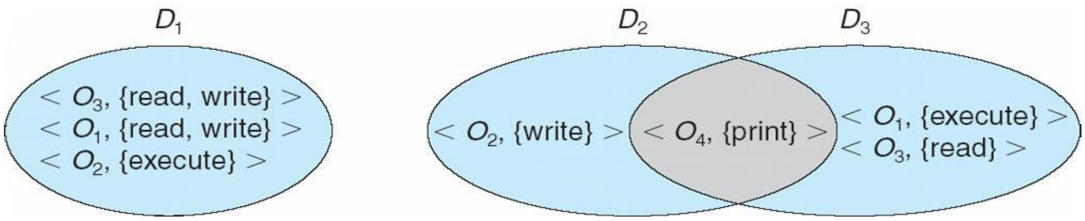
DOMAINOFPROTECTION

- A computer system is a collection of processes and objects. Objects are both hardwareobjects(suchastheCPU,memorysegments,printers,disks,andtapedrives)andsoftware objects (such as files, programs, and semaphores). Each object (resource) has auniquenamethatdifferentiatesitfromall otherobjectsinthesystem.
- The operations that are possible may depend on the object. For example, a CPU can onlybe executed on. Memory segments can be read and written, whereas a CD-ROM orDVD-ROM can only be read. Tape drives can be read, written, and rewound. Data filescan be created, opened, read, written, closed, and deleted; program files can be read,written,executed,anddeleted.
- Aprocessshouldbeallowedtoaccessonlythoseresourcesforwhichithasauthorizationand currentlyrequires to complete process

DomainStructure

- A domain is a set of objects and types of access to these objects. Each domain is anorderedpairof<object-name,rights-set>.
- Example,ifdomainDhastheaccessright<fileF,{read,write}>,thenallprocessexecuting in domain D can both read and write file F, and cannot perform any otheroperationonthatobject.

- Domains do not need to be disjoint; they may share access rights. For example, in belowfigure, we have three domains: D_1 D_2, and D_3. The access right < O_4, (print}> is sharedby D_2 and D_3,it implies that a process executing in either of these two domains can printobjectO4.
- A domain can be realized in different ways, it can be a user, process or a procedure. ie.eachuserasadomain, each process asa domainor eachprocedureasadomain.



ACCESSMATRIX

- Our model of protection can be viewed as a matrix, called an access matrix. It is ageneralmodelofprotectionthatprovidesamechanismforprotectionwithoutimposingaparticular protectionpolicy.
- Therowsof theaccess matrixrepresentdomains,andthe columnsrepresentobjects.
- Eachentryinthe matrixconsists ofasetofaccessrights.
- The entry access(i,j) defines the set of operations that a process executing in domain Dicaninvokeon object Oj.

domain \ object	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

- In the above diagram, there are four domains and four objects—three files (F1, F2, F3)and one printer. A process executing in domain D1 can read files F1 and F3. A processexecuting in domain D4 has the same privileges as one executing in domain D1; but inaddition,itcan alsowrite ontofilesF1andF3.
- When a user creates a new object Oj, the column Oj is added to the access matrix withtheappropriateinitializationentries,asdictatedbythecreator.

The process executing in one domain and be switched to another domain. When we switch aprocess from one domain to another, we are executing an operation (switch) on an object (thedomain).

Domainswitching from domain D_i todomain D_j isallowedifand only ifthe accessrightswitch access(i,j). Thus, in the given figure, a process executing in domain D_2 can switch todomain D_3or to domain D_4. A process in domain D_4can switch to D_1, and one in domainD_1canswitch to domainD_2.

domain \ object	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch
D_3		read	execute					
D_4	read write		read write		switch			

Allowingcontrolledchangeinthecontentsoftheaccess-matrixentriesrequiresthreeadditional operations: copy,owner,andcontrol.

domain \ object	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute		

(a)

domain \ object	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute	read	

(b)

The ability to copy an access right from one domain (or row) of the access matrix to another isdenotedbyanasterisk(*)appendedtotheaccessright.Thecopyrightallowsthecopyingofthe access right only within the column for which the right is defined. In the below figure, aprocess executing in domain D_2 can copy the read operation into any entry associated with fileF_2. Hence, the access matrix of figure (a) can be modified to the access matrix shown in figure(b).

Thisschemehastwovariants:

1. A right is copied from access(i,j) to access(k,j); it is then removed from access(i,j). Thisactionis a transfer of a right, ratherthan acopy.
2. Propagation of the copy right- limited copy. Here, when the right R* is copied fromaccess(i,j) to access(k,j), only the right R (not R*) is created. A process executing indomain$D_k$cannotfurthercopytherightR.

We also need a mechanism to allow addition of new rights and removal of some rights. Theowner right controls these operations. If access(i,j) includes the owner right, then a processexecutingin domain Di, canaddand removeanyright inanyentryincolumn j.

For example, in below figure (a), domain D1 is the owner of F1, and thus can add and deleteany valid right in column F1. Similarly, domain D2 is the owner of F2 and F3 and thus can addand remove any valid right within these two columns. Thus, the access matrix of figure(a) canbemodifiedtotheaccessmatrixshowninfigure(b)asfollows.

domain \ object	F_1	F_2	F_3
D_1	owner execute		write
D_2		read* owner	read* owner write
D_3	execute		

(a)

domain \ object	F_1	F_2	F_3
D_1	owner execute		write
D_2		owner read* write*	read* owner write
D_3		write	write

(b)

A mechanism is also needed to change the entries in a row. If access(i,j) includes the controlright, then a process executing in domain D_i, can remove any access right from row j. Forexample, in figure, we include the control right in access(D_3, D_4). Then, a process executing indomainD_3canmodifydomainD_4.

object / domain	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch control
D_3		read	execute					
D_4	write		write		switch			

IMPLEMENTATIONOFACCESSMATRIX

Differentmethods of implementingtheaccessmatrix(whichissparse)

- GlobalTable
- AccessListsforObjects
- CapabilityListsforDomains
- Lock-KeyMechanism

1. GlobalTable

- Thisisthesimplestimplementationofaccessmatrix.
- A set of ordered triples <domain, object, rights-set> is maintained in a file. Whenever anoperation M is executed on an object O_j, within domain D_i, the table is searched for atriple <D_i, O_j, R_k>. If this triple is found, the operation is allowed to continue; otherwise,anexception(or error)condition is raised.

Drawbacks-
Thetableis usuallylargeandthuscannot bekeptin mainmemory.AdditionalI/Oisneeded

2. AccessListsforObjects

- Each column in the access matrix can be implemented as an access list for one object.The empty entries are discarded. The resulting list for each object consists of orderedpairs<domain,rights-set>.
- It defines all domains access right for that object. When an operation M is executed onobject$O_j$inD_i,searchtheaccesslistforobjectO_j,lookforanentry<D_i,R_K>withMϵR_k.Iftheentryisfound,weallowtheoperation;ifitisnot,wecheckthedefaultset.IfMisinthedefaultset,weallowtheaccess.Otherwise,accessisdenied,andanexception condition occurs. For efficiency, we may check the default set first and thensearchthe accesslist.

3. CapabilityListsforDomains

- A capability list for a domain is a list of objects together with the operations allowed onthoseobjects.Anobjectisoftenrepresentedbyitsnameoraddress,calledacapability.
- To execute operation M on object O_j, the process executes the operation M, specifyingthe capability for object O_j as a parameter.Simple possession of the capability meansthataccessisallowed.

Capabilitiesaredistinguishedfromotherdatainoneoftwoways:

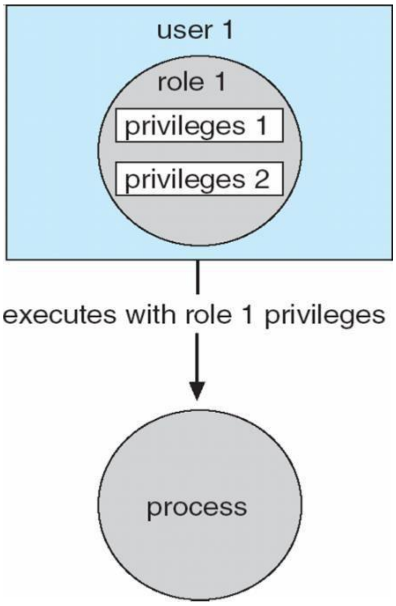
1. Eachobjecthasatag todenoteitstypeeitherasacapabilityoras accessibledata.
2. The address space associated with a program can be split into two parts. One part isaccessible to the program and contains the program's normal data and instructions. Theotherpart,containingthecapabilitylist,isaccessibleonlybytheoperatingsystem.

4. ALock-KeyMechanism

- Thelock-keyschemeisacompromisebetweenaccesslistsandcapabilitylists.
- Eachobjecthasalistofuniquebitpatterns,calledlocks.Eachdomainhasalistofuniquebitpatterns,calledkeys.
- Aprocessexecutinginadomaincanaccessanobjectonlyifthatdomainhasakeythatmatchesone ofthelocksoftheobject.

ACCESSCONTROL

- Each file and directory are assigned an owner, a group, or possibly a list of users, and foreachofthoseentities,access-controlinformationisassigned.
- Solaris 10 advances the protection available in the Sun Microsystems operating systembyexplicitlyaddingtheprincipleofleastprivilegeviarole-basedaccesscontrol(RBAC). Thisfacilityrevolvesaroundprivileges.
- A privilege is the right to execute a system call or to use an option within that systemcall.
- Privileges can be assigned to processes, limiting them to exactly the access they need toperformtheirwork.Privilegesandprogramscanalsobeassignedtoroles.
- Users are assigned roles or can take roles based on passwords to the roles. In this way, auser can take a role that enables a privilege, allowing the user to run a program toaccomplish aspecifictask,asdepictedinbelowfigure.
- This implementation of privileges decreases the security risk associated with super usersandsetuidprograms.



REVOCATIONOFACCESSRIGHTS

Thecapabilities aredistributed throughoutthesystem,wemustfind thembeforewecanrevokethem.Schemesthat implementrevocationforcapabilitiesincludethefollowing:

1. **Reacquisition**- Periodically, all capabilities are deleted from each domain. If a processwants to use a capability, it may find that that capability has been deleted. The processmay then try to reacquire the capability. If access has been revoked, the process will notbeabletoreacquirethecapability.
2. **Back-pointers**- Alistofpointersismaintainedwitheachobject,pointingtoallcapabilitiesassociatedwith that object.When revocation isrequired, we can followthesepointers,changingthecapabilitiesasnecessary.
3. **Indirection**- The capabilities point indirectly to the objects. Each capability points to auniqueentryinaglobaltable,whichinturnpointstotheobject.Weimplementrevocation by searching the global table for the desired entry and deleting it. Then, whenanaccessisattempted, the capabilityis foundto pointtoanillegal tableentry.
4. **Keys**- A key is a unique bit pattern that can be associated with a capability. This key isdefined when the capability is created, and it can be neither modified nor inspected bythe process owning the capability. A master key is associated with each object; it can bedefinedorreplacedwiththeset-keyoperation.

When a capability is created, the current value of the master key is associated with thecapability. When the capability is exercised, its key is compared with the master key. Ifthekeysmatch,theoperationisallowedtocontinue;otherwise,anexceptionconditionisraised.
In key-based schemes, the operations of defining keys, inserting them into lists, anddeletingthemfromlistsshouldnotbeavailabletoallusers.

CAPABILITY-BASEDSYSTEM

Here,surveyoftwocapability-basedprotectionsystemsisdone.

1. **AnExample:Hydra**

- Hydra is a capability-based protection system that provides considerable flexibility. Afixedsetofpossibleaccessrightsisknowntoandinterpretedbythesystem.Theserights include such basic forms of access as the right to read, write, or execute a memorysegment.Inaddition,auser(oftheprotection system)candeclareotherrights.
- Operations on objects are defined procedurally. The procedures that implement suchoperationsarethemselvesaformofobject,andtheyareaccessedindirectlybycapabilities. The names of user-defined procedures must be identified to the protectionsystem if it is to deal with objects of the user defined type. When the definition of anobject is made known to Hydra, the names of operations on the type become auxiliaryrights.
- Hydra also provides rights amplification. This scheme allows a procedure to be certifiedas trustworthy to act on a formal parameter of a specified type on behalf of any processthatholdsarighttoexecutetheprocedure.Therightsheldbyatrustworthyprocedureareindependent of, and mayexceed,therightsheld bythecallingprocess.
- When a user passes an object as an argument to a procedure, we may need to ensure thatthe procedure cannot modify the abject. We can implement this restriction readily bypassinganaccessright thatdoesnothavethemodification(write)right.
- Theprocedure-callmechanismofHydrawasdesignedasadirectsolutiontotheproblemofmutuallysuspicioussubsystems.
- A Hydra subsystem is built on top of its protection kernel and may require protection ofits own components. A subsystem interacts with the kernel through calls on a set ofkernel-definedprimitivesthatdefineaccessrightstoresourcesdefinedbythesubsystem.

2. **AnExample:CambridgeCAPSystem**

- A different approach to capability-based protection has been taken in the design of theCambridgeCAPsystem.CAP'scapabilitysystemissimplerandsuperficiallylesspowerful than that of Hydra. It can be used to provide secure protection of user-definedobjects.

CAPhastwokindsof capabilities.

1. The ordinary kind is called a data capability. It can be used to provide access to objects,but the only rights provided are the standard read, write, and execute of the individualstoragesegmentsassociatedwiththeobject.

2. The second kind of capability is the software capability, which is protected, but notinterpreted, by the CAP microcode. It is interpreted by a protected (that is, a privileged)procedure,whichmaybewrittenbyanapplicationprogrammeraspartofasubsystem.Apar

ticular kindofrightsamplificationisassociatedwithaprotectedprocedure.

QUESTIONBANK

1. Whatisafile?Distinguishbetweencontiguousandlinkedallocationmethodswiththeneatdiagram.
2. Explainfileallocationmethodsbytakinganexamplewiththeneatdiagram.Writetheadvantagesand disadvantages.
3. Explainfreespacemanagement.Explaintypicalfilecontrolblock, withaneatsketch.
4. Distinguishbetweensingleleveldirectorystructureandtwoleveldirectorystructures.Whatareitsadvantagesanddisadvantages?
5. Explainthe accessmatrix modelofimplementingprotectioninoperatingsystem.
6. For the following page reference string **1,2,3,4,1,2,5,1,2,3,4,5**. Calculate the page faultsusingFIFO,OptimalandLRUusing3and4frames.
7. ExplainDemandpagingindetail.
8. For the following page reference string **7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1**. CalculatethepagefaultsusingFIFO,OptimalandLRUusing3and4frames.
9. Explaincopy-on-writeprocessin virtualmemory.
10. What is a page fault? with the supporting diagram explain the steps involved in handlingpagefault.
11. Illustratehowpagingaffectsthesystemperformance.
12. Explainthe varioustypesof directorystructures.
13. Explainthe variousfileattributes.
14. Explainthe variousfileoperations.
15. Explainthevariousmechanismof implementingfileprotection.