Module I

ntutpter 1 *j* Overview of Operating System 1-1 to 1-30

Syllabus * Introduction ; Operating System Structure and operations, Process management, Memory management, storage management, Protection and security, Distributed and special purpose Systems; System Structure: Operating system services and interface, System calls and its types. System programs, Operating System Design and implementation, OS structure. Virtual machines, OS debugging and generation, System boot.

Module II

Chapter 2 : Process Management 2-1 to 2-32

Syllabus : Process concept : Process Scheduling, Opemtio n on process and Interprocess communication; Multithreading. lh?oc : Multithreading models and thread libraries, threading issues. Process Scheduling: Basic concepts. Scheduling algorithms an Criteria. Thread Scheduling and Multiple Processor Scheduling, —

Module IP |

Module IV

Module VI

Chapter 6 : Distributed Systems

P.8
System : Network ... 6.8.1
Communication Str... 6.8.2
Naming and ...
Protoc... Distribu,ed rae sys,em: 3
K file access. Stateful Versus Statfe, X '
Licata Distributed Synchronization Mutual Ex... 6.9
K P I 1 anrl Uparflocic nanrlltriA, &
Concurrency Control ...

□□□

CHAPTER 1

Module I

Overview of Operating System

~ ®yl,abua J?P'S..

Introduction : Operating System Structure and ... storage management, Protection 7n◁1 security oXteXnd ZT" 1 Memory mana 9eme "k Structure: Operating system services and interfaro e P Ca purpose Systems; System Operating System Design and ZtemeZ ion Os' X, X T u its tvbes. SyS,em rams, generation. System boot Im Plem6 "'o p on, OS structure. Virtual machines, OS debugging and

Syllabus Topic ; Introduction

1.1 Introduction to Operating System

☞ What ts OS?

➔ (Dec. 14, June 15, Nov. 15)

Q- What is operating system?

MU - Dec. 2014, June 2015, Nov. 2015, 2 Marks

An operating system is system software which manages, operates and communicates with the computer hardware and software. To complete the execution user program need many resources. The main job of the operating system is to provide resources and services to the user program. So without operating system, a computer would be useless.

☞ **Definition of operating system**

- An operating system acts as an interface between the user and hardware of the computer and also controls uL execution of application programs.
- Operating system is also called as resource manager.

☞ **Need of Operating System**

- Basically operating systems identifying penorm tasks, for example keyboard 8mn P fmm the tnp,lt devices such as devices such aT **'• Sending out Put to output of files and d "TM'" Or, pri ■"* etC1 and kce P'n8 'rack peripheral devfcXh on the disk> coo'«>Jling storage devices, Printers, scanners,'audio mixers
- The h * called CPU f * com Puter system is a processing unit unii to complete the execution and computer system should offer the service to allocate processing uni" to users program.
- The operating system allocates memory to user program as per need.
- In the same way, user programs interact with the other user programs through devices like key board or a mouse or even a joy stick.

Syllabus Topic : Operating System Structure

1.2 Operating System Structure

- As single user cannot keep either CPU or I/O uc vices busy all the time, operating system supports multiprogramming.
- In multiprogramming, operating system organizes the jobs in such a way that the CPU always get one job to execute, Hence, Multiprogramming improves CPU utilization.
- Multiple jobs remain in memoiy so that CPU can take any one for processing.
- During processing of job if I/O is required then CPU does not remain idle it takes other job for processing till I/O completes.
- Multiple users are permitted by time shared operating system to share the machine at the same time.
- Time shared system is capable of completing each action or command in a very less time. As a result, each user need can be fulfilled in very less time.
- A particular user program executes in allocated time by system and after expiration of slot, system gets allocated to other user program.

- So the allocation of computer resources in time slots to several programs simultaneously IS in time sharing logg ed manner. Consider the example of multiple use* tn same server.
- The resources in server machine i e, basis giving the allocated to each user on time sharing the feeling to each user that only he or she is using server machine exclusively.

Designs of the operating system

- Following are some of the designs of the operating system that is tried practically.

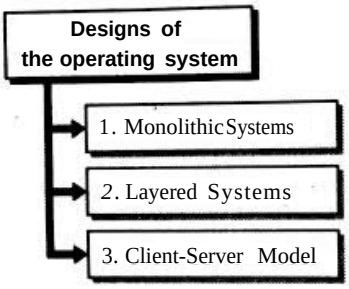


Fig. Cl.l : Designs of operating system

-F 1,2 J Monolithic Systems

Q. Explain monolithic system.

- This type of operating system can be treated as having no structure.
- The operating system is constructed as a set of procedures and each procedure can call any other ones whenever needed.
- Each procedure in the system has a precise interface in terms of parameters and results.
- Every procedure can call any other one, if the called one offers some useful compulation that the calh$_n$ – procedure needs, caning
- In this technique, compilation of all the procedures or files containing the procedures is <2 ““ , " ’ " “ ' rin'toa
- Every procedure is able to be seen procedure so the$_{rc}$ is noinfonnation “?very other
- A little structure is possible to i$_{mDotf}$ f^{m} systems, neolithic
- The system calls offered by the $_{0}$Derat - requested by putting the parameters in g Sysleni are place (e.g., on the stack) and then Wl<kfl ned instruction. $^{-xe}$cutuig $_{a}$ trap
- Due to this instruction machine gets SV/ , mode to kernel mode and transfers ^{1} fr$_{o}$m u®er operating system. ^{c}^{o}ntr$_{o}$] to the
- The operating system then fetches the find out which system call is to be execut ^{1},eters and

This organic suggests a basic StnMUrt for operating system :

- A main program that caUs demaoded %
- A Xwn of service procedures that perfa system calls-
- A collection of utility procedures that assist the serv procedures.
- In ihis representation, there is single service procafc for each system call which takes care of it.
- The utility procedures perform things that are dej. by several service procedures, such as fetching$_4$ from user programs.
- This separation of the procedures into three layers shown in Fig. 1.2.1.

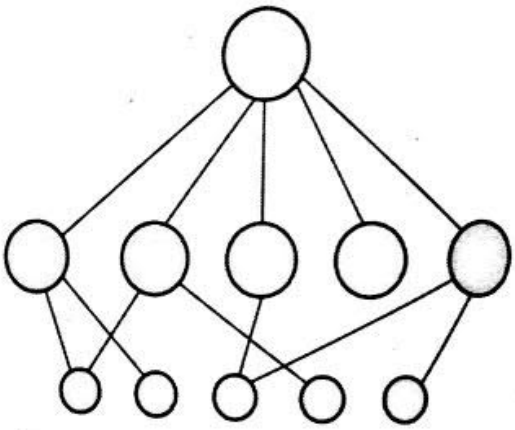


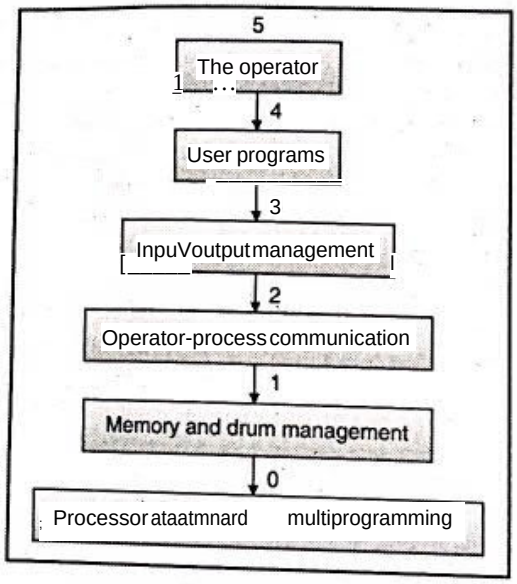
Fig. 1.2.1 : A simple structuring model ft$_{r}$ a monolithic system

➜ 1.2.2 Layered Systems

Q. Explain layered system in detail.

- Another appr system upon
- below it. $_{o}$f layeR , one constructed
- The first system built in this way was the THE syste deve' the ^{N} Xrlj£ Technische Hog school Eindhoven and students, , ^{y} & W, Dijkstra (1968)
- As sho
- Lay$_{er}$ 0 F ,8 1 2 2 016 system had totaJ 6 Ja) ers.
- ^{T}his |$_{ayer}$ Was
- ““ssor, co$_{nte}$, t^{SpOBsiWe} to handle allocation ^{5} f
- “ ^{mets} spired. ening when interrupts
- fbs la yer u, as res Ponsible to offer «* ^{J}
- “““Ptogran ^{g} of the CPU. Uy er 1
- ?* Memory management is handled by tbis la 5 was responsible to al locate space for proc main memory.

- If (here is no space in main memory, then it also allocates space on a 512K word drum used for holding parts of processes (pages).
- The layer 1 software was concerned of making needed pages were brought into memory' whenever they were required,



Hg* 1.2.2 : Structure of the THE operating system

☞ **Layer 2**

- " " ka,ion betw« n each PTOe« and,he
- oXr con'X e , Ch Pr « eSS effcCliVCly had ,,S Own

☞ **Layer 3**

- P* management of I/O devices and buffering the MsXr slreams and from thcm was handl «* b y
- On top of layer 3 each process could be able to deal with abstract I/O
- of devl'ces ™ devi « s wi 'b gtxtd properties, instead with many peculiarities.

☞ **Layer 4**

- The user programs wore resided in this layer
- mer XoToToZ < te, or I/O management. CM «™ d about process,

☞ **Layers**

The system operator process was located in layer 5.

- Th
- astern XulTT °«d of ‹he WayS 10 make Fig- 1.23, the onenf approach shown in number of layers. «g SyMem 1S broken up int0 a
- The bottom l
- the highest *layer\s hv* h is layer ° , * *the hardware* and N, which is the user interface.

- An operating system layer is an implementation of an abstract object made up of data and the operations that can manipulate those data.
- A characteristic operating system layer consists of data structures and a set of routines that can be invoked by higher-level layers.
- The same layer can too invoke operations on lower-level layers.

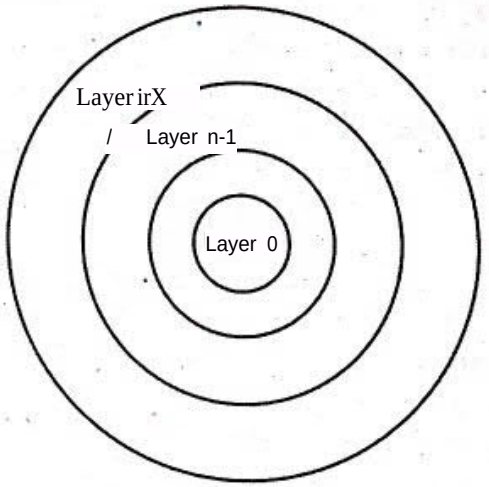


Fig. 1.23 : A layered operating system

☞ Advantages

The advantages of layered approach are that

- It keeps muuh better control over the computer and over the applications that utilizes that computer.
- Implementors have more liberty in changing the ntemal workings of systen and in ıting modular operating systems.
- Ease of construction and debugging.

☞ **Limitations of layered approach**

- defin vari tr o i Ubk W , th the layered a PP roach is to lower-level layers, cautious planning is necessary As, 3 iaycr can Use only
- SSr"" indined 10 be less 'fficient

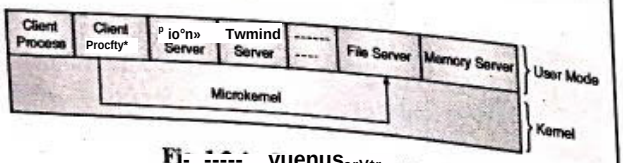
■*1.2.3 Client-Server Model

Q. Explain client-server model in detail.

- A s a large pan of the traditional operating system code of s imX ahi8herlayerCMS , UleVM/370g0tal0t
- However, VM/370
- XT ; i" g a « «s iX
- *A* trend in modern operating systems is to take the idea of moving code ui, into higher layers even further and

remove as much as possible from kernel mode, leaving a minimal microkernel.

- The usual approach is to implement most of (be operating system in user processes.
- To request a service, such as reading a block of a file, a user process (client process) sends the request to a server process, which then does the work and sends back the answer.
- In this model, shown in Fig 1.2.4, all the kernel handles the communication between clients and servers.
- By splitting the operating system up into parts, each of which only handles one facet of the system, such as file service, process service, terminal service, or memory' service, each part becomes small and manageable.
- As all servers run as user-mode processes, and not in kernel mode, they do not have direct access to the hardware.
- As a consequence, if a bug in the file server is triggered, the file service may crash, but this will not usually bring the whole machine down.
- Another advantage of the client-server model is its adaptableness to use in distributed systems.
- The communication of client with a server takes Place by sending messages.
- When client sends a message to Scrver 5| necessary for client to know whether th? " not processed in its own local maclZ 2 message * sent to a server on a remote machine in network **
- Either server is present on client m,.t.- remote machine in network from cli™ h ,,,, 011 a it appears to be : a request was sent JT"* View came back. . 1 dnd a response



Fi- ----. . vuenus erVtr Model

1-2.4 Monolithic Kernel vs . Microkernei

(Juqq 15 m ov. 15, Dec. 16)

Q. Differentiate between monolithic and microkernel
MU - June 2015, Nov 2015, 5 Marks

Q.7 What is Kernel ? Describe briefly the approaches of designing Kernel. **MU - Dec. 2016, 5 Marks**

- Tlte majority of CPUs supt T— opemtton i.e. kerne! mode and nserfc™ 0 mode,
- In kernel mode, ah instruction, of executed, and the entire metnory allc *ed is accessible throughout the egmff* « all reg'° **
- On the contrary to this, in user mode, memory register access is restricted.
- The CPU gets switched to kernel mode while operating system code.
- The only means to switch from user mode tn mode is through system calls as implemented operating system.
- The operating system can be put in full system calls are the only basic services anoS system provides.
- The operating system can also be put in f u jj hardware helps to restrict memory and registerto*
- If virtually entire operating system code is excqjj kernel mode, then it is a monolithic program ifa. in a single address space.
- The disadvantage of this approach is that it is d— to change or adapt operating system compos without doing a total shutdown and perhaps eveui: recompilation and reinstallation.
- Monolithic operating systems have drawbacks te viewpoint of openness, software engine reliability, or maintainability.
- If the operating system is organized into two par can provide more flexibility.
- In the first part, £et m °d u tes for managig hardware is kept wnich can uniformly well be exe in user mode.
- For example, rr-- memory management module keeps allocated and free space to the processes.
- 11 is required to when execute in kernel mode al the* (MMU) Xl? of the Menior y Managewerij areset.
- execute in ?' ro rnt contains only code IM' operating system mode. It ,s SCCODd P 3 1 Actually a .
- fot - context lcrokcn ,el require only contain *' ma "ipulati ng the ,t MMi' Setting deViCe , irrupts, MMU, and capturing
- Micrcikem i - 10 calls o n " Conta ins the code to pass sys®* 1T todui es arir i Proper user level operates Fi S' tZS sh rCtUn their sults. ystem with th?* that the organization of oper injs approach.

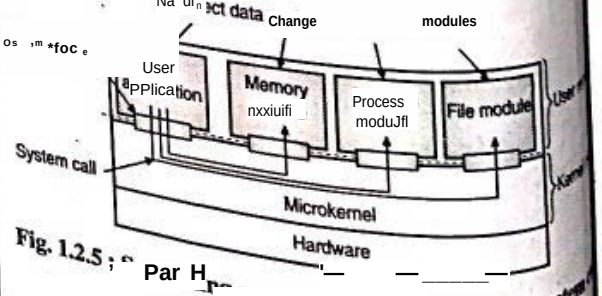


Fig. 1.2.5 ; S Par H

U»r UCatiOnS rrOra operating system a microkernel

Difference between monolithic and microkernel

Q. Differentiate between monolithic and microkernel → (June 15, Nov, 15)

MU-June 2015, Nov. 2015. 5 Marks

Sr. No.	Parameters	Microkernel	Monolithic kernel
1.	Definition	In microkernel, set of modules for managing the hardware is kept which can uniformly well be executed in user mode. A small microkernel contains only code that must execute in ---------- ££nd mQde U is the part of the operating system	If virtually entire operating system code is executed in kernel mode, then it is a monolithic program that runs in a single address space
2.	Address Space	There is separate address space for user services and kernel ---------- _service$,	There is same address space for user services and kernel services
3.	--	---------- Microkernel size is smaller than monolithic kernel.	It has a larger size compared to microkernel
4.	Execution speed	Execution speed is slower than monolithic kernel	Execution speed is faster than microkernel.
5.	Flexible	It is more flexible. Any module can be replaced.	It is not flexible as compared to microkernel. Its component cannot be changed, without doing a total shutdown and perhaps even a full recompilation and reinstallation,
6.	Crash	Crash of one service does not affect working of other part of microkernel.	After crash of service complete operating system fails.
7.	Communication	Jn microkernel, communication is through message passing.	Kernel calls the function directly.
8,	Drawbacks	Performance degradation due to context switching and communication overhead between different modules.	Monolithic operating systems have drawbacks from the viewpoint of openness, software engineering, reliability, or maintainability. It has lack of flexibility and having huge size.

Syllabus Topic j Operating System Operations

1.3 Operating System Operations

Q Bylain operating system operations.

- M<deni operating systems are interrupt driven. Operatmg system sits idle and WflU in case ' there are *o CXeCUte , n o VO devices to Offer service given n o USCr tO Whom res POnse is io be
- this case operating SyStem jUS ‹ WaiLS fOr so ™thing !o When interrupt or trap occurs then events are iigX
- 'lecX'd n thcn '■o" di,i0 " ariS " in SyStem durin 8 J"*"™ * « detected by CPU is called trap.
- Trap ur exactly same point of time of program execution Software interrupts are trap or exceptions. Example ts division by zero.
- After receiving an interrupt, operating system cany out some housekeeping so that it can resume its computation once it is through servicing the interrupt.
- After this a searching is carried out in the **interrupt vector or interrupt table.**
- Ihis table remains in kernel-memory space and includes address of the code in the device driver that is to service the interrupt.
- The interrupt handler is [1] then executed. When the handler finishes, control of the CPU is returned to the code that was executing before the interrupt occurred.
- Interrupt service routine (ISS) deal with interrupts.
- The design of the operating system should be in such a way that one erroneous or malicious program should not affect other programs.

1.3.1 Dual Mode Operation

- There are two modes of operating system.
- The majority of CPUs support at least two modes of operation i.e. kernel mode and user mode.

- executed, and the entire memory an sc t of all registers allowed to be
- is accessible throughout the execution.
- On the contrary to this, in **user mode**, memory an register access is restricted. The CPU gets switch d to
- kernel mode while executing operating system cede.
- The only means to switch from user mode to kerne mode is through system calls as implemented by operating system.
- Hardware contain mode bit which indicate kernel mode when 0 and user mode when set to 1- When user application is running then system is in user mode.
- When user application request the service from operating system then transition from user mode to kernel mode take place.
- During booting of the system, hardware is in kernel mode. Once the operating system is loaded and start executing user application system goes in user mode.
- Due to dual mode operation, protection from misbehaving users is achieved. This protection can be provided by allowing the execution of privileged instructions in kernel mode only.
- Any attempt to execute these privileged instructions in user mode cause trap to operating system.
- The instruction to enter in user mode, Instructions for I/O control, timer management, and interrupt management are some of the examples of privileged instructions.
- ♦From user mode, control is switched to operating system through interrupt, a trap, or a system call.
- The interface between OS and user programs is deflned by toe set of system calls that the operating system
- System call is the call for the operating system to sometaskonbehalfoftoeusefsp rogram
- X:x::x:r interfactbetw
- There are two ways for the nroc e ss to switch from the user mode to the kernel mode.
- One is when a user process explicitly request kernel mode by issuing a system call.
- Other is during the can take over to Tm ition of user process kernel task. so "* syste " housekeeping

13.2 Timer

- Computer contains sta l clocks containing crysy oscillator, a counter, and a holding register.
- A quartz crystal under tension produces a periodic high accurateness, usually in the range of Xi — * megahertz to a few Gigahertzes, depending on the crystal chosen.
- By electronic circuitry, this base signal can be multiplied by a small integer to get frequencies up to several gigahertz and more than this.
- Computer contains al least one circuit, than produce synchronizing signal and given to many circuits in computer.
- This synchronizing signal is fed into the counter io make it count down to zero. When the counter gets to zero, it causes a CPU interrupt.
- In **one-shot mode,** after the clock is started, holding register value gets copied into the counter and then decrements the counter at each pulse from **the** crystal.
- When the counter value becomes zero, it causes an interrupt and stops until it is explicitly started again by the software.
- In **square- wave mode,** after getting to zero an causing the interrupt, the holding register i automatically copied into the counter, and the whole process is repeated again indefinitely.
- These periodic interrupts are called **clock ticks.** Ew computer has a battery-powered **backup** clock to si the current time when machine is switched off.
- Universal coordinated time is used to synchronize docks of the machines.
- Operating system ensures to set timer to in terni before hand over the control to user application.

Syllabus : Process Management

1.4 Process Management

Q. Explain process in operating system management function

- A program or n cation under execution is called a process. A process includes program code, execution context require any resources on disk. On disk it do* "1

- A program gets executed in main memory. So it should be transferred from disk to main memory.
- To complete execution, program needs many resources and competes for it. These resources are memory, CPU, files etc
- naw it becomes process. From the computation context X oMiew. a process is defined by CPU state, memory contents and execution environment,
- Process manager implements the process management functions.
- tomnWrogramming. single CPU is shared among
- k X processes remain busy in completing UO CPU is allocated to only one process at a given point of time.
- Here some policy is required to allocate CPU to J ess, called as CPU scheduling.
- If multiple users are working on the system, operating system switches the CPU from one user process to other.
- User gets the illusion that only he or she is using the system.
- Process synchronization mechanism is required to ensure that only one process should use critical section.
- Process communication and synchronization, deadlock handling, suspension and resumption of processes and creation and deletion of the processes etc are some o the activities performed in process management.
- Program counter contains address of the instruction from which execution resumes after context switch.
- The process management activities involves .
  1. To provide control access to shared resources like file, memory, I/O and CPU.
  2. Control the execution of user applications,
  3. Creation, execution and deletion of user an system processes.
  4. Resume a process execution or cancel it.
  5. Scheduling of a process.
  6. Synchronization, interposes communication an deadlock handling for processes.

Syllabus Topic : Memory Management

1.5 Memory Management

Q. Explain memory management function of operating system.

- Memory is an important resource of the computer system that needs to be managed by the operating system. To execute the program, user needs to keep the program in main memory.
- The main memory is volatile. Therefore a user needs to store his program in some secondary storage which is non volatile.
- Every process needs main memory since a process code, stack, heap (dynamically- allocated structures), and data (variables) must all reside in memory.
- The management of main memory is required to support for multiprogramming. Many executables processes exist in main memory at any given time.
- Different processes in mam memory have different address space.
- Memory manager is the module of the operating system responsible for managing memory.
- Programs after completion of execution move out of main memory or processes suspended for complehon of I/O can be swapped out on secondary storage to free the main memory for other processes.
- New processes are required to be loaded in main memory. If available main memory is not sufficient to hold all processes swapping between main memory and secondary memory is done.
- Memory managers move processes back and forth between memory and disk during execution.
- So it is required that operating system should have some strategy for the management of memory.
- The memory management activities handled by operating system are :
  1. Allocation of memory to the processes.
  2. Free the memory from process after completio n of execution.
  3. Reallocation of memory to a program after used block becomes free.
  4. Keep irack of memory usage by the process.

Syllabus Topic : Storage Management

1.6 Storage Management

- The operating system gives a uniform, logical view of information storage for making the use computer system convenient to users.

operties of its

- The OS abstracts from the phy»"J P[r]orage unit, the storage devices to describe a logical flle
- .™«d out by operating system
- The mapping of files is carorf out y pe accessed by onto physical media. These files are operating system via the storage devices.

1.6.1 Flle Management

Q. Explain file management in OS.

- Following are the necessities for long term information storage,
  - c It must be possible to store a very large amount of information.
  - o The information should not loss after termination of the process using it.
  - o Several processes must be able to access the information simultaneously.
- In order to fulfill the above requirements, it is necessary to store information on disks and other secondary storage in units called **files.**
- A file is a named collection of related information that is recorded on secondary storage.
- The data cannot be written directly to secondary storage unless they are within a file. Processes can then read information from the file. It also can write new information into the file if needed.
- After process termination, information in file should remain retained and should not vanish.
- A file should only vanish when its holder clearly
- " "■ tiag system manages the files ,
- rue system describes how files are structured, named, accessed, used,
- The fle mana protected implemented. gement activities nf consist of : operating system
  - ,■ Creation Md deletion of files and d - rectories.
  - 2 - -mvide access to fi »es J and Nation of storage We for files.
  - '• Wnback-apof files.
  4. File Security.
  5. Providing iheprtauti ves for Elating Qles and Atones.

1.6.2 Mass Storage Management

Q. Write note on mass storage management.

- Main memory « computer system . Storage dev devices are used ' ssjon devices (network cards (disks, tapes), modems), and human-interface devi « i (screen keyboard, mouse).
- Two main types of storage technology used today are and optical storage. Primary magnetic "'"Alices are: Diskettes, Hard disks (both fixed "movabJe), High capacity floppy disks, Disk cartridges, and Magnetic tape.
- Primary optical storage devices are: Compact Disk, Read Only Memory (CD ROM), Digital Vtdeo Disk Read Only Memory (DVD ROM) , CD Recordable (CD R), CD Rewritable (CD RW), Photo C.
- The hard disk is used as main storage device in your computer system. It is magnetic type of storage device.
- Within one hard disk unit, many physical disks are present. Each disk is known as a platter.
- Several metal platters are present within the hard disk and these are coated with a special magnetic material.
- The platters rotate many thousands of times within a second for accessing the data. Magnetic read and write head are present which floats just above the surface of the platter.
- Material used for making the platters is alunwiu glass, or ceramic and two rcad/write heads are pre$£ one for upper and lower surface.
- Platt® are arranged in stack because of which the
- its cyfindw' 116 rea<Uwrite heads often ls referred "
- Cylinder is the location of a single track through all Platter
- read from and • i he mechanis ni, time required
- Now days it is
- partitioned ana 6 PUt o f hard disk - Hard disk can drive. Storam. partl,ion can be used as sepa or terabytes capacity o f hard "ink can be in gigabyte
- The device management tasks include :
  1. To open, close and write device drivers,
  2. Communicate, co ntrol and monitor the device driver.

1.6.3 Ca chin g

Q. "Wain how caching is implemented?

The memory system can be viewed as hierarchy of 4 layers Top layer contains many registers which are inside the CPU.

As material used to build the registers and CPU is same, these registers are faster just like CPU.

These registers can be accessed quickly. For these registers, typical access time is around 1 nanosecond and storage capacity is around 1KB.

The second layer of the hierarchy is cache memory. The frequently used data and instructions are cached in cache memory to improve the performance this memory is controlled by the hardware.

The division of primary memory is done into **cache lines** of 64 bytes, having addresses 0 to 63 in cache line 0, 64 to 127 in cache line 1, and so on. The most frequently used cache lines are kept in a high-speed cache sited inside or very close to the CPU.

If cache hardware finds that, needed line by program is in cache then it is cache hit.

If it is not in cache then it is treated as cache miss and needs to pay for more time as request would be satisfied from main memory.

- For cache, typical access time is around 2 nanosecond and storage capacity is around 4 MB.
- In main memory, a buffer is reserved for disk sectors called as disk cache.
- The cache keeps a copy of data in some of the sectors on the disk.
- To fulfill an I/O request for a particular sector, first disk cache is checked to see whether the sector is in disk cache.
- If it is present, then request is fulfilled via the cache. If not present, then required sector is read into the disk cache from the disk.
- The disk cache improves the performance as some request is satisfied from it.
- The property of locality of reference is used .When a block of data is present in cache to fulfill a single I/O request, it is expected that same block will be needed in future.
- Due to disk cache mechanism, time required to read from and write to hard disk is improved. Now days it is the part of hard disk.
- Hard disk can be partitioned, and each partition can be used as separate drive. Storage capacity of hard disk can be in gigabyte or terabytes. For magnetic disk, typical access time is around 10 nanosecond and storage capacity is around 1 to 4 TB.
- The recently read data remain in disk cache. Sometimes adjacent data which is expected to be accessed next time is held by disk cache. Some disk cache offers write caching also.

1.6.4 I/O Systems

- As we know that I/O is one of the main functions of an operating system and is used to control the entire computer's Input/Output devices.
- Basically it should have to issue commands to the devices, catch interrupts, and handle errors.
- It should also provide an interface between the devices and the rest of the system that is simple and easy to use.
- The I/O code represents a significant fraction of the total operating system. Computers operate with many kinds of devices.
- As we know that it include storage devices (disks, tapes), transmission devices (network cards, modems), and human-interface devices (screen, keyboard, mouse).
- Following are the components of I/O subsystem.
  - o Buffering, Caching, and Spooling which are memory management components.
  - o Device driver interface.
  - o Drivers for specific devices.

1.7 Protection and Security

Q. Explain protection and security mechanism of OS.

- Protection mechanisms refer to the particular operating system mechanisms which are used to protect information, files and resources in the computer.
- Policy means whose data should be protected from whom and mechanism is how system put into effect these policies.
- In some systems, a program called reference monitor is used to impose a protection.
- Any attempt to access the resource is verified by reference monitor whether it is legal or not.

- The referee makes a decision on the basis Of policy table.
- Modem protection mechanism developed to improve the reliab.hty of « P lex sysle m in which use of sharedresources» inv
- There are many reasons to offre the protectio n lo avoid protect n
- h ts needed to prov.de the ol0 an access mischievous, deliberate vtolahon
- rr T m a h c sure that, each ram component ringin the system uses system of the „ per defined policies to ensure the rehab.l y SySlero r he resources of computer
- Policies to make use of the resources ich is system are put into effect by mechamsm wh offered by protection.
- Some of the policies are included in the design of the rvctpm Other policies are decided by the managemen of a system
- Many policies are defined by the individual user of the system to protect their own files and programs.
- A protection system should be flexible enough to offer different and put into effect the different types of policies.
- Different types of applications have different types of resource use.
- The policies designed should allow to offer this resource use need.
- The resource use of the applications can change over the period of time.
- Hence, instead of relying totally on operating system, application programmer should use protection mechanism to protect the resources created against misuse.
- The system is secure if its resources are used and accessed as proposed under all conditions. Total security is impossible to achieve.
- We must have mechanisms to make security breaches an exceptional occurrence, rather than the rule.
- The challenge in developing operating systems security is to design security mechanisms that safeguard uroce« execution and their generated data in an environment with complex interactions.
- While protection of the system related to intern 1 environment, security, on the other hand " consideration of the external environ™ •' , re< 1 uires within which the system operates.
- Only sufficient protection system is not enough for total security- refer m tb C particular operatin protection n*' 1 wbic h are used to pro | ~ system reechan the computer security encomp asses the information re of securi ty. overail problem Thre e of more important
- Seci rity has many aspects. ones tr c
  - The nature of rite threats,
  - The nature of intruders.
  - Accidental data loss.

1.7 1 Threats

uiewnoint of security, computer systems have From * L s wi* corresponding threats to them, three general goals.

These are shown in fig- c • ■

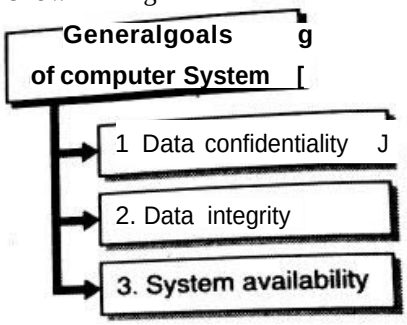


Fig. C1.2 : General goals of computer systems

1. Data confidentiality

- Threat to this goal is publicity of data. Aiwa} confidential data should remain confidential. £
- System should always ensure that, the access rightstol data should be given only to authorized person. a
- At least, the owner of the data should be able to staJ who can access what, and the system should enfant these specifications.

2, Data integrity

- Tl Weat t0 * s g° a l is modification of data. Af unauthorized users should not be able to make modification to any data without the o wnCl Permission,
- Data modification includes not only changing the data but also deleting data and addition of false data as well
- The system it remain urich 0 alWays £ uar antee that data sto red i it, a ged until the owner decides to chang

3. System availability

- System should be usable at any time. Nobody should disturb the system to make it unusable.
- Denial of service attack is increasingly common to make system unavailable.
- Protecting individuals from misuse of information about them is called privacy.
- This quickly gets into many legal and moral issues. System should also ensure the privacy to individual user.
- The system contains many hardware objects such as CPUs, memory segments, disk drives, printers, magnetic tapes and many software objects such as processes, files, databases, or semaphores.
- Each object can be referenced by its unique name. On every object a finite set of operations can be performed. For example, WAIT and SIGNAL on semaphores and READ and WRITE on files.
- A system should enforce a mechanism, to restrict the processes from accessing the needed objects for which they are not unauthorized.
- The mechanism should also ensure to restrict processes to a subset of the legal operations when that is needed.
- For example, process P has a permission to read file F but not of performing write operation on it.
- A set of object and rights pair is called **domain.** Each pair denotes an object and some subset of the operations that can be performed on it,
- One domain corresponds to one user and specify the permissions to user for certain activities. Consider the following three domains.
- It is possible for the same object to be in multiple domains. [Read, Write, execute j rights are available on each object. At a particular time of execution, each process executes in some protection domain.
- In that domain there is some set of objects it can access, and for each object it has some set of rights shown in square brackets,
- During execution, processes can go from one domain to other domain. The rules for domain switching are very much depends on and varies from system to system
- In Unix every process is defined by user-id and group-id(uid.gid).
- Two processes with the similar (UID, GID) combination will have access to precisely the same set of objects.

- Process often switches from user area to kernel area, tnat is, from one domain to other.
- The kernel part has access to a different objccls from the user part.

Sy,1abua **Topic : Distributed Sya~**

1.8 Distributed System

- A computer network is defined as a set of communicating devices that are connected together by communication links.
- These devices include computers, printers and other devices capable of sending and/or receiving information from other devices on the network.
- These devices often called as node in the network. So computer network is interconnected set of autonomous computers.
- A distributed *system is defined as set of autonomous computers that appears to its users as* a jingle coherent jys/em. Users of distributed system feel that, they are working with a single system.
- Distributed system is like multicomputers spread worldwide. Each node in distributed system is having its own CPU, RAM, network board. OS, and disk for paging.

Main characteristics of distributed system

Q. What are the characteristics of distributed system? Explain.

- A distributed system comprises computers with distinct architecture and data representation. These dissimilarities and the ways al) these machines communicate are hidden from users.
- The manner in which distributed system is organized internally is also hidden from the users of the distributed system.
- The interaction of users and applications with distributed system is in consistent and identical way, in spite of where and when interaction occurs.
- A distributed system should allow for scaling it.
- Distributed system should support for availability. It should be always available to the users and applications in spite of failures.
- Failure handling should be hidden from users and applications.

Comparison of muttlprocaaaoh muttlcomp and distributed ayatem

Parenwlara	Multlprocewx	MulUcornput«<	Distributed System
Node configuration	CPU	CPU, RAM, net interface	Comp*'** cfxnput
Node penpherafs	Al stared	Shared exc Maybe disk	Full s* pef node
Location	Same rack	Same room	Possibly worldwide
Internode communication	Shared HAM	Dedicated interconnect	Traditional network
Operating systems	One, shared	Multiple. same	Possibly aN different
Filesystems	One, shared	One, shared	Each node has own
Administration	One organization	f One organization	Many organizations

- Network contains computers with different architectures (heterogeneous).
- To offer the single system view of distributed system in this heterogeneous environment, a middleware layer is often placed between higher layer that comprises of user and applications and lower layer comprising the operating system and communication facilities.
- This organization of distributed system is called as middleware is shown in Fig. 1.8.1.

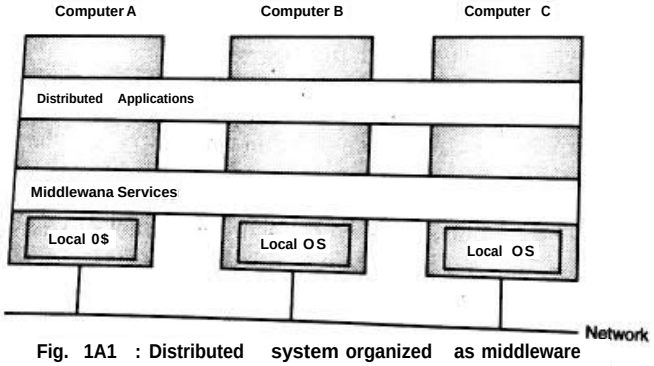


Fig. 1A1 : Distributed system organized as middleware

- In Fig. 1.8.J. applications are tunning on three different machines A, B and C in network
- As distributed systems are built on top of computer networks, which are of two types LANs (LocaX Networks), and WANs (Wide Area Network s).
- LAN covers one room, building or campus WAN ... city, country, or wh Ole world Ethemet is

ex amp* of LAN. Internet is sands of ... and can be co'XX one WAN. sepor 01 Machines * n i7 internet communicate by excha,, ging packets.

- Each P** et CO1 tu lins address of source and destiny routing tables. It extracts addr« s . Router contain; inc0 ming Packet ano look up the table to find which I on and thus to which outgoing line to send the P
- This'prccess is repeated until the packet is delivered to
- X uZg° are very dynamic «« «« updated tinuousfy as routers and links go down and come back up and as traffic conditions change.
- Protocol is set of rules by which Compaq communicates with each other.
- Many protocols are present such as router. router protocols, host-host protocols, and others. Prot«; 01 stack layers different protocols on top of one another This protocol stack is used by all modem network Different layer in protocol stack deal with differ issues.
- Most of the distributed systems use the Internet as a base. Hence, these system uses two important Internet protocols: IP (Internet protocol) and TCP (Transmission Control Protocol),
- IP (Internet Protocol) is a datagram protocol. In IP, a sender sends datagram of up to 64 KB over the network and no guarantees are given for its delivery.
- The datagram may be fragmented into smaller packets and travel independently, possibly along different routes.
- At destination host when all packets reaches, they are * assembled in correct order as per sequence number and delivered to the application.
- IP protocol has two versions, v4 and v6. Version 4(V4) are currently in use and v6 is up and coming. IP v* packet starts with a 40-byte header that contains eat 32 bit source and destination address with other fields.
- These are called **IP addresses** and routing is carried out using these addresses.
- intereT t ' o ffer re iable comm unication in * (Transmi i° Control Protocol), is prese rehable communication TO nt on top

- yCP H mikes use of IP to offer connection-oriented process always listen the incoming c ion on port number.
- Remote process is specified by IP address of machine and port number.
- Sender first estat lishes the connection and sends bytes over that connecti ion which is guaranteed to come out (hc other end und imaged and in the correct order.
- Th$_C$ TCP gives' this S$^{UEirantee\ by\ using\ sequenc?}$ jILimOLTSt checksums, and retransmissions of incorrectly received packets.
- r. machine is identified by IP address and it is difficult manage such list of huge number of IP addresses, DNS (Domain Name System) was invented as a database that maps ASCII names for hosts onto their IP addresses.
- This naming system pemtits the mail program on the sending machine to search the destmation host s IP address in the DNS database, establish a TCI connection to the mail daemon process there, and send the message as a file.
- The user-name is sent along to identify which mailbox to put the message in.
- A tightly-coupled operating system is called as a **distributed operating system (DOS),** and it manages multiprocessors and homogeneous multicomputer.
- The main objective of distributed operating system •* <° hide the facts of managing the underlying hardware such that it can be shared by multiple processes.
- The loosely-coupled **network operating system (NOS)** is used for heterogeneous multicomputer systems.
- Management of the underlying hardware is an important issue for a network operating system.
- The difference from traditional operating systems comes from the fact local services are made available to remote clients.

Syllabus Topic : Special Purpose Systems

1,9 Special Purpose Systems

Q. Explain various special purpose systems.

Apart 'from general purpose computer system, some classes of computer systems have limited functions and ns objectives are deal with limited computation domains. These are shown in Fig. C1.3.

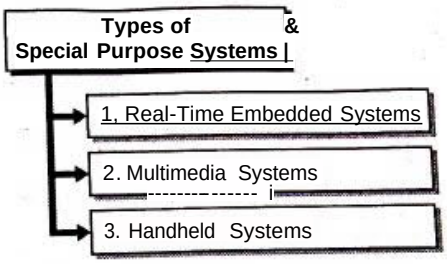


Fig. CL3 : Special purpose systems

➜ 1.9.1 Real-Time Embedded Systems

- Embedded computers are the most common form of computers. These devices are used in every field of our life, from car engines and manufacturing robots to VCRs and microwave ovens.
- These devices perform very specific tasks. These embedded systems differ significantly.
- These can be general-purpose computers, running standard operating systems such as UNIX or Linux with special-purpose applications to implement the functionality.
- Some of the systems are hardware devices with a special-purpose embedded operating system providing just the functionality desired.
- The embedded systems are becoming more multifaceted and complex today. Also these systems will affect our life with more involvement.
- This means they will bear more and more responsibilities on their shoulders to solve real lime problems to make our life easier.
- So real time operating system needs to be effective to manage more complex real time applications.
- Real time operating systems must respond quickly. These systems are used in an environment where a large number of events (generally external) must be accepted and processed in a short time.
- Real time processing necessitates quick dealing and characterized by providing instant response. For example, a measurement from a petroleum refinery indicating that temperature is getting too htgh and might demand for immediate attention to avoid an explosion.
- In real time operating system swapping of programs from primary to secondary is not frequent.
- Most of the time, processes remain in primary memory in order to provide quick response, therefore, memory.

system is less demanding management in rea! time compared io other system

- The primary functions of the real time operfltW g syslem are :
  - Management of the CPU and other resources prrcments of an application. to fulfill the irqui
  - Synchronization witfl and responding to the system event.
  - Efficient movement or 'ata amc ng jesses and out coordination among these processes.
- In addition to these primary functions, there are ce uIT accond-y functions that am no. compd -y b included *to* enhance die performance. Tbeseam
  1. to provide an efficient management of KA ..
  2. To provide an exclusive access to die computer resources.
- ftw more examples of real lime processing are :
  1. Airlines reservation system.
  2. Air traffic control system.
  3. Systems that provide immediate updating.
  4. Systems that provide up to the minute information on stock prices.
  5. Defense application systems like as RADAR.

-> 19.2 Multimedia Systems

- Apart from conventional data, now days operating system should be able to handle multimedia data.
- Multimedia data comprises audio and video files as well as conventional files.
- Multimedia data should be delivered to the application in defined time constraint. E.g. Video frames must be streamed as 25 frames per second.
- Multimedia depicts a broad range of applications that are in well-liked use today.
- These comprise audio files such as MP3 DVD movies vdeo conferencing, and short video clips of movie previews or news stories downloaded over the Internet
- Multimedia applications may also broadcasting over the World Wide Web Fn broadcasting of speeches or sporti „g evems -omprise r example,
- Multimedia applications include comhinw audio and video. For example, a movie m a"" b o*h separate audio and video tracks. * Co„ S,S‹ Of

ini s delivered to de sItt op

- Multin-" a 8l,p, 'X" O d directed toward smaller personal « D,p "" As M(J cellar telephones as WelJ devices also support for
- The design t* ,,,™.dia systems. ""Toneme * S,em Sho Uld requirements of

-> 1.9.3 Handheld Sy*™" 9

Q. Write m .S andMjdsystenm.

- Personal Digital Assistants (PDA_J* for example pa■i and pocket-PCs and cellular telephones are examples of handheld systems.
- Several of such system uses special-purpose embedded operating systems.
- These devices are having small size and have a small amount of memory, slow processors, and smal. d isplay screens.
- Physical memory size of these devices is bet Wee „ 512 KB and 128 MB.
- Therefore it is a job of operating system and applications to manage manor efficiently.
- A second concern to developers of handheld devices is the speed of the processor used in the devices. Handheld devices require faster processors with compare to PC.
- Faster processors need more power. So it is obvious that, handheld devices require large battery size and will occupy more space for battery.
- Therefore most handheld devices use smaller, slower processors that consume less power.
- As a result, the operating system and applications must be designed not to toll the processor.
- The final issue deal with program designers fof handheld devices is I/O. A short of physical space restricts input methods to small keyboard: , handwriting recognition, or small screen-based keyboards.
- The srnall display screens also provide constraints on output options.
- The — u! display for a handheld device limits to 3 inches square with compares to 30 inches in PC.
- Therefore reading e-mail and browsing "" p,8„ ■ " ●* duced into smaller displays.
- A solution to this problem is to deliver small set of web page and display called as web clipping.

AyllnbusTopic: Operating System Services

1.10 Operating System Services

-> **(May** 16)

Q. • Torvices provided by operating system.
MU ■May 2016. 5 Marks

Following are the six services provided by operating system for efficient execution of users application and to make convenience to use computer system.

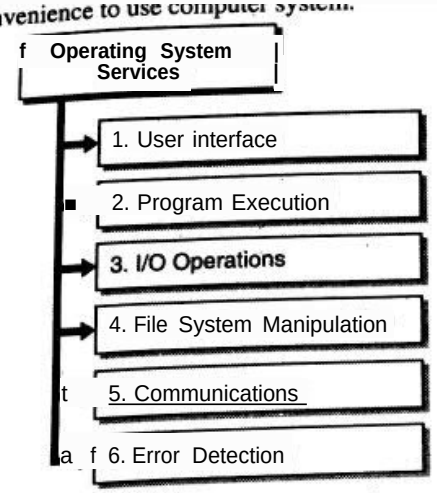


Fig. C1.4 : Operating system services

➜ 1. User Interface

- User interface is essential and all operating systems provide it. Users either interface with the operat.ng system through command-line interface or graphical user interface or GUI.
- Command interpreter executes next user-specified command. A GUI offers the user a mouse-based window ..nd menu system as an interface.

■4 1 Program Execution

- The operating system provides an environment to run users programs efficiently.
- The resources needed Io the programs to complete execution are provided by operating system ensuring optimum utilization of computer system.
- Memory allocation and deal location, processor allocation and scheduling, multitasking etc functions are performed by operating system.
- The operating system has all rights of resource management. User program does not given these rights.

-4 3. I/O Operations

- Each program requires cannot produce output without taking any input.
- This involves the use of I/O. During execution, program requires to perform I/O operations for input or output.
- Until I/O is completed, program goes in waiting state. I/O can be performed in three ways i.e. programmed I/O. Interrupts driven I/O and I/O using DMA. The underlying hardware for the I/O is hidden by operating system from users.
- These I/O operations make it convenient to execute the user program.
- Operating system provides this service to user program for efficient execution.

-4 4, File System Manipulation

- Program takes input, processes it and produces the output. The input can be read from the files and produced output again can be written into the files.
- This service is provided by the operating system. All files are stored on secondary storage devices and manipulated in main memory,
- The user does not have to worry about secondary storage management.
- Operating system accomplishes the task of reading from files or writing into the files as per the command specified by user,
- Although the user level programs can provide these services, it is better to keep it with operating system.
- The reason behind this is that, program execution speed is fundamental to VO operations.

➜ 5, Communications

- Cooperating processes communicates with each other. If communicating processes are running on different machines then messages are exchanged among them and they gets transfer through network.
- This communication can be realized by making the use of user programs by customizing it to the hardware that facilitates in transmission of the message.
- Customization of the user programs can be done by means of offering the service interface to operating system to realize communication among processes distributed system.
- In this way the communication service will be provided by operating system and user programs from will be free from taking care of communications.

-4 6. Error Detection

- In order to prevent the entire system from malfunctioning, the operating system continually keep watch on the system for detecting the errors.

- User programs kept free from such error detection to improve the performance.
- On the contrary, if this right would have been kept with user programs, most of the user program would have wasted time in error detection and actual work would be minimized resulting in performance degradation.
- Operating system needs to carry out complex tasks during the process of error detection.
- Such task comprises deal location of many resources such as processor, memory etc.
- If these tasks are again kept with user programs, then it may disturb the normal operation of operating system.

1.11 Objectives of Operating System

➔ (June 15, Nov. 15)

Q. Explain different objectives of operating system.
MU - June 2015. Nov 2015. 4 Marks

5

Following are the three objectives of Operating System,

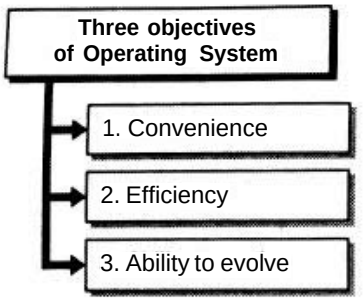


Fig. Cl .5 : Objectives of operating system

➔ **1. Convenience**

Computer system can be conveniently used due to operating system.

➔ **2. Efficiency**

- Computer system comprises of many resources.
- AH these resources are utilized by users application in efficient manner due to operating system.

➔ **3. Ability to evolve**

- The design of the operating system permits the efficient developmentjesting.
- It is also supports for flexibility by allowing for addition of new system fu netlons without interferin with service.

1.11.1 Functions of Operating System

➔ (Dec. 14, June 15, Nov. 15)

Q Exp lain different functions of OS?
MU ■Dec 2014. June 2015, No-. 2015. 3 Marks

- Operating system comprises different modules each of having its own collection of defined inputs and Qutp uts.
- These different modules or components of Oper... I system cany out specific tasks to offer the C0 M functionality of the operating system.
- The most important functions of the operating system are shown in Fig. C1.6.

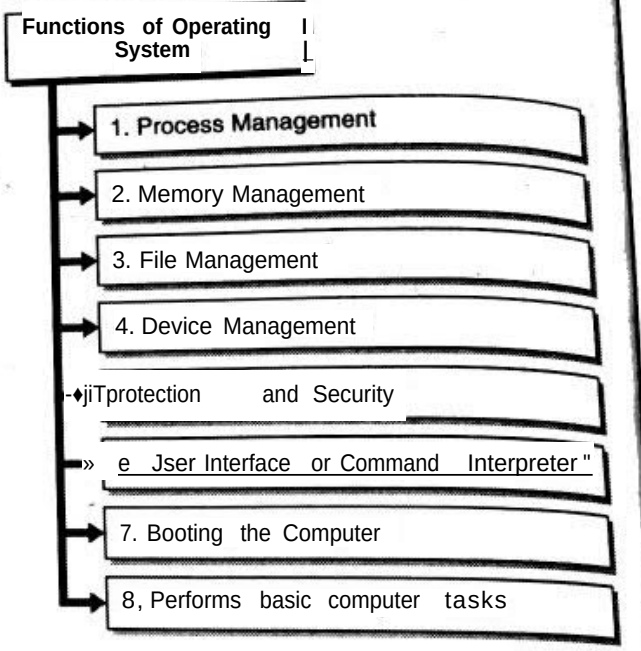


Fig. C1.6 : Functions of the operating system

➔ **1. Process Management**

The process management activities involves :

1. 1 o provide control access to shared resourceslib file, memory, VO and CPU.
2. Control the execution of user applications.
3. Creation, execution and deletion of user it system processes.
4. Resume a process execution or cancel it.
5. Scheduling of a process.
6. Synchronization, interposes communication deadlock handling for processes.

➔ **2. Memory Management**

Following memory management related function carried out by OS :

L It allocates the primary memory as well as secondary memory to the user and system processes.

2 Reclaim the allocated memory from all the processes that have finished its execution.

3 Once used block becomes free, OS allocates it again to the processes.

4 Monitoring and Keeping track of how much memory used by the process.

3. File Management

The file management activities of operating system consist of :

1 Files and directories are created and deleted by OS.

2 OS offer the service to access the files and also it allocates the storage space for files by using different methods of allocation,

3. It keeps back-up of files.

4. It offers the security for files.

4, Device Management

The device management tasks include :

1. Device drivers are opened, closed and written by OS.

2 Keep an eye on device driver. Communicate, control and monitor the device driver.

5. Protection and Security

- The resources of the system are protected by the operating system.
- In order to offer the needed protection, operating Sys X s use of user authentication, fde attributes such as read, write, encryption, and back-up o

6. User Interface or Command Interpreter

User interacts with computer system through operating between the user system. Hence OS act as an intent 2>d the computer hardware. Th.s user mterface o.fere through set of commands or a 8raph * ,"" nteraction (GUI)- Through this interface user with the applications and the machine har ware.

.4 7 Booting the Computer

- The process of starting or restarting the computer ts known as booting.
- If computer is switched off comp, Jm'booting is the A then it is called cold booting. to restart the process of using the operattng system computer.

8. Performs basic computer tasks

- The management of various peripheral devices such as the mouse, keyboard and printers is carried out by operating system.
- Today most of the operating systems are plug and play. These operating systems automatically recognize and configure the devices with no user interference.

Syllabus Topic : Operating-System Interface

1.12 User Operating-System Interface

Q. Write short note on user-OS interface.

User Interface

- User interface is essential and all operating systems provide it.
- Users either interface with the operating system through command-line interface OF graphical user interface or GUL
- Command interpreter executes next user-specified command.
- A GUI offers the user a mouse-based window and menu system as an interface.

Graphical User Interface or Command Interpreter

- User interacts with computer system through operating system. Hence OS act as an interface between the user and the computer hardware.
- This user interface offered through set of commands or a Graphical User Interface (GUI).
- Through this interface user makes interaction with the applications and the machine hardware.
- GUI stands for graphical user interface. « J permits use nf icons or other visual indicators to interact wit computer, rather than using only text via the command line.
- Due to GUI. it is not necessary to remember the commands. It provi • the ease ofuse user need to interaction with know any programming language for inter* computer.
- r 0 * OSL ure some of dleZraung nux, Apple os and systems that offer GUL

Syllabus Topic : System Calls

1.13 System Calls

➔ **(June 15, Nov. 15, May 16)**

q. What are system calls?
MU - June 2015, Nov. 2015. 4 Marks

O. Write short note on system calls.
MU - May 2016. 5 Marks

- The interface between OS and user programs is defined by the set of system calls that the operating system offers. System call is the call for the operating system to perform some task on behalf of the user's program.
- Therefore system calls make up the interface between processes and the operating system.
- The system calls are functions used in the kernel itself. From programmer's point of view, the system call is a normal C function call.
- Due to system call, the code is executed in the kernel so that there must be a mechanism to change the process mode from user mode to kernel mode.
- In the UNIX operating system, user applications do not have direct access to the computer hardware.
- Applications first request to the kernel to get hardware access and access to computer resources.
- During execution when application invokes a system call, it is interrupted and the system switches to kernel space.
- The kernel then saves the process execution context of interrupted process and determines purpose of the call.
- The kernel warily makes sure that the request is valid and that the process invoking the system calls has sufficient privilege.
- Some of the system calls can only be invoked by a user with super user privilege.
- If the whole thing is fine, the kernel -quest in kernel mode and can access The 7 the dnvers in charge of con troUing the hardwarc device
- The data of the calling process can . modified by kernel, as it has access to m Je read " d space , But - " not execute any codeT' 7 " user application, for clear security rcaS ons . * ""
- When the kernel finishes the ■ -stores the process execution * 881 *18 of request , h when the system call was invoked"'"' ,hat saved

to be calling program which

After this, control returns

persists executing.

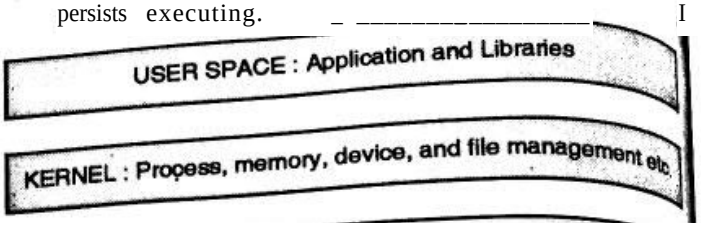


Fig. 1.13.1: The kernel

- As shown in Fig. 1-13-1 the kernel is a central module of most computer operating systems. Its main responsibilities are to shown in Fig. C1.7.

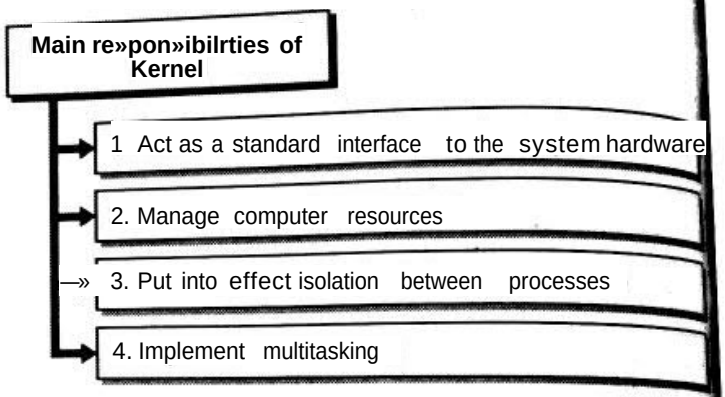


Fig. C1-7 : Responsibilities of kernel

➔ 1. Act as a standard interface to the system hardware

For example, while reading a file, application does not have to be aware of the hard-drive model or physi geometry as kernel provides abstraction layer to the hardware.

➔ 2. Manage computer resources

- As several users and programs shares machine and devices, access to those resources must be synchronized.
- The kernel implements and ensures a fair access as the dZiceT SUCh Pr0CeSS or, thC memOly and the

➔ 3. Put into effect isolation between processes

- The kernel guarantees ** process cannot corrupt the execution environment of another process.
- Any process cannot access the memory hat the kernel has allocated to other process

➔ * n ,*Jternen t multitasking

- Each PWcess gets the false i-

- Actually. several processes COD,pe *e cons,antl ,, f „ the kernel keeps switching the active system resources 1 process for each processor behind the scene

☞ Ways for the process to switch from the user mode to the kernel mode

There are two ways for the process to switch from the user mode to the kernel mode. These are:

- A user process can explicitly request to enter in kernel mode by issuing a system call.
- During the execution of user process kernel can take over to carry out some system housekeeping task.
- The kernel mode is both a software and hardware slate. Modem processors offer a advantaged execution mode, called as Supervisor Mode in which only kernel runs.
- The privileged operations are such as modifying special registers, disabling -interrupts, accessing memory management hardware or computer peripherals.
-]f it is not in supervisor mode, the processor will reject these operations.
- System calls take place in different ways, depending on the computer in use. Apart from the identity of the desired system call, more information is needed.
- The precise type and amount of information differ according to the particular operating system and call. Consider the example of getting the input.
- We may require specifying the source, which can be file or device.
- We also need to specify the address and length of the memory buffer into which the input should be read.
- The device or file and length may be implicit in the call.

☞ Ways ol parameter passing to operating system

Parameters can be passed to the operating system by following three different ways ;

1. Pass the parameters in registers.
2. If there are more parameters than registers, then the parameters are generally stored in a block, or table, in memory, and the address of the block is passed as a parameter in a register. This is the approach taken by Linux and Solaris.
3. Program can place the parameters onto the stack which then popped off the stack by the operating system.

Some operating systems favor the block or stack method, because those methods do not limit the number or length of parameters being passed.

Syllabus Topic : Types of System Calls

1.13.1 Types of System Calls

➔ (June 15, Nov. 15)

> Q. Explain any five system calls.
> MU - June 2015. Nov. 2015. 6 Marks

System calls are categorized in five groups. These are shown in Fig. C1.8.

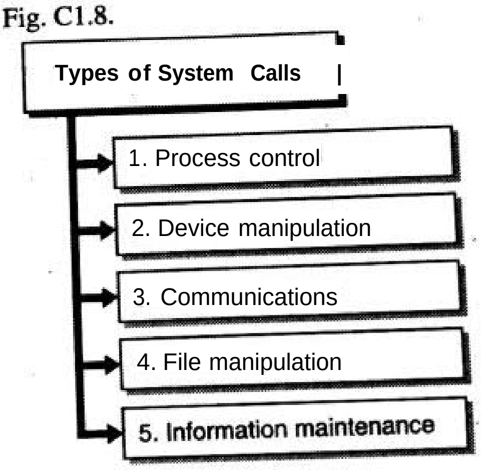


Fig. C1.8 : Types of system calls

Sr. No.	Group	Examples
1.	Process control	end, abort, load, execute, create process, terminate process, get process attributes, set process attributes, wait for time, wait event, signal event, allocate and free memory
2.	Device manipulation	request device, release device, read, write, reposition, get device attributes, set device attributes, logically attach or detach devices
3.	Communications	create, delete communication connection, send, receive messages, transfer status information, attach or detach remote devices.
4.	File manipulation	create file, delete file, open, close, read, write, reposition, get file attributes, set file attributes

Sr. No.	Group	Examples
5.	Information maintenance	get time or dale, set time or date, get system data. set system data, get process, file, or device attributes. set process, file, or device attributes

1.13.2 Some Examples of System Calls

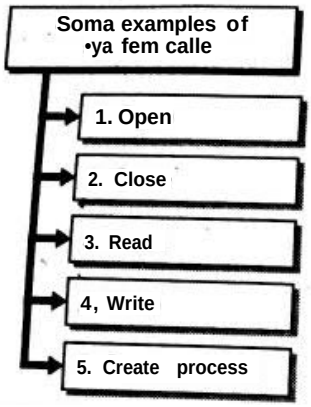


Fig. Cl.» : Examples of system calls

➜ **1. Open**

- Open system call request to the operating system for using a file.
- The files path is used as argument to specify the file,
- pie 'flags' and 'mode' arguments to this call specifies the way of using the file.
- On, successful approval by operating system, it returns a f‹,e descriptor" which is a positive integer.
- Returning of value-1 indicates denial of access to fi und 'eimo" needs to be checked to get reason o f denial"

'* **X Close**

- Close() informs to the operating system that file with said file descriptor is used.
- After this, same file descriptor can be reused by operating system.

➜ **3. Read**

- **ReadfJ** specifies to the OS the number of (siae y hvf fo read from the file opened in file descriptor "2 -, es
- h also specify 10 pu(the ytes into the locati °n Pointed to by W.
- The return value is number of bytes actually read.

➜ 4. Write

- Writef) **is similar to** read() system call, Only it writes **the bytes instead of reading them.**
- **It returns the number of bytes actually written** which is **almost invariably** "size".

➜ 5. Create process

- During the execution process can create new process L using create process system call.
- The process which creates new process is called the parent process, and the new processes are called the children of that process.
- Newly created processes may in turn create nt . processes, creating a tree of processes.

Syllabus Topic : System Programs

1.14 System Programs •

Q. Explain various system programs that are associated with operating system.

- System programs offer *a* suitable environment for development of application programs and its esecufo.
- Out of these system programs, a number of are on user interfaces to system calls and rest of the system programs are much more complex.
- Operating systems are supplied with these programs and are classified as below :

o **FH* management** t System programs in this category usually are responsible for rnanipulstit. of files and directories. These programs general), create, delete, copy, rename, print, dump and li the files.

o **Status information** : The status informationi lhe sys,em 1S for example: date, time, total fra memory or disk space in hand, number of used Present in the system, etc.

Some system programs simply inquire for tii information. Other gives the information regard"! complete performance, logging, and debugs S„ c programs have more complex d®' 1 Characteristically, these programs carries out formatting and printing oi output to the terminal

files or other output devices or display it in a window of the graphical user interface.

- **File modification :** System program such as text editor is used to create and modify data or information in files stored on disk or other storage devices. Some particular commands are used to look for contents of fifes or carry out conversions of the text.
- **programming-language support :** The language translators such as compilers, assemblers, interpreters come with operating system to translate the user programs to object code.
- **Program loading and execution :** After translation of the user program in object code it needs to be loaded into memory for carrying out the execution. The different types of loaders such as absolute loaders, relocatable loaders, linkage editors, and overlay loaders are offered by the system in order to load the program in memory.
- Systems required for debugging the higher-level languages or object code arc also offered.
- **Communications :** Some of the system programs offer the way for creating virtual connections among processes, users, and computer systems. They permit users to exchange messages, to browse web pages, to send electronic-mail messages, remote login and transferring the files from one machine to another.

- Just like the system programs, many applications too are the part of operating system.
- These programs are called as system utilities or **application programs.**
- It comprises the applications such as web browsers, word processors and text formatters, spreadsheets, database systems, compilers, plotting and statistical-analysis packages, and games.

1.14.1 Comparison between System Program and Application Program

Q. Compare system and application program.

Sr. No.	Parameters	System Prog ram/Sy stem Software	Application Program/Software
1.	Definition	System programs are designed to manage and control computer hardware to help the users program for efficient execution and to offer the services to them.	These software helps the user to carry out a specific tasks as per users requirement.
2.	Purpose	It is general-purpose.	Its purpose is specific.
3.	Environment	System software is itsen responsible to create environment to execute itself and other application programs.	Environment required for application program is created by system software.
4.	Responsibility	ft is responsible to control and manage entire computer system.	ft controls particular task for which it was intentionally designed.
5.	Execution	ft executes all the time when system is on to offer the services to user programs.	Its execution is as per the need of user.
6.	Examples	Operating system, compilers, loaders, text editors. assemblers are some of the examples of system prog...	Word processors or any software designed to perform specific task.

Syllabus Topic : Operating System Design and Implementation

1.15 Operating System Design and Implementation

There are many problems in the design and implementation of opending fl— ™ ere are certain solutions to these problems.

1.15.1 Design Goal*

Q. Explain design goals of oporadng system

- The initial problem that needs to be tacen is to define **goals and specifications.**
- Which hardware is considered and what type of system is decided to design will affect the design of the system.
- Types of system include the single or multiuser, real time, distributed, general purpose, interactive or batch system. This is the highest level of design.
- It is very difficult to identify the **requirements**. These requirements are categorized in two fundamental groups: ***user* goals and *system* goals.**
- Users always wishes the system which is convenient to use, simple to learn and to make use or u. reiwuis, safe and protected and having high performance.
- Due to lack of general agreement on how to accomplish these goals, the above properties are not useful in the design of system.
- Person responsible to design, create, maintain, and operate the system also desires the properties like: Simple to design, implement, and maintain, flexibility, reliability, error free, and efficient. Such requirements are unclear and may be inferred in different ways.
- Single solution to the problem of defining the requirements for an operating system does not exist.
- The different systems in existence show that different requirements can leads to a large diversity of solutions for different environments.

1.15.2 Separating Policies from Mechanisms

- MecW sms decide the way of doing something whereas policies determine what will be done.
- It is necessary to keep policies separate from mechanisms to achieve the flexibility
- Changes in policies occur over the period of time. A extreme level, each change in policy would need change in the underlying mechanism.
- A mechanism which does not affect by changes policy would be more preferred.
- If policy is changed then certain parameters of
- SyS 16 er example of the mechanism for J priority to certain types of jobs over others.
- nolicy and mechanism is do
- If "'Xel'y *en mechanism can be used to S P d Jon that I/O-bound jobs shou ld J priJy over CPU-bound jobs or to support the reve
- The essential set of P<>>i free P rimiti « bujj blocks is implemented by the operating system navi microkernel design.
- Consequently, it allows to add more adva mechanisms and policies by means of userJ kernel modules or by means of user p regr j themselves.
- The allocation of resources should be on the bad policy decision.

1,15.3 Implementation

q. Explain implementation of operating syst design.

- Traditionally assembly language was used implement the operating system.
- Now high level languages arc used to implement operating system.
- Due to this code is easy to read, understand and del and implementation process became faster.
- After recompilation operating system can be ported other machine.
- On the other side, if high level languages are use(implement the operating system the speed will reduced and storage required would increase.
- These issues are negligible as assembly W routines can be developed for large progrnins compilers can optimize the code.
- The operating system routines responsible bottleneck can be replaced by assembly W routines after system functions correctly.
- In this way performance can be improved. Useof* data structures and algorithms also impro es performance.
- Operating system should also support for monitori system performance.

- Bottlenecks can be identified by monitoring system pcrformance.
- n atinc system should include the code to calculate Xdisplay measures of system behaviour.
- In many systems, the operating system carry out this iob by producing trace listings of system behaviour.
- AU necessary events are logged with their time and important parameters are written to a file.

Syllabus Topic: Virtual Machines

1,16 Virtual Machines

q. What is virtual machine?

- By means of CPU scheduling and virtual machine (VM) techniques, host OS can produce illusion that each process has its own processor and memory.
- This memory is virtual memory. The VM offers an interface that is the similar to the underlying bare hardware.
- The processes running are the guest processes and each process gets virtual copy of the underlying computer. This guest process is a operating system.
- In this way single machine runs the multiple operating systems in parallel, each in its own VM,
- From availability and security point of view, using separate computer to put each service is more preferable by organizations. If one server fails, other will not be affected.
- It is also beneficial in case if organizations need to use different types of operating system.
- However, keeping separate machines for each service is costly. Virtual machine technology can solve this problem. Although this technology seems to be modem but idea behind it is old.
- The VMM (Virtual Machine Monitor) creates the illusion of multiple (virtual) machines on the same physical hardware.
- A VMM is also called as a hypervisor. Using virtual machine, it is possible to run legacy applications on operating systems no longer supported or which do not work on current hardware.
- Virtual machines permits to run at the same time applications that use different operating systems.
- Several operating systems can run on single machine without installing them in separate partitions. Virtualization technology plays major role in cloud computing.

1,16.1 History

- Internet standards virtual machines were also present earlier. Around I960, IBM developed two hypervisors SIMMON and CP-40 which was research project.
- CP-67 was reimplemented version of CP-40. CP-67 formed the CP/CMS which was virtual machine OS for the IBM System/360 Model 67. In 1972, CP-67's reimplemented version was launched as VM/370 for the System/370 series.
- In 1990 IBM replaced System/370 by System/390. In all this journey, although machines were renamed but underlying architecture remained unchanged to maintain backward compatibility.
- There was better improvement in hardware technology and newer machines were bigger and faster as well.
- Over the period of time, although hardware technology improved, but virtualization was there and supported by all the machines.
- In 2000, IBM released the . z-series which was backward compatible with the System/360. Z-series had supported 64 -bit virtual address spaces.
- All of these systems supported virtualization decades before it became popular on the x86.
- The early releases of OS/360 were strictly batch systems. But, due to requirement, many 360 users, decided to have timesharing, so various groups, both inside and outside IBM decided to write timesharing systems for it.
- The TSS/360, the first time sharing system, arrived late and it was so big and slow that few sites converted to it. It was finally neglected. And it its development cost was very high around some $50 million (Graham, 1970).
- But a group at IBM's Scientific Center in Cambridge, Massachusetts, produced a radically different system that IBM eventually accepted as a product, and which is now widely used on its remaining mainframes.
- This system, originally called CP/CMS and later renamed VM/370 (Seawright and MacKinnon, 1979), was based on an astute observation :
  - o A timesharing system provides
  - o Multiprogramming and
  - o An extended machine with a more convenient interface than the bare hardware.
- The essence of VM/370 is to completely separate these two functions.

- The heart of the system, known as the virtu-1 machine monitor, runs on the h-rdwam multiprogramming, providing not one, but several virtual machines to y up, ds , Fig. 1.16.1,

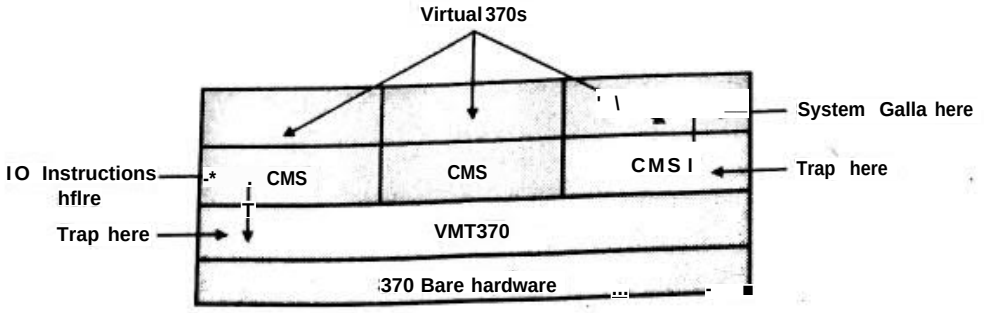


Fig. 1.16.1

- However, unlike all other operating systems, these virtual machines are not extended machines, with files and other nice features.
- Instead, they are *exact* copies of the bare hardware, including kemel/user mode, I/O, interrupts, and everything else the real machine has.
- To maintain the backward compatibility, the defects in Intel 386 were automatically carried forward into new CPUs for 20 years.
- Hence virtualization has been a problem on x86 architecture.
- The same instructions behave differently when executed in kernel mode with compare to their execution in user mode, for example, instructions carrying out I/O and changing MMU settings.
- All these instructions are called as sensitive instructions. Some other instructions cause trap if run in user mode called as privileged instructions.
- If sensitive instructions are subset of privileged instructions then machine is virtualizable.
- In 2005, Intel and AMD added virtualization in their CPUs. On the Intel CPUs it is cal ied **VT (Virtualization Technology);** on the AMD CPUs it is called **SVM (Secure Virtual Machine).**
- **Containers are created in order to run virtual machines in it. When a guest operating system is initiated in a** container, it carry on running there unti) it causes exception and traps to the hypervisor, for example by executing an I/O instruction.
- The set of operations that trap is controlled by a hardware bitmap set by the hypervisor.

1.16.2 Benefits

Q. What are the benefits of virtual machine?

Following are the benefits of creating virtual machines

- Several virtual machines are protected from each other and also host system is protected from virtual machines.
- **If guest operating system contains virus then it** damage this operating system but not other operating system. Also host operating system is affected by this virus.
- The direct sharing of resources is not present Thsfi sharing is possible.
- The network of VMs is possible where infamfc sharing can be done over virtual commu , network. The research and development in operal system is possible due to virtual machine concept.
- System remains unavailable till changes made rr tested in operating system.
- lime required for this is system development time.: case of virtual machine, system development isdone virtual machine instead of on a physical machine.
- The virtualized workstation permits for quick potM and testing of programs in changeable enviromnatt
- , .he same Way* Quality-assurance engineers can' r* ir appJications in several environments with ouymg, powering. and maintaining a computer fot environment.
- VM°hy combining two separate physical machine

- In this manner, two lightly loaded system can be converted in one heavily loaded system.
- For extensive acceptance, the design of virtual machines must be standardized with the intention that any virtual machine will run on any virtualization Platform.
- The "Open Virtual Machine Format" can be realized to be successful in combining virtual-machine formats.

1,16.3 Simulation

- Because of virtualization, guest OS and applications believes to be executing on native hardware.
- Simulation is other system emulation methodology like virtualization in which host system has one architecture and compilation of guest system was carried out on other architecture.
- This is just like running instructions on new computer system that were compiled on the old computer system.
- The programs could be executed in an elmulalor that converts each of the old systems instructions into the native instruction set of the new computer system.

1,16.4 Para-Virtualization

- Para-virtualization offers to the guest similar but not identical preferred system of the guest.
- 11 is necessary to mooiiy the guest to run on Para-virtualized hardware.
- Due to this resources can be utilized more efficiently. A virtualization layer is required. Solans operating system contains zones or contamers produce virtual layer between OS and appheattons.
- In this case, hardware virtualization is not done but OS and associated devices are virtualized so that processes within containers believe that only they arc on the system.
- Containers can be more than one each having its own applications, network stacks, network address ports user accounts, and so on.
- CPU resources are partitioned among the containers and system wide processes.

1 16.5 Implementation

- The implementation of virtual machine (VM) concept is difficult.
- The creation of exact copy of underlying machine needs most of the work,
- This underlying machine has user mode and kernel mode, Since VM software is OS, it can run in kernel mode.
- The VM itself can run in user mode. It is necessary to have virtual user mode and virtual kernel mode.
- Both of these modes must run in physical user mode. Just like user mode to kernel mode transition on real machine, there should be transition from virtual user mode to virtual kernel mode on VM.

This transfer can be carried out as follows :

- System call by program running on VM in virtual user mode causes transfer to the VM monitor in the real machine.
- VM monitor then alter the register content and program counter for VM to simulate the effect of system call.
- The VM is the restarted which is now in virtual kernel mode. Virtual I/O may take less or more time than real I/O. As CPU is being multiprogrammed among many VM*s, they can be further slows down. For virtualization, hardware support is also important. All general purpose CPU supports for virtualization.

1.16.6 Examples

Following two, VMware workstation and java virtual machine are the examples of VMs.

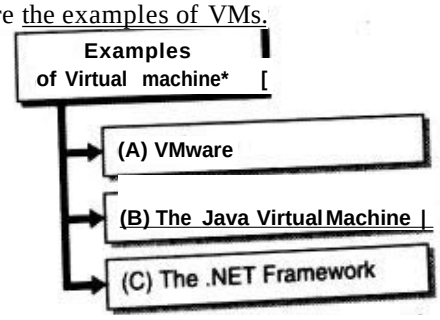


Fig. CLIO : Examples of virtual machine

-> 1.16.6(A) VMware

Explain VMware in detail.

- VMware workstat.cn is a establ. he X86 and
- VMware Workstation tuns as an application on a host OS Examples of host OS are Widows or Linux.

- This permits this host system to run a number of different guest operating systems in parallel as independent virtual machines.

Appikalfon

- In Fig, 1.162. Linux is running as host OS. Three operating systems, free BSD, Windows NT. and Windows XP are running a-s guest operating systems.

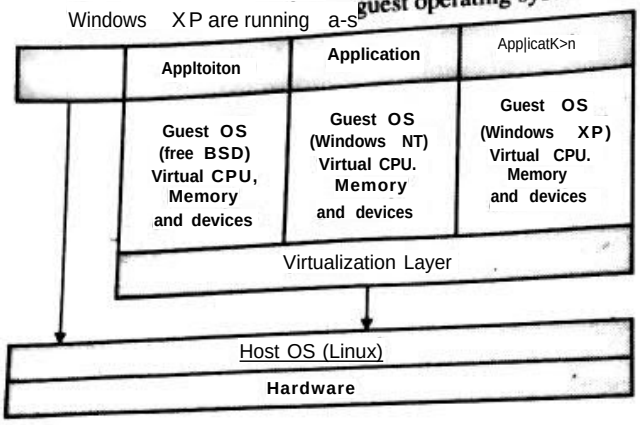


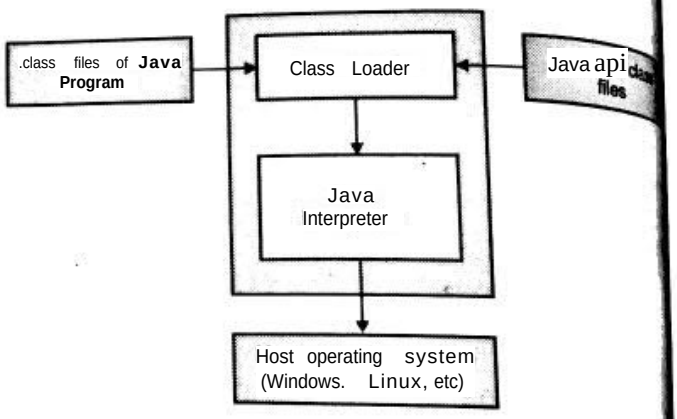
Fig. 1.16.2 : VMware Architecture

- The virtualization layer offer abstraction of physical hardware into isolated VMs which are running as guest operating systems.
- As shown in Fig. 1.16.2, each VM has its own virtual memory, CPLF,devices and network interfaces etc.
- The guest file is the copy of host OS file in file system.
- The guest instance of file is separate copied file due to which protection of guest instant against disaster is achieved.

-> 1.16.6(B) The Java Virtual Machine

0, Write short note on JVM.

- 1be Java is object oriented programming language in which program contains one or more classes.
- The compiler produces architecture independent bytecode for each class file.
- This bytecode can run on any implementation of the java virtual machine.
- The JVM has class loader and java interpreter. The java interpreter runs the architecture independent bytecodes
- The compiled .class files are loaded by class loader from both the Java program and the Java API which are then executed by java interpreter.
- Once the loading of the class is done It k ' k < verifier to check .class file whether it is ' J1* " ' b ode anddoes..... alld Java erflow or underflow the stack.
- It also makes sure that the bytecode does not carry out pointer arithmetic, w ic h could lead to illegal memory access.
- his verification is successful then it is run by the Java interpreter- The JVM also proves garbage collection automatically-
- The implementation of JVM *can* be carried out as software on the top of host OS or as a pan of web browser.
- The interpreter interprets bytecode one at a M me - faster interpretation is achieved by just in A compiler. t)
- The JIT first converts bytecode in machine instructions and subsequent invocation of the methods a,c carried out using these cached machine instruction.
- It avoids bytecode interpretation in su)bsequent invocation.
- The hardware implementation of JVM can be carried out in chip which is faster than software implementation.

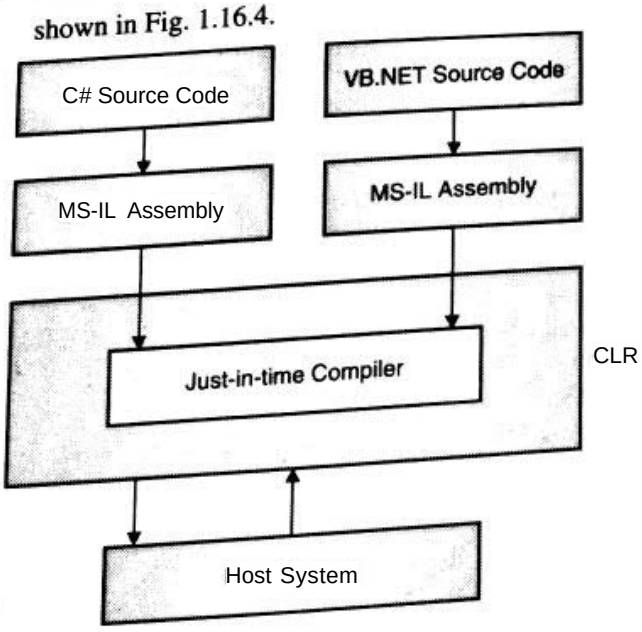


Fig, 1.163 : The java virtual machine

➜ 1.16.6(C) The .NET Framework

- The .NET Framework includes set of class libraries an executing environment which creates platform developing the software.
- A program written for .NET framework does require specific hardware or operating system.
- The execution of the program can be carried successfully by any architecture which imp .NET.
- The de squired for this execution is abstra execution environment and VM is provided ac J as intimidate between execution environ un derlymg architecture.

- The NET framework contains common language runtime (CLR) which is implementation of .NET VM.
- The programs are written in C# and VB.NET and compiled into Microsoft Intermediate Language (MS-IL) which is intermediate architecture independent language.
- The compiled files (assemblies) having extensions as , EaC nr DLL contain MS LL instructions and metadata.
- The CLR loads assemblies into the Applicatton Domain upon program execution.
- As executing program requires instructions, the CLR translates the MS-IL instructions inside the assemblies into native code that is precise to the underlying architecture using JIT compilation.
- The instructions then will carry on running as native code for the CPU.
- The architecture of the CLR for the .NET fmmework is shown in Fig. 1.16.4.



. CLR architecture for .NET framework

Fig. 1.16.4

ng

- yTbus TOPIC: OperaTigjB S

1.17 operating SystemDguggl

Q. Explain OS debugging in detail.

Debugging includes finding and fixing the errors in ed out for both hardware system. Debugging activity is cam performance can and software. By fixing the bugs tn system, pe be improved,

1,17.1 Failure Analysis

- Operating system writes error information in log file or takes core dump in case of failure of process.
- The core image is maintained in file for afterward analysis.
- Debugger tool probes the executing programs or core dump. It is challenge to debug the user level process.
- As kernel is more complex and having large size, its debugging is complex task as user level debugging tools are not available. When kernel crashes, its memory state is saved to crash dump.
- The tools for OS debugging are different than for process debugging.
- If failure occurs in file system code, then it is not possible for kernel to write log in file in file system before rebooting. In this case, memory slate is saved on the disk area where file system is not present.
- After rebooting, this information is the gathered and saved in file of file system for later analysis.

1.17.2 Performance Tuning

- It is necessary to m<nitor the system performance to discover bottlenecks.
- It is also important to add the code for computing and displaying the measures of system behavior.
- Operating system produces the fisting °f *system* behavior, A log is ms intained for an req with thei r time and important parameters.
- This log is then written to a file. Afterward, processing is carried an analysis program to know system performan' and to recognize bottlenecks and inefficten cies. recommended improved system of a
- The simulation can be carried out with these same traces as input Traces also can facilitate people to locate errors in operating-system behavior.
- In other method of performance tuning, a system can have interactive tools which permits the users and administrators to ask the different components state of the system to find bottlenecks.

1 17.3 DTrace

- DTrace ts the 1 ? pr nroce sses and kernel. the facility that permits to add probes dynamically in running user p-

- By giving query™™--; ...es using D program™ g about the kernel, -be language. surpd«"« am irjc5 can he determined, sysiem sure, and proves.
- It is impossible to debug interaction between user level "*""TM' CTdeWi "'rbo-h the codes and
- These (ootserN understands ta mea. the rarera ctions.
- These toolset should * aH e to debug all the area of the system andshoaMa. :t the J av yim t reliability.
- It the tool is not in use then shou ... impact as per use.
- ZZdl -se requirement, and oner safe, dynamic, low imp«t debugging environment.

'Syllabm Topic : Oporatlng Systom GeneroUon

1.18 Operating System Generation

[g, Explain operating system generation. J

- Operating system can be designed for different machines at diversity of sites with varied peripheral configuration.
- System then configured for particular computer site called as system generation (SYSGEN).
- The distribution of OS is usually done on disk, on CD-ROM or DVD-ROM. or as an "ISO" image.
- This distributed OS is a file tn the format of a CD-ROM or DVD-ROM.
- The SYSGEN program reads from a given file, or request the operator of the system for information relating to the exact configuration of the hardware system, or query the hardware straight way to know about components present there.
- The following information is needed to determine.
  - o The CPU is to be used. Options installed like ...T 2 nSttUC,i "« floatin 8 Point arithmetic etc. Description of each CPU in case of multiple CPU system.
  - o The way of fu..., auullK the boot disk. The number of sections, or "partitions " in t» and what will go into each partition? " into,
  - o The totaJ men 'ory available. This is d L done uy mferencmg the ntentoiy j ocations t:' address is generated dJegaJ
  - o The available devices, their device number, type and model of device mteX'pt number and characteristics.
  - The preferred OS options, parameters values
  - o used The number of buffers and size to by values, the required CPU schedulmg a gorithm, rhe maximum number of proce SSes to be supported and so on.
- A system administrator can use this information to modify os source code.
- The compilation of modified OS is then carried out to produce output-object version as per system description.
- Io another approach, tables are created and modules are selected from precompiled library.
- After linking the modules, generated OS is formed.
- next approach, complete table driven system Cj# be created.
- Code selection is at execution time rather than compile or link time*

Syllabus Topic : System Boot

1.19 System Boot

Q. Explain booting process of the system in detail

- Unless the operating system is available in memory use by hardware computer cannot be ready to use
- The process of starting a computer by loading the kernel in main memory is known as *booting* the system.
- A small program called as the **bootstrap program** < **bootstrap loader** finds the kernel, loads it into main memory, and starts its execution.
- On some computer systems, for example PCs, follow two-step process.
- In first step simple bootstrap loader obtains a « implex boot program from disk, and this complex Program then loads the kernel.
- After a CPU receives a reset event, for example, loaded Up Or rebooled , the instruction regi wecution starts the defl,le<1 memOty l0Ca,i0 ...
PragrJ u inRe ea Lf° ," P pragram resides. <"'Only Memorv fROMl.

- When system starts up the RAM initially is in unknown state. ROM is suitable because it needs no initialization and cannot be infected by a computer virus.
- The bootstrap program runs diagnostics to decide the state of the machine. If the diagnostics pass, the program can carry on with the booting steps.
- It also initializes all parts of the system, from CPU registers to device controllers and the contents of main memory.
- Il then starts the operating system. Several systems like cellular phones, PDAs, and game consoles store the complete operating system in ROM.
- If the operating systems are small in size, storing it in ROM is appropriate for, simple supporting hardware, and rugged operation.
- A difficulty with this approach is that changing the bootstrap code needs changing the ROM hardware chips.
- This problem is solved by some system using **Erasable Programmable Read-Only Memory (EPROM).**
- EPROM is read only apart from when clearly given a command to become writable.
- The characteristics of ROM go down somewhere between those of hardware and tho.se of software. Therefore all forms of ROM are also known as **firmware.**
- For large size operating systems like Windows. Mac OS X and UNIX or for systems that change often, the bootstrap loader is stored in ROM. and the operating system is on disk.
- In this case, the bootstrap runs diagnostics and has a small piece of code that can read a single block at a fixed location (say block zero) from disk into memory and execute the code from that boot block.

1.20 ExamPack (University and Review Questions)—

<r Syllabus Topic : Introduction

q. What is operating system?

(Refer section 1.1){2 Marks)

(Dec. 2014, June 2015, Nov. 2015)

or **Syllabus Topic : Operating System Structure**

q Explain monolithic system. *(Refer section 1.2. 1)*

Q. Explain layered system in detail. *(Refer section 1.2.2)*

Q. Explain client-server model in detail. *(Refer section 1.2,3)*

Q. Differentiate between monolithic and microkernel. *(Refer section 1.2.4)* (5 Marks)

(June 2015, Nov 2015)

Q. What is Kernel ? Describe briefly the approaches of designing Kernel. *(Refer section 1.2.4)* (5 Marks) (Dec. 2016)

q. Differentiate between monolithic and microkernel. *(Refer section 1.2.4)* (5 Marks)

(June 2015, Nov. 2015)

☛ **Syllabus Topic : Operating System Operations**

q. Explain operating system operations. *(Refer section 1.3)*

tr **Syllabus Topic : Process Management**

Q. Explain process management function of operating system. *(Refer section 1.4)*

☛ **Syllabus Topic : Memory Management**

q. Explain memory management function of operating system. *(Refer section 1.5)*

& Syllabus Topic : Storage Management

Q. Explain file management in OS. *(Refer section 1.6.1)*

Q. Write note on mass storage management. *(Refer section 1.6.2)*

Q. Explain how caching Is implemented ? *(Refer section 1.6.3)*

ur **Syllabus Topic : Protection and Security**

Q. Explain protection and security mechanism of OS. *(Refer section 1.7)*

cF' **Syllabus Topic : Distributed System**

Q. What are the characteristics of distributed system? Explain. *(Refer section 1.8)*

☛ **Syllabus Topic : Special Purpose Systems**

Q. Explain various special purpose systems. *(Refer section 1.9)*

ffrrwetinoSyslemfW:

Q. write note on handheld syrrfe-™. *(Refer section 1.9.3)*

☞ SyH .bu. **Topic: operating SX' Wn , t Services**

Q. Explain services provided by < operating system, **(May 2016 >** *(Refer section 1.10)* (5 Marks)

Q. Explain different objectives of operating system. *(Refer section 1.11)* **11)** (4 Marks) **(June 2015, Nov 2015)**

Q. Explain dinereni functions of OS? *(Refer section 1.11.1)* (3 Marks) **(Dec 2014, June 2015, Nov. 2015)**

☞ **syllabus Topic : Operating-System Interface**

Q. Write short note on user-OS interface. *(Refer section 1.12)*

☞ **Syllabus Topic : System Calls**

Q. What are system calls ? *(Refer section 1.13)* (4 Marks) **(June 2015, Nov. 2015)**

Q. Write short note on system calls. *(Refer section 1.13)* (5 Marks) ("ay 201 6)

☞ **Syllabus Topic : Types of System Calls**

Q. Explain any *five* system calls. *(Refer section 1.13.1)* (6 Marks) **(June 2015, Nov. 2015)**

☞ **Syllabus Topic : System Programs**

Q. Explain various system programs that are associated with operating system. *(Refer section 1.14)*

Q. Comp are system arid application program. *(Refer section 1.14.1)*

☞ **Syllabus Topic : Operating System Design Implementation**

a. Explain design goals Of Operating system. *(tefer section 1.15.1)*

Q. Explain implementation of operating system design *(Refer section 1.15.3)*

☞ **Syllabus Topic : Virtual Machines**

Q. What is virtual machine? *(Refer section 1.16)*

Q. What are the benefits of virtual machine? *(Refer section 1.16.2)*

Q. Explain VMware in detail. *(Refer section 1.16.6(A)*

Q. Write short note on JVM. *(Refer section 1,16.8)*

☞ **Syllabus Topic : Operating System Debugging**

Q. Explain OS debugging in detail. *(Refer section 1.17)*

☞ **Syllabus Topic : Operating System General**

Q. Explain operating system generation. *(Refer section 1.18)*

☞ **Syllabus Topic : System Boot**

Q. Explain booting process of the system in detail *(Refer section 1.19)*

Module II

CHAPTER 2

Process Management

Syllabus Topics

Process concept : Process Scheduling, Operation on process and Interprocess communication; Multithreading, Process : Multithreading models and thread libraries, threading issues; Process Scheduling; Basic concepts, Scheduling algorithms and Criteria, Thread Scheduling and Multiple Processor Scheduling,

2.1 Introduction

- Process manager implements the process management functions. In multiprogramming, single CPU is shared among many processes. If many processes remain busy in completing I/O, CPU is allocated to only one process at a given point of time.
- Here some policy is required to allocate CPU to process, called as CPU scheduling. If multiple users are working on the system, operating system switches the CPU from one user process to other.
- User gets the illusion that only he or she is using the system. Process synchronization mechanism is required to ensure that only one process should use critical section.
- Process communication, deadlock handling, suspension and resumption of processes and creation and deletion of the processes etc are some of the activities performed in process management.

Syllabus Topic : Process Concept

2.2 Process Concept

➔ (Dec. 14)

q. What do you mean by process?

MU - Dec. 2014. 3 Marks

Q. Explain the concept of process.

- A program or application under execution is called as process. A process includes the execution context. A program resides on the disk. On disk it does not require any resources.
- A program gets executed in main memory. So it should be transferred from disk to main memory.
- To complete execution, program needs many resources and competes for it.
- Now it becomes process. From the computation context point of view, a process is defined by CPU state, memory contents and execution environment.
- A CPU state is defined by the content of the various registers such as Instruction Register (IR), Program Counter (PC), Stack Pointer (SP) and general purpose registers.
- A small amount of data is stored in CPU registers. Memory contains program code and its predefined data structures.
- Heap is reserved memory area for dynamically allocation of memory to the program at run time.
- In stack, program local variables are allocated and return values of function call are stored. Some register values are also saved in stack.
- Execution environment includes open files, communication channels to other processes etc. following are the components ol the process :
  - o The object code that is to be executed.
  - o Resources required by the program to complete the execution.
  - o The data on which program will operate.
  - o Program execution state.

2.3 Context Switch

Q. What is context switch?

- When CPU switches from one process to other, a context switch occurs. It is the switching of the CPU (Central Processing Unit) from one process or thread to another.
- When context switch occurs, the operating system restores the information of currently executing process and allocates the same later used. When CPU starts executing the process, the restored information is used to resume the process.
- The context of a process is represented in the Process Control Block (PCB) of a process.
- The information needs to be restored includes address of the next instruction to be executed (program counter), CPU register contents, pointers to the memory allocated to the process, schedule accounting information, changed state, I/O state information etc.
- While context switch, system does not perform any useful work. So context switch is pure overhead on the system.
- The speed of context switching depends on the number of registers that must be copied, memory speed, etc. context switch speed varies from system to system.

Syllabus Topic : Operation on Processes

2.4 Operations on Processes

Q. What are the operations performed on process?

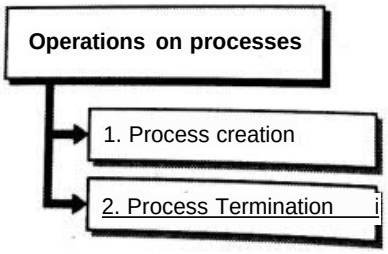


Fig. C2.1 : Operations on processes

2.4.1 Process Creation

Q. Explain process creation.

Processes are created because of the following principal events. These are shown in Fig. C2.2

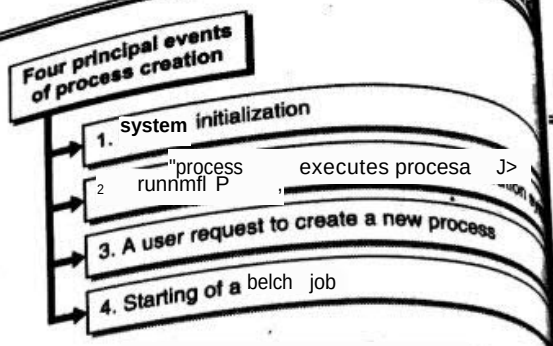


Fig. C2.2 : Events of process creation

1. System initialization

- After booting OS many processes are created.
- If running process executes process creation system call
- Currently executing process creates processes by issuing create process system call needs these processes to assist it in its execution.
- Creating new processes is mainly helpful when work to be carried out can easily be formulated in terms of several related, but otherwise independent interacting processes.

3. A user request to create a new process

- In interactive systems, users can start a program by typing a command or (double) clicking an icon.

4. Starting of a batch job

- In this case users can submit batch jobs to the remote system. When the OS makes a decision that it has the resources to execute another job, it creates a new process to execute the next job from the input queue in it.

During the execution process can create new processes using *create process* system call.

- The process which creates new process is called **parent** process, and the new processes are called **children** of that process.
- Newly created processes may in turn create other processes, creating a **tree** of processes.
- In UNIX or the Windows family of operating systems Processes are identified by unique process identifier (or **pid**), which is typically an integer number. The command is used in UNIX to obtain the process identifiers.

- By using —mmand ps — -vaiipicie information for al] processes currently active in the syst era can be obtained.
- Process requires certain resources like CPU time, memory, files, I/O devices io complete its task.
- The subprocess also needs these resources. Subprocess can get its needed resources directly from the operating system, or it may be forced to a subset of the resources of the parent process.
- The parent can share the resources among its children or it can divide the resources to allocate to its children.
- If child process is restricted to a subset of the parent's resources overloading the system by creating too many subprocesses can be avoided.
- The initialization data may be passed along by the parent process to the child process after creation of the process.

After creation of a new process, two possibilities exist related to execution :

— The parent and its children both executes in parallel.

- Until few or all the children terminates, parent will wait and will not be terminated.

With respect to the address space of the new process again two possibilities exist :

- Program and data of child and parent process is same. It means child is duplicate of its parent.

— Program and data of child and parent process is different. It means child process has a new program loaded into it.

➜ 2.4.2 Process Termination

Q, Explain process termination. J

- When process finishes the execution of last statement, it terminates. After this it request the operating system to delete it by using the *exit ()* system call. Just then, the process may return an integer to its parent process by using *wait()* system call.

-- Then all allocated resources to the process like physical and vl&al memory, open files, and VO buffers are freed by the operating system. Termination can happen in other situation also.

- By executing the suitable system call, any process can terminate the other process. Normally, parent of t e process to be terminated, executes this system call.

- OtteTwise, users could randomly kill Mch others jobs. -- is necessary that a parent requires knowing the idenlitifec of its children.
- Thus, when a process creates a new process, the identity of the child process is passed to the parent.
- Any of the child processes execution can be terminated by child for a diversity of reasons, such as these :
  - ° If the allocated resource usage is exceeded by child. (The parent must have a means to examine the state of its children.)
  - o There is no longer requirement of allocated task to the child.
  - o In many systems, If the parent is terminated then OS does not permit a child to carry on execution.
- For example, VMS system does not permit a child to carry on execution after its parent process is terminated.
- If all children of particular parent are terminated then it is called as cascading termination. After normal or abnormal termination of parent, cascaded termination is carried out in much system. It is initiated by the OS.
- In UNIX, exit() system call is used to terminate the process. The wait() system call is used by parent to wait until child is terminated.
- The terminated child's process identifier is returned by the wait() system call. Because of this process identifier, parent process comes to know about which child is terminated.
- After the termination of the parent, the rnif process is assigned as parent to all terminated children.
- This *init* process collects all terminated children's status and execution statistics after termination of their parent.

2.5 Process Control Block

➜ (Dec. 14)

Q. Explain role of process control block. MU - Dec. 2014. 5 Marks

Q. Explain process control block. J

- Any process is identified by its process control block (PCB). PCB is the data structure used by the operating system to keep track on the processes.
- All the information associated with process is kept m process control block. There is separate PCB for each process.

module keeps track c the status of the processes. same state PCB's of process; etc) are linked ready list. (ready, executing. such as giving a specific name in chain Of all these pn* cks the PCB chain to see if any device.

If device become. process i* w.i>i"8 fOf ,he

If the processes are available in that device chain, then back to ready state to request that device again

A typica J processed!' is shown in rifr 2 5.1.

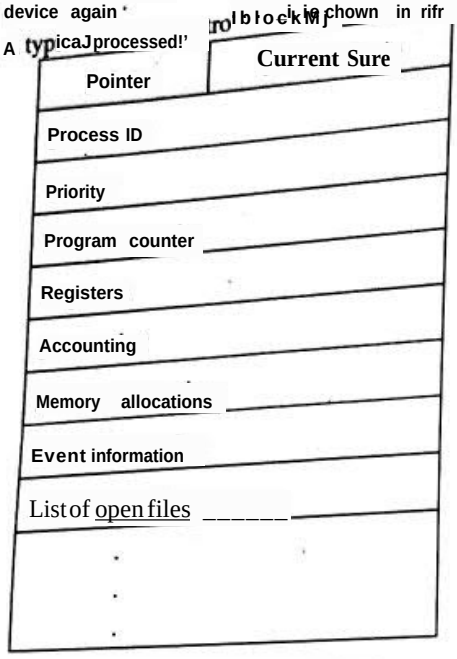


Fig. 15.1 : Process control block

»■ Pointer

This field points to other process's PCB. The scheduling list is maintained by pointer.

Current state

Currently process can be in any of the state from new, ready, executing, waiting etc as described above.

ProcessID

Identification number of the process. Operating system assign this number to process to distinguish it from other processes.

Priority

Different process can have different priority. Priority field indicate the priority of the process.

ram counter

CPU is given to other proc

After c reviously executing proc«Ss

When dlocated to it- Program instruction

ntext switch.

Refll

purpose register, index register

It inclu* gene m ulators etc. number of regis stack pointers and and type of register differs as per the architecture computer. The state information stored after interrupt again used by the process to resume the execution.

AccoU nt fl ca)culating the process's priority Information™- relative to othe Processes. information

This may me ude accounting abou It also inc i u des amount of CPU mbers "d "rS "me used, time limits, process „u and so on.

Memory allocation

This information may incMe the value of baM and Lt register. It includes paging. segmentation re lated information depending on the memory system used Address space allocated to the process etc.

Event information

For a process in the blocked state this field contain information concerning the event for which the proces is waiting.

List of open files

Files opened by the process.

After creation of the process, hardware registers and flags are set as per the details supplied by loaders or linker Al any time the process is blocked, the processor registers content are generally placed on the stack and the pointer to the respective stack frame is stored in the PCB. In ihh fashion, the hardware state can be restored when the process is scheduled and resume execution again.

2.6 Process States and Process State Transition Diagram

Q. Draw and explain process state transition diagraml

During execution, process changes its state. Pro® state contains five states: activity of the process. Pro®

– Each process remains in one of these five states. There is a queue associated with each state of the process.

– Process resides on that queue as per the state in which it resides. The states are listed in Fig. C2.3.

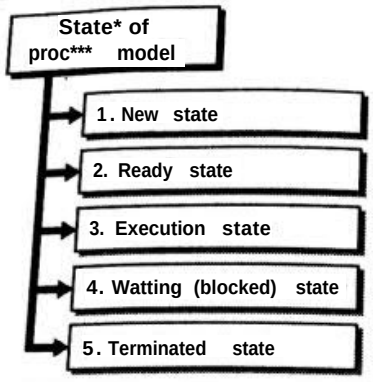


Fig. C23 : States of Process Model

-> 1. New state

The new process being created.

-> 2. Ready state

A process is ready to run but it is waiting for CPU being assigned to it.

*♦ 3. Executing state

A process is said to be in running state if currently CPU is allocated to it and it is executing.

■4 4. Waiting (blocked) state

A process can't continue the execution because it is waiting for event to happen such as I/O completion. Process is able to run when some external event happens.

➥ 5. Terminated state

The process has completed execution.

The process state transition diagram is shown in Fig. 2.6.1.

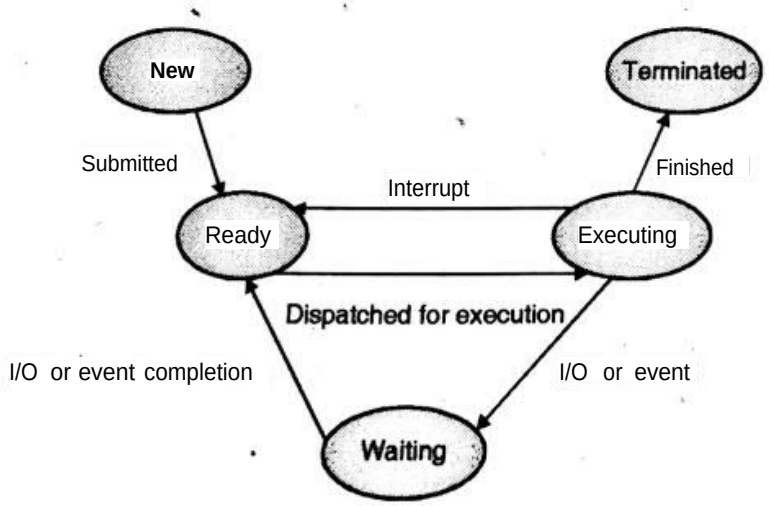


Fig. 2.6.1 : Process state transition diagram

- When the process is created, it remains in new state. After the process admitted for execution, it goes in ready state.

- A process in this state, wait in the ready queue. Scheduler dispatches the ready process for execution i.e. CPU is now allocated to the process.

~ When CPU is executing the process, it is in executing state. After context switch, process goes from executing to ready state.

– If executing process initiates an I/O operation before its allotted time expires, the executing process voluntarily give up the CPU.

- In this case process transit from executing to waiting state. When the external event for which a process was waiting happens, process transit from waiting to ready state.

- When process finishes the execution, it transit to terminated state.

2.7 Process vs. Thread

– A thread is a single sequence stream within a process. Threads are also called as lightweight processes as it possess some of the properties of processes. Each thread belongs to exactly one process.

- In operating system that support multithreading, process can consist of many threads.

- These threads run in parallel improving the application performance. Each such thread has its own CPU state and stack, blit they share the address space of the process and the environment.

- Threads can share common data so they do not need to use interprocess communication.

– Like the processes, threads also have states like ready, executing, blocked etc. priority can be assigned to the threads just like process and highest priority thread is scheduled first.

- Each thread has its own Thread Control Block (TCB). Like process context switch occurs for the thread and register contents are saved in TCB.

- As threads share the same address space and resources, synchronization is also required for the various activities of the thread.

o- Difference between thread and process

Sr. No*	Parameters	process	Thread
1.	Definition	Program in «« ion called as PrOCeSS , It is known as heavy weight process.	Thread is P* 1 Of prOCeSS , h « ''so called flight wtigiPP rocess
2.	Context switch	Process context switch compared to thread conte... . "Lcuse it needs interface of operating system.	Thread context switch takes less time as compared W Process cont ext switch because it needs only interrupt to kernel only-
3.	Creation	New Process creation nation takes more time as compared to newthread creatio j	New thread creation takes less time as compared to new process creation.
4.	Termination	New Process termination takes more tune as compared to new thread termination.	New thread termination takes less time as compared to new process termination.
5.	Execution	Each proceTexecutes the same code but has its own memory and file resources.	All threads can share same set of open files, child processes.
6.	implementation	If implementation is process based, then blocking of one process cause the blocking of other server process until the first process unblocked and these are not allowed to execute until blocked process is unblocked.	In multithreaded server implementaijon, if one thread is blocked and waiting, second thread in the same process could execute.
7.	Multiple resources	Multiple redundant processes use more resources than multiple threaded process.	With compare to multiple redundant process, multiple threaded processes need fewer resources.
8.	Address space	Context switch flushes the MMU (TLB) registers as address space of process changes.	No need to flush TLB as address space remains same after context switch because threads belong to the same process.

Syllabus Topic : Process Schedul ing

2.8 Process Scheduling

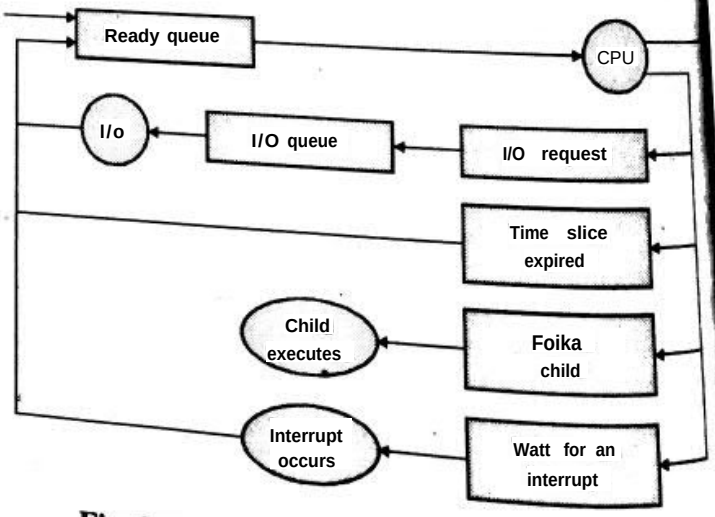
In order to improve CPU utilization, multiprogramming concept makes available several execution. Processes ready for

2.8.1 Scheduling Queues and Schedulers

- System places ail the ?nt ...ered processes « job queue.
- When ~ arere y le d f; « Slde >" job queue. for CPU. they kept in ... ««uti on and wait
- AW from lhe job ; queue -

other queues. ... operating system also has

device, the ?? =Pt«respecti vedevjce * Pedlar



Fi

8 2 8 1 1 Of"n Uin8 ,<liaBran , representation scheduling

- After allocation of CPU to the process start the execution. While executing the process, one of the numbers of events could occur.
- Until the process finishes the execution, it travels *between various* scheduling queues. The operating system selects the processes from queues based on some policy.
- This selection is carried out by the program called as scheduler. There are three types of schedulers to schedule a process. These are shown in Fig. C2.4.

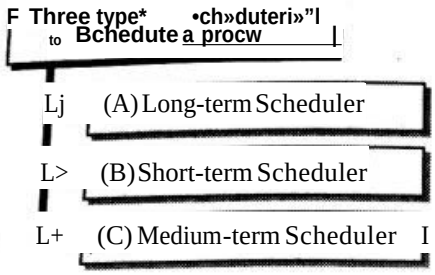


Fig. CZ4 **: Types of scheduler**

-> 2.8.1(A) Long-term Scheduler

Q. Explain long-term scheduler.

- When programs are submitted to the system for the purpose of processing, long term scheduler comes to know about it.
- Then its job is to choose processes from the queue and place them into main memory for execution purpose.
- CPU bound processes require more CPU time and less I/O time until execution completes.
- On the contrary, I/O bound processes use up more time in doing I/O and require less CPU time for computation.
- The main job of the long term scheduler is to provide a balance mix of I/O bound and CPU bound jobs.
- The number of processes in memory for execution and degree of multiprogramming is related to each other.
- More number of processes in memory for execution indicates degree of multiprogramming is high. Long term scheduler controls the degree of multiprogramming.
- If the average rate of new process creation and average departure rate of processes leaving the system is equal then degree of multiprogramming is steady.
- The Jong term scheduler is not present in timesharing operating systems.
- When process state transition take place from new to ready, then there long term scheduler come into picture for scheduling purpose.

➜ 2.8.1(B) Short-term Scheduler

Q. Explain short-term scheduler.

- Processes which are in ready queue wait for CPU. A short term scheduler chooses the process from ready queue and assigns it to the CPU based on some policy.
- These policies can be First Come First Served (FCFS), Shortest Job First (SJF), priority based and round robin etc, Main objective is increasing system performance by keeping the CPU busy.
- It is the transition of the process from ready state to running state. Actual allocation of process to CPU is done by dispatcher.
- Short term scheduler is faster than long tern scheduler and should be invoked more frequently compare to long term scheduler.

■4 2.8.1(C) Medium-term Scheduler

Q. Explain medium-term scheduler.

- If the degree of the multiprogramming increases, medium-term scheduler swap out the processes from main memory.
- The swapped out processes again swapped in by medium-term scheduler.
- This is done to control the degree of multiprogramming or to free up a memory.
- This is also helpful to balance the mix of different processes, some time sharing operating system have this additional scheduler.

2.8.1(D) Comparison of Three Schedulers

Q. Compare the functions of different types of schedulers.

Sr. No.	Long-term Scheduler	Short-term Scheduler	Medium-term Scheduler
1,	Selects processes from the queue and loads them into memory for execution.	Chooses the process from ready queue and assigns it to the CPU.	Swap in and out the processes from memory.

Sr. No.	Long-term Scheduler	Sbort-tenn Scheuler	Medium-term Scheduler
2.	Speed is less than the short term scheduler.	Speed is very fast and invoiced frequent ly than long term scheduler.	Speed is in between both short term scheduler and long term scheduler.
3.	Transition of process state from Ne* to Ready-	Transition of process state from Ready to Executing.	No process state transition'
4.	Not present in time sharing system.	Minimal in time sharing system.	Present in time sharing system.
5.	Supply a reasonable mix of jobs, such as I/O bound and CPU bound	Select a new process to CPU allocate frequently.	Processes are swapped in and out for balanced process mix.
6.	It controls degree of multiprogram ming through placing processes in ready queue.	It has control over degree of multiprogramm ing as it allocates processes to CPU for execution.	Reduce the degree of multiprogramm ing by swapping the processes in and out.
7.	It is also called as job scheduler.	It is also called as CPU scheduler.	It is used for swapping.

Syllabus Topic: interprocew CommuSZT

2.9 -J»~MssComm unication

rasa-

- Message passi mg
- Shared memory

ModusofInterpret

Z2

2.9.r MessagePa8Sn9

- «tion and communication
- synch requiremenU should satisfied with each other Synch
- Tfe requued to achieve the mu
- processes do communic
- Independen P proce sses may need to other but coope Coopemdve processes information- communicates through shared memory Or passing-
- Message naming provides both functions, u passing has the further benefit that it lends nation in distributed systems multiprocessor and un shared-memory systems-
- Following are the two primitives used io passing :
  1. send (destination, message)
  2. receive (source, message)
- This is the minimum two operations reqjjJ processes to send and receive the messages. sends data in the form of a message to another pn indicated by a destination. A process receives di executing the receive primitive, indicating the: and the message.
- Communication by sending and receiving mess require synchronization. The receiver cannot tea message until it has been sent by another process,
- The sending process is blocked until the mess received, or it is not after the send primitive ei by process. Similarly, when a process issues a primitive, there are two possibilities :
  - Previously sent message is received and e continues.
  - there is no waiting message, then either process is blocked until a message arrived the process continues to execute, abando -l attempt to receive,
- Thus
  - both the sender and receiver can be blo nonblocking.

Particul°? blnad o nS are comraon - although

«®*wi Xi iplemented: y have only one

01 4

- **The blocking send and blocking receive :** Until the message is handed over, the sender and receiver both gets blocked.
- **The nonblocking send and blocking receive :** After sending the message, the sender continues its work but receiver remains blocked until message is arrived to it. This permits a process to send one or more messages to a multiple destinations as quickly as possible. Here receiver is in need of message so that it can resume the execution . So it gets blocked until message arrives.
- **Nonblocking send, nonblocking receive :** Neither party is required to wait.
- **The nonblocking send and non blocking receive:** Both sender and receiver will not wait and both will continue the work.

- Message passing system should give guarantee that messages will be correctly received by receiver.
- Receiver sends acknowledgement to sender after receiving the message.
- If acknowledgement not received in defined time then sender resend the message. It also offers authentication service.

2.9.2 Shared Memory

- Cooperating processes require an Interprocess Communication (IPC) mechanism that will allow them to exchange data and information.
- 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 of messages exchanged between the cooperating processes.

Syllabus Topic : Multithreading

2,10 Multithreading

- A thread is a single sequence stream within a process. Threads are also called as lightweight processes as it possess some of the properties of processes. Eac thread belongs to exactly one process.
- In operating system that support multithreading, process can consist of many threads.
- These threads run in parallel improving the application performance. Each such thread has its own CPU state and stack, but they share the address space of the process and the environment.
- Threads can share common data so they do not need to use interprocess communication. Like the processes, threads also have states like ready, executing, blocked etc. priority can be assigned to the threads just like process and highest priority thread is scheduled first.
- Each thread has its own Thread Control Block (TCB). Like process context switch occurs for the thread and register contents are saved in TCB.
- As threads share the same address space and resources, synchronization is also required for the various activities of the thread.

2.11 Types of Threads

Q. Explain user level and kernel level threads with advantages and disadvantages.

Threads can be implemented in two ways.

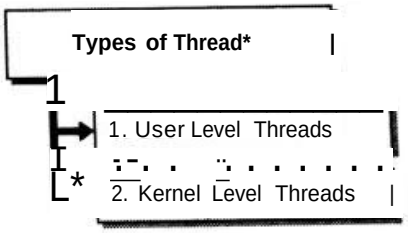


Fig. C2.6 : Types of Threads

1. User Level Threads

- In user level implementation, kerne) unaware of the thread. In this case, thread package entirely put in user space. Java language supports threading package.
- User can implement the multithreaded application in java language.
- Kernel treats this application as a single threaded application. In a user level implementation, all of the work of thread management is done by the thread package.
- Thread management includes creation and termination of thread, messages and data passing between the threads, scheduling thread for execution, thread synchronization and after context switch saving and restoring thread context etc.

of thread requires less time 3,14

-------destroying

.. Creation and «s r cost o f allow * (he memory'

threads

while destroying lcss tin*- S , Tifias

need to flusii

- Both op - XX. no context switch. User-

address 5p* exin low over- rfiead, and can

performance. Fig. 2.11.1

nehieve high <* F hreads.

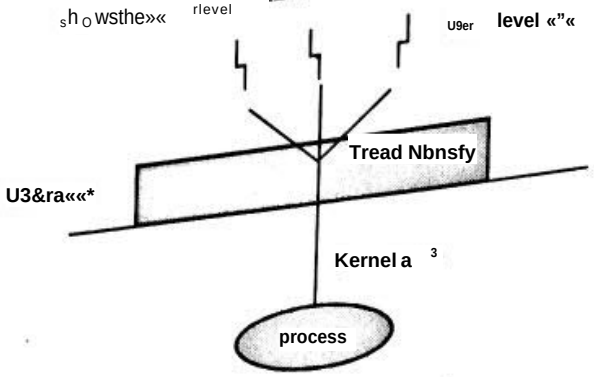


Fig. 2.11.1 : User level threads

Advantages of user level threads

- Thread switching does not require flushi... value of CPU register doing CPU accounting. On y need to be stored and reloaded again.
- user level threads are platform independent, ttey can execute on any operating system.
- Scheduling can be as per need of application.
- Thread management are at user level and do y thread library. So kernels burden is taken by threading package. Kernels time is saved for other acuvittes

Disadvantages of user level threads

- If one thread is blocked on I/O, entire process gets blocked.
- The applications where after blocking of one thread other requires to run in parallel, user level threads are of no use.

Z Kernel Level Threads

- In this, threads are implemented in operating system's kernel. The thread management is carried out by kernel.
- All these thread management activities are carried out in kernel space. So thread context and process context switching becomes same.

Application can be written as multithr cation are supported as

- Kernel threads are generally more S than the user threads.
- The kernel Tough one thread in a process is bl process even *0* does not block lhe thread
- Kernel can sim < onrnultipI e processes, from the same p kernel level thrMds
- contest swttc are gene raUy requires more time
- Kernel threads than the user threads. create and manage

Syllabus Topg- Process - Multithreading Models

2,1 2 MultithreadingModels

Q. Explain van' * 3 * 03 models.

Hering the advantages of user level and fe, Xi treads, a hybrid threading model using X of threads can be implemented.

- The Solaris operating system supports this hybr model.
- In this implementation, all the thread manage™ functions are carried out by user level thread package user space. So operations on thread do not reqs kernel intervention.
- The advantage of hybrid model of the threads is that, - the applications are multithreaded then it can I advantage of multiple CPUs if they are available. 0 threads can continue to make progress even if kernel blocks one thread in a system function,
- There are three types of multithreading models. These are shown in Fig. C2.7.

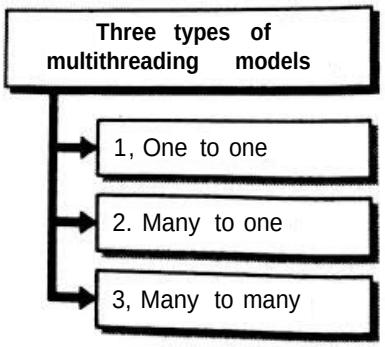


Fig. C2.7 : Types of multithreading models

-♦ 1. One to one model

- In this model relationship between user level thread and kernel level thread is one to one.
- It means that there is mapping of a single user-level thread to a single kernel-level thread.
- Because of such type of relationship multiple threads executes in parallel leading to more concurrency.
- However, since it is needed to create kernel thread for every new creation of user thread.
- So application performance will be degraded. Windows series and Linux operating systems try to minimize this problem by restricting the expansion of the thread count. OS72, Windows NT and windows 2000 use one to one relationship model,

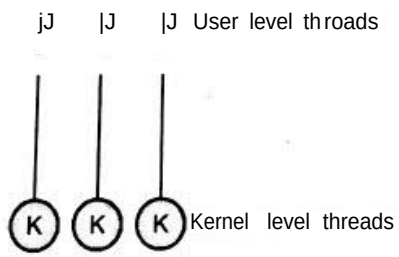


Fig. 2.12.1

-F 2. Many to one model

- In this model relationship between user level thread and kernel level thread is many to one.
- It means that there is mapping of many user-level threads to a single kernel-level thread.
- In this model management is done in user space. When one thread makes a system call for blocking, the entire process gets blocked.

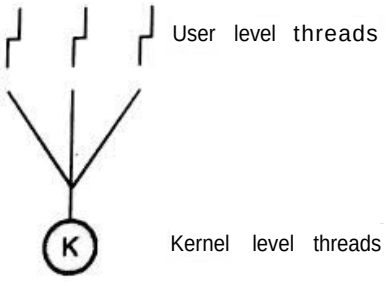


Fig. 2.12.2 : Many to one model

- At a time only one thread can access the Kernel thread at a time, so many other threads cannot execute in parallel on multiple processors.
- **Therefore concurrent execution of threads cannot be achieved.**
- **This type of relationship facilitates an effective context- switching environment, easily implementable even on simple kernels with no thread support**

➜ 3- Many to Many Model

- Many to many association exist between user level thread and kernel level thread in this model. It means that more number of user-level threads are allied to equal or less number of kernel-level threads.
- The necessity of altering code in both kernel and user spaces leads to a level of complexity not present in the one tone and many to one model.
- Like many-to-one model, this model offers an efficient context-switching environment as it repel from system calls.
- The keen complexity offers the potential for priority inversion and suboplima! scheduling with minimum coordination between the user and kernel schedulers.

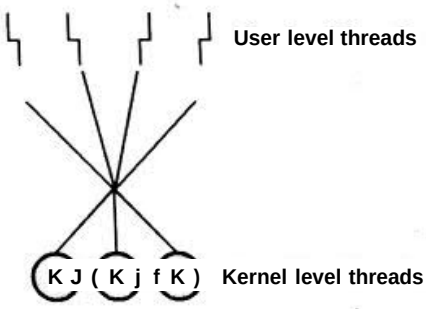


Fig. 2.12.3 : Many to Many model

Syllabus Topic : Thread Librads*

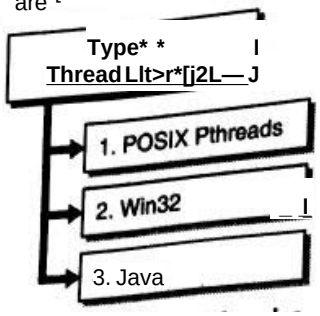
2.13 Thread Libraries

q Write note on thread libraries.

- Programmers can create and manage the threads using API that are provided by thread library.
- Thread library can be implemented at user level where it runs in user space without kernel support. Hence calling a function in library is local function call m user space.
- Kernel level library <ssupported by operating system The data structures and code remains in kernel space.

– Hence any invocation of function in library » system call to the kernel but nc ,bread libraries that are in use today.

Following are [he main Ihre



Fig, C2.8 : Thread libraries

.» 2.13.1 POSIX Pthreads

– Pthreads is the threads extension of the POSIX standard that can be provided as either a user-level or kernel-level library.

– Pthreads define API for the creation of threads and synchronization. As it is a specification, the des.gn of operating can choose their own way or implementation of specification.

- Solaris. Linux, Mac OS X, and Tru64 UNIX are the systems that implements Pthreads specification. For all the types of Windows operating systems,, a shareware implementation-is available in public domain.

➜ 2.13.2 Win32 Threads

– It is kernel-level library available on windows system. The approach for creation of threads using Win32 is similar to Pthreads.

- Just as Pthreads, in Win32 CreateThread( ) function is used to create the threads.

– The required set of attributes for the thread is passed to this function. Security information, stack size, and a flag which is set to specify if the thread is to start in a suspended state are the examples of these attributes.

-» 2.13.3 Java Threads

– Threads can be dmeclly carried out in Java programs Of 'he and management

– As most of the time, JVM is runs on the top of a host operating system

lwleawWbynukmg p, usual1 on the host system. existi ng

– VfindoWS systems. Java threads are HC0Ce n ted using the Wi032 and *0 UNIX and Linux systems. <4

– h set of f* atureS * aVallable jav k

– The . API to manage and create the tfe ads.

– 8ndjaVa r) method in java program is the e

– The Csingie threati in JVM. <

least single thread is supp

– HenCe In first approach of thread creati b, S siscmatedwhichis ram. a new clas ;

– Thread Class and then overrides its run() method

Icond approach a class is defined that implem Runnable interface.

Syllabtis Topic : Threading Imim,

2.14 Threading Issues

Q. What are different threading issues?

Following are the threading issues that needs considered with multithreaded programs.

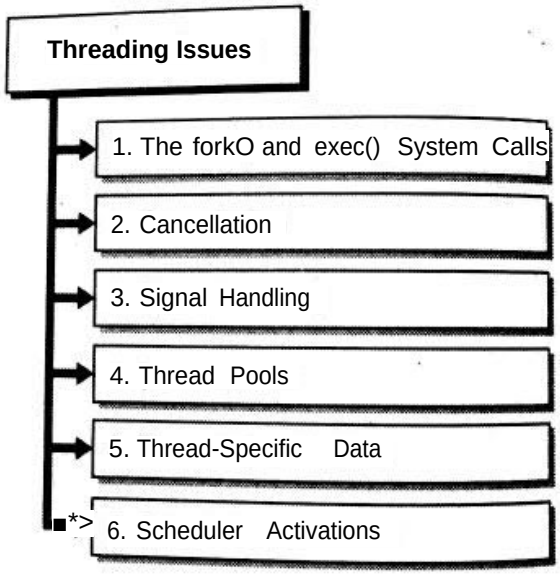


Fig. C2.9 : Threading issues

➜ 2.14.1 The fork() and exec() System Cails

– A separate duplicate process is created with fork system call. The meaning of fork() and exec() syste calls is different in multithreaded program.

– call? or UNIX Vcrsions - hen single thread of progra ( ) then it duplicate all the threads of progra

– LtLT."" UN". -r »-pi—«*

- When thread invokes exec( ) system call, the whole process including all the threads will be replaced by program that is given as parameter to exec( ) call.
- The use of particular version of forkf) is carried out as per application need.
- If it is necessary to call exec( ) just after forking then no need to duplicate all the threads. If exec( ) is not called just after forking then duplication of all the threads is required.

-♦ 2.14.2 Cancellation

- Termination of thread before it completes the execution is thread cancellation. If simultaneously many threads are searching through database and one thread return the result then cancellation of remaining threads is done.
- The thread chosen for cancellation is called as target thread. Following two scenarios arc there to cancel the target thread.
  - **Asynchronous cancellation.** Instant termination of the target thread is done by one thread
  - **Deferred cancellation.** The periodic check is carried out by target thread to decide whether it should terminate, permitting it an opportunity to terminate itself in an orderly fashion.
- The asynchronous cancellation is troublesome in situation if cancellation of thread is carried out to which system resources have been allocated.
- Asynchronous cancellation is also difficult in case thread is in middle stage of updating data that it i shared with other threads. Although operating system will reclaim the resources from cancelledI thread some resources will be there with cancelled thread and cannot be reclaimed.
- In deferred cancellation, thread to be cancelled checks the flag to know whether it should be cancelled or not. This always allows safe cancellation.

➜ 2.14.3 Signal Handling

In UNIX, process is informed by signal when some event is occurred. The signals can be synchronous or asynchronous depending on source and reason for event being signal. Following pattern is followed by synchronous or asynchronous signals.

- When particular event occurs, the signal is generated.
- This generated signal is undelivered to the process.
- The delivered signal should be handled

Following are the examples of synchronous signals.

- Illegal memory access.
- Division by zero.

If the executing program carries out above two actions then generated signals are delivered to the same process which performed above actions. Hence, these signals are called as synchronous signals. When signals are due to events of external process then the executing process receive asynchronous signals

Following are examples of asynchronous signals.

- Terminating process by applying some specific keystrokes.
- Timer expires.

Both types of signals are handled either by default signal handler or user defined handler. If program is multithreaded program then following options are available for delivery of these signals to the process..

- Signal can be delivered to the thread to which it is applicable.
- Signal can be delivered to every thread in the process.
- Signal can be delivered to specific threads in the process.
- Allocate any one particular thread to take delivery of all signals for the process

The way signals are delivered depends on types of signals generated.

➜ 2.14.4 Thread Pools

- There should be bound on number of threads active in system. If unlimited number of threads exists in system then many resources will be consumed by them.
- Thread pool is solution on this issue. When process starts up then some number of threads are created. These threads are then placed in pool and they waits or work.
- If server comprising these threads receives request it passes it to any of the available waiting thread in pool.
- Once thread completes the requested work, it returns to pool and waits for other work.
- If thread is not available in pool then server has to wait til] thread will be available.

Following benefits are provided due to thread pool :

- Instead of waiting to c rente[3] thread* servicing a request with an existing thread is always faster.
- A thread pool p vrdcs the * * . y necessary on that caist at ony ofic P ,,)rtrg c number of parallel .ystems that caano< suppo threads.

-» 2.14.5 ThrMd-Speel fic Data

- Always threads of a P[nl] d la offers beitefi'*' ° p cess. Surely, -his shanng multithreaded programming. the data rf [hat
- On the other hand, i. some situation, each thread might require IB own copy « f « rta in data, da(a is a thread-specific data.

➜ 2.14.6 Scheduler Activations

- tils import to consider common. » n between the needed by kernel and the thread library, whtcb may be the nw-to-nw "d also ** tiu kernel level model of threads.
- This coordination permits the number of kernel t hreads CO be dynamically adjusted to guarantee the best performance.
- Several systems implementing either the many-to-many or user and kernel level model of threads place an intermediate data structure between the user and kernel threads.
- This data structure is called as lightweight process, or LWP. For user level library LWP is virtual processor to schedule the application to run.
- LWP is attached with kerne) thread and OS schedules LWPs as kernel threads to run on physical processor. With blocking of kernel thread, LWP also blocks and tace user lhread assoeiated w · h ih ·s Lwp also blocks.
- Scheduler activation is the method f™ communication between the user-thread library and the kernel.
- The kernel offers an application with some number of virtual processors (LWPs) schedule user threads on[o ' 6 appllcatlon can Processor. an available virtual

Syll Topic Ruling = Basic Concepts

Process get 1

2.15 Process scn e<Julin g

2.15.1 Sc MduU rg Decisions

SCh ..uling decisions re made in following situati ons.

- A fter cr eation of new process. Decision i s n eded
- ch005 ingei P arC nt or child process for execution
- When any P eSS K kit, new process from ready que is selected
- is blo keu due to I/O or because of
- When process again new process from ready q y cue other reason.
- selected.
- On the occurrence of interrupt.

, M Type selwdullng

What is differenc Tbetween preemptive and preemptive scheduling?

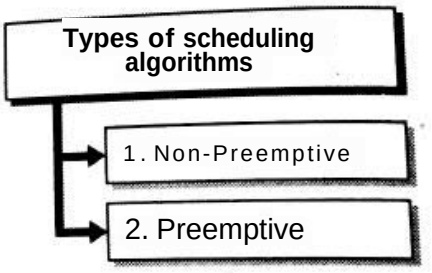


Fig. C2.10 : Types of scheduling algorithms

■4 1. Non-Preemptive

Non-preemptive algorithms are designed so that once a process is allocated to CPU, it does not free CPU until it completes its execution.

➜ 2. Preemptive

Preemptive algorithms allow taking away CPU from a process during execution. If highest priority process arrives in the system, CPU from currently executing low priority process is allocated to it. It ensures that always highest priority process will be executing.

Dispa,cher and dispatch latency

- Shod term scheduler allocates CPU to ready pw[6] based on some P"U$_C$». I. ,s the trattsltlodolttef process from ready suae ,orunningsute .

- Actual allocation of process to CPU is done by dispatcher. Short term scheduler is faster than long term scheduler and should be invoked more frequently compare to long term scheduler.
- The amount of time dispatcher needed to bring to stop one process and starts running of other process is called as dispatch latency.

Syllabus Topic : Scheduling Criteria

2.16 Scheduling

2.16.1 Scheduling Criteria

Q. What are the criteria for evaluation of scheduling algorithm performance?

- In multiprogramming, many programs remain in memory at the same time.
- Processes carry out I/O operations while performing the execution. Since I/O operations generally consume more time to accomplish with compare to CPU instructions, multiprogramming systems hand over the CPU to another ready process whenever a any process invokes an I/O operation.
- Short term scheduler allocates inc precedes iv vi CPU as per some policy called as scheduling algorithms.
- The main aim of the scheduling is to improve performance by keeping CPU busy all the time.
- Criteria for the for performance evaluation of the scheduling strategy is :
  - **CPU Utilization :** It is amount of time CPU remains busy.
  - **Throughput ;** Number of jobs processed per unit time.
  - **Turnaround time :** Time elapsed between submission of job and completion of its execut.on.
  - **Waiting time :** Processes waits in ready queue to get CPU. Sum of limes scent in ready queue is waiting time.
  - **Response Time :** Time from submission till the first response is produced.
  - **Fairness :** Every process should get fair share of the CPU time.

Syllabus Topic; Scheduling Algorithms

2-1 S_2 **Scheduling Algorithms**

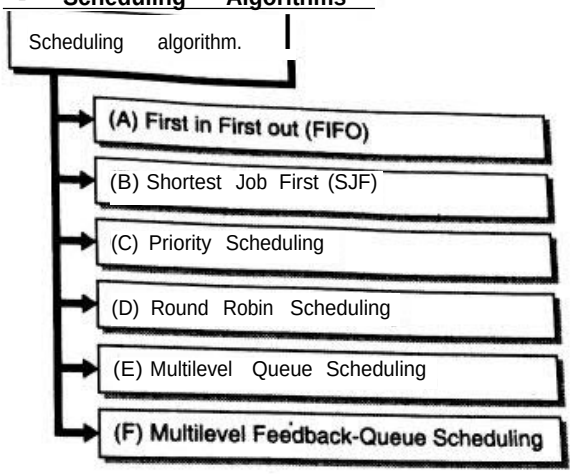


Fig. C2J 1 : Scheduling algorithms

➜ 2.16.2(A) First in First Out (FIFO)

Q. Explain FIFO scheduling algorithm.

- This is a Non-preemptive scheduling algorithm, FIFO strategy allocates the CPU to processes in the order of their arrival.
- This algorithm treats ready queue as FIFO. A process does not give up,
- CPU until it either terminates or performs I/O, If the longer job assigned to CPU then many shorter jobs has to wait. As long processes can hold the CPU, this algorithm gives fewer throughputs.
- This algorithm can be easily implemented using FIFO queue. In time sharing system every user should get the CPU time at regular interval. So FIFO algorithm is not useful in such situations.
- FIFO algorithm is inappropriate for interactive systems; large fluctuations in average turnaround time are possible.
- If we assume that arrival time is zero. Consider the following example :

Process	Burst time
P1	24
P2	03
P3	3

Assume that processes arrive in the order P1, P2, and P3.The Gantt chart shows the result.

P1	P2	P3
0	24	32

Turnaround time for P2 = (8 - 0) = 8

Turnaround time for P3 = (32 - 0) = 32

Turnaround time for P1 = (3 - 0) = 3

Average Turnaround time = (3 + 8 + 32) / 3 = 14.33

Waiting time for P1 = 0

Waiting time for P2 = 3

Waiting time for P3 = 8

Average Waiting time = (0 + 3 + 8) / 3 = 3.66

- Average waiting time is Significantly reduced in this example.
- The waiting time and average turnaround time vanes-order of job arrival varies.

16.2(B) Shortest Job First (SJF)

Q. Explain SJF and SRTN scheduling algorithms.

- Shortest Job First (SJF) may be preemptive or non preemptive.
- Shortest Job First (SJF) improves the average response time.
- Ready queue is maintained in order of increasing job length. When a job comes in, insert it in the ready queue based on its length. SJF minimizes the average wait time because it gives service to less execution time processes before it gives service to large execution time processes.
- While it minimizes average wait time, it may punish processes with high execution time.
- If shorter execution time processes are in ready list, then processes with large service times tend to be left in the ready list while small processes receive service.
- It may happen in extreme case that always short execution time processes will be served and large execution time processes will wait indefinitely.
- This starvation of longer execution time processes is the limitation of this algorithm.
- Consider the example discussed for FCFS algorithm. In this example it is assumed that SJF is non preemptive.
- If the order of arrival is P2, P3, P1, the order of execution time will be 3, 5, and 24. There is significant reduction in average turnaround time and average waiting time.

Consider the following example of four jobs based on shortest job first (SJF) algorithm.

Process	Arrival time	Burst time
P1	0	10
P2	1	5
P3	2	8
P4	3	15

P1	P2	P3	P4
0	10	15	23

Turnaround time for P1 = (10 - 0) = 10

Turnaround time for P2 = (15 - 1) = 14

Turnaround time for P3 = (23 - 2) = 21

Turnaround time for P4 = (38 - 3) = 35

Average Turnaround time = (10 + 14 + 21 + 35) / 4 = 20

Waiting time for P1 = 0

Waiting time for P2 = (10 - 1) = 9

Waiting time for P3 = (15 - 2) = 13

Waiting time for P4 = (23 - 3) = 20

Average waiting time = (0 + 9 + 13 + 20) / 4 = 10.5

- In Preemptive Shortest Job First (SJF), if newly arrived process execution time with compare currenriy executing process, then the CPU will be given to newly arrived process.
- The preemptive Shortest Job First (SJF) is called **Shortest Remaining Time Next Scheduling (SRTN)**.

Consider the above example for preemptive Shortest Job First (SJF). The Gantt chart shows the result.

P1	P2	P3	P1	P4
0	1	6	14	23	38

- P1 arrives at time 0. so get the CPU as it is only process in the queue. At time 1, process P2 arrives hai execution time 5 (less than remaining time 9 of pl)- P1 is preempted and P2 is scheduled. At time 2, process P3 arrives having execution time 8 which is greater than remaining time of P2.
- So P2 will complete the execution. After P2, process P3 will execute as its execution time is less than P1 (9) and P4 (15). Process P1 has remaining execution time 9. So it is scheduled next. Finally P4 will complete the execution.

Turnaround time for P1 = (23 - 0) = 23

Turnaround time for P3 = (14—2) = 12
Turnaround time for P4 = (38 - 3) = 35
Average Turnaround time = (23 + 5 + 12 + 35) /4 = 18.75
Waiting time for Pl = (14 - 1) = 13
Waiting time for P2 = (i - l) = O
Waiting time for P3 = (6 - 2) = 4
Waiting time for P4 = (23 - 3) = 20
Average Waiting time = (13 + 0 + 4 + 20) / 4 = 18.75

➜ 2.16.2(C) Priority Scheduling

Q. Describe priority scheduling algorithm.

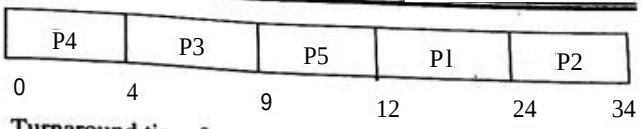
- In priority scheduling, each process has a priority which is a integer value assigned to it.
- Smallest integer is considered as highest priority and largest integer is considered as lowest priority. Always highest priority process gets the CPU.
- In some system, largest number is treated as a highest priority and it depends on implementation.
- Jf priorities are internally defined then some measurable quantity such as time limits, memory requirements, the number of open files, and the ratio of average I/O burst to average CPU burst are used to compute the priorities.
- External priorities are assigned on the basis of factors such as importance of the process, the type and amount *of* funds being paid for computer use, the department sponsoring the work, and other, often political, factors. All these factors are not related to the operating system.
- Preemptive and non preemptive SJF is a priority scheduling where priority is the shortest execution time of job.
- In this algorithm, low priority processes may never execute. This is called **starvation**.

Solution to this starvation problem is **aging.** In aging as time progresses, increase the priority of the process so that lowest priority processes gets converted to highest priority gradually.

Consider the following example for priority scheduling.

The Gantt chart shows the result.

Process	Burst time	Priority
P1	12	4
P2	10	5
P3	5	2
P4	4	1
P5	3	3



Turnaround time for P1 = (24-0) = 24
Turnaround time for P2 > (34-0) = 34
Turnaround time for P3 = (9—0) = 9
Turnaround time for P4 = (4-0) = 4
Turnaround time for P5 = (12-0) = 12
Average Turnaround time = (24 + 34 + 9 + 4 + 12) / 5
= 16.6
Waiting time for P1 = 12
Waiting time for P2 = 24
Waiting time for P3 = 4
Waiting time for P4 = 0
Waiting time for P5 = 9
Average Waiting time = (12 + 24 + 4 + 0 + 9) / 5 = 9.8

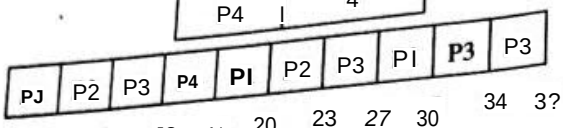
➜ 2.16.2(D) Round Robin Scheduling

Q. With the help of example, explain Round Robin Scheduling algorithm.

- Round Robin Scheduling is designed especially for time-shari ng systems where many processes get CPU on time sharing basis.
- In this algorithm, a small unit of time called time quantum is defined.
- CPU is allocated to each process for this time quantum period of time. To implement this scheduling, ready queue treated as FIFO queue.
- The new processes go to the tail of queue and each time CPU chooses the process from head of queue.
- When time quantum expires, context switch occurs and CPU switches to other process which is scheduled next.
- The time quantum is fixed and then processes are scheduled such that no process get CPU time more than one time quantum.
- If process is executing and request for I/O the process goes in waiting (blocked) state.
- After the completion of I/O, process again gets added at the tail of ready queue. The time quantum should not be very small or very large.
- If the time quantum is very large, the algorithm will behave just like FCFS.
- A smaller time quantum increases context switches leading to performance degradation (less throughput) as context switches elapses more time.

Consider the following example. If we ... seconds then the Gantt chart shown ... quantum of 4 mm ... the result.

proce**	gural lime
Pl	11
P2	7
P3	15
P4	4



0 4 8 J2 1* 20 23 27 30 34 3?

Turnaround time for P^1 = (30-0) = 30

Turnaround lime for P2 = (23-0) = 23

Turnaround rime for P3 = (37 - 0) = 37

Tur7iarou „d time for P4 = (16-0) = 16

Average Turnaround time = (... + 23 + 37 +]6) / 4 = 26.5

Waiting time for Pl = 0 + *16 – 4) + (27 – 20) = 19

Waiting time for *P2* = 4 + (20-8) = 16

Waiting time for P3 = 8 + (23- 12)*(3O-27)-22

Waiting time for P4 = 1 2

Average Waiting time = (19 + 4 + 22 + 12)/4 = 14.25

→ 2.16.2(E) Multilevel Queue Scheduling

Q. Eqilain multilevel queue scheduling algorithm.

- Sometimes it is necessary to categorize the processes into different groups. For example, separation is made between interactive processes and batch processes.
- The response-time need of these two processes can be dissimilar. So these processes can have different scheduling requirement. Also, interactive processes may have higher priority over batch processes.
- In multilevel queue scheduling algorithm, there are multiple ready queues. The allocation of process to the particular queue depends on property of that process such as memory size, priority of the process, or its type.
- Scheduling algorithm for different queues can be different. The round robin scheduling algorithm can be used for interactive processes queue and first come first serve scheduling algorithm can be used for batch processes queue.
- In addition to scheduling for each queue, there must be scheduling between the different queues. The fixed-priority preemptive scheduling IS used for scheduling between *the* different q_{Ueues} .

sf seeSSSS hc five 9 ueuCS and its ...
Let USe° oskler "scheduling algorithm. P "M ... ultileV q "written below in the o rder of ... queues are Number 1 indi Cates ... ass" d to them. V ... pri* ority and 5 is lowest priority.

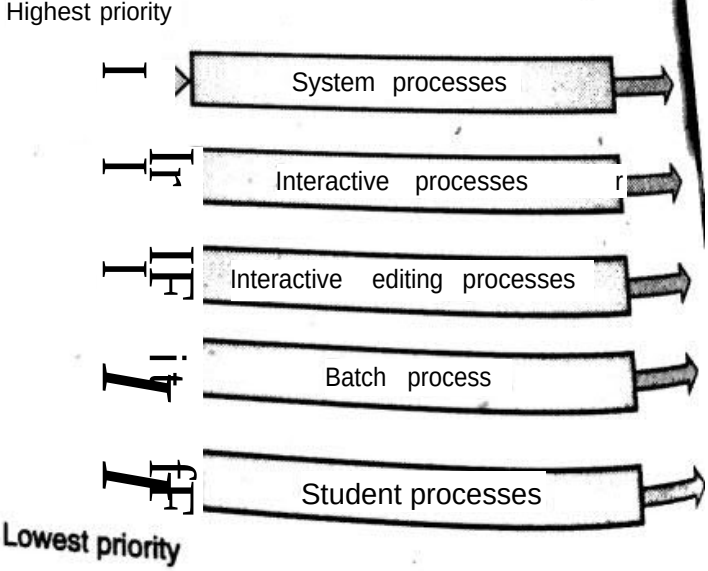
1. System processes queue
2. Interactive processes queue
3. Int eracdve editing P r — S queue
4. Batch processes queue
5. Student prases queue

... higher priority queue has complete priority ... Ever * ononty <i es - unless and until higher ... lowest ... XZ become empty, processes in 4 ... queue cannot run. . I ... L... example, processes tn tnteracuve editi 1 ... could not execute unless and until '.1 ... Zesses in the queueS ior SyStem p3 ... interactive processes finish the execution and ... queues becomes empty. 1

If process in lower priority queue is executed the same time other process belonging to 1 ... priority queue arrives then currently executing pro... in lower priority queue should be preempted. If s, ... process entered the ready queue while a student p... was executing, the student process would be preem... and batch process would be scheduled for execution.

In this algorithm, a fixed amount of CPU tlme K g... to each queue and within this time different proce... from this queue are scheduled.

For example, the interactive process queue caul given 70 percent of the CPU time for round j scheduling among its processes, whereas the h queue get 30 percent of the CPU to give to its procs on an FCFS basis.



2.16.2(F) Multilevel Feedback-Queue Scheduling

Q- Explain multilevel feedback queue schedUii?? algorithm.

- In multilevel queue scheduling processes are not permitted to transfer from one queue to the other Also queue assigned to the processes are permanent and cannot be changed, leading to the low scheduling cost, but it is not flexible.
- The multilevel feedback-queue scheduling algorithm processes can move from one queue to other. The CPU-bound processes use more CPU time and hence it will be transferred to a lower-priority queue.
- I/O-bound processes use less CPU time. Keeping CPU bound processes in low priority queue automatically leads to keeping the I/O bound and interactive processes in the higher-priority queues.
- There is a chance of starvation because of processes waiting for longer period of time in a lower-priority queue. It is avoided by aging; by transferring these waiting processes in a higher-priority queue.
- Consider the example of a multilevel feedback-queue scheduler with four queues, queue-0, queue-1, queue-2 and,queue-3. Initially the scheduler starts executing all processes in queue-0.
- Once all the processes in queue-0 finish the execution, and queue-0 becomes empty, scheduler will consider the processes in queue-1 for execution. After queue-1 becomes empty then scheduler will consider the processes in queue-2 for execution
- In the same way, processes in queue-3 will only be executed if queue-0, queue-1 and queue-2 are empty.
- A process arriving for queue-1 will preempt a process in queue-2. In the same way process that arrives for queue-0 will preempt a process in queue-1.
- Similarly if process arrives for queue-2 preempts process in queue-3.A process from the ready queue is placed in queue-0.

Multilevel feedback queue scheduling with 3 queues are shown in figure 3.9. If the process does not finish the execution within allocated time quantum of in queue-0, it is moved to the tail of queue-1• tum

The process at the head of queue- 1 is given a quan of 16 milliseconds if and only queue-0 is empty.

- If it does not finish ... queue- 2, Proce ... is placed into basis but are run nni ... queue-2 are run on an FCFS
- If the process ... qUCUeS 0 and less then this ... burs of 8 milliseconds or priority to it tk ... gives highest CPU. finish i™T ... get the burst ... its next UO
- ... above ... below 24 lower priority than shorter pr e ... with ... automatically go down to queue 2 and are served in FCFS order with any CPU cycles left over from queues 0 and 1.

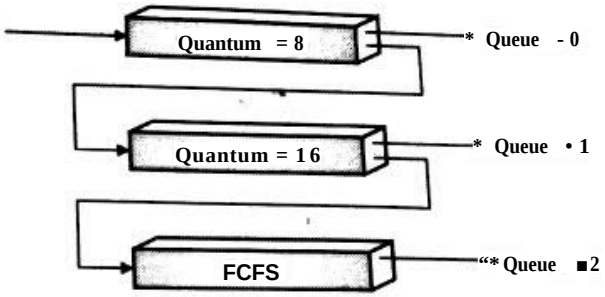


Fig. 2.16.2 : Multilevel feedback queues

2.17 Examples on Uniprocessor Scheduling Algorithms

Example 2.17.1 **MU - Dec. 2014, 10 Marks**

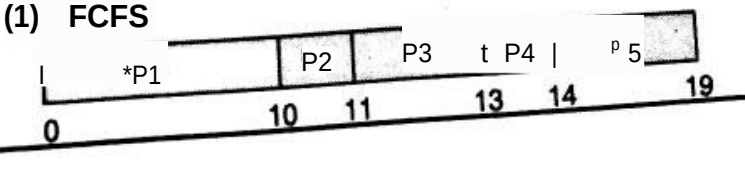
Consider the following set of processes, with the length of the CPU burst given in milliseconds:

Process	Burst Time	Priority
PI	10	3
P2	1	2
P3	2	
P4	1	
P5	5	2

The processes are assumed to have arrived in the order Pt, P2 P3 P4, P5 al1 at tlme 0 Draw ,0Ur Gantt Cha ,1S illustrate the execution of these processes using toe following scheduling algorithms: FCFS. WF nonpreemptive, priority (a smaller priority number implies a hrgher pnort and RR (quantum = 1) and also calculate turnaround W average waiting time.

Solution :

(1) FCFS

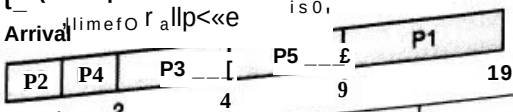


Arrival time for all processes is 0.

process	priority	Finish time	TAT	WMiti t,rne
PI	3	10	10	0
P2	1	H	11	to
P3	3	13	13	11
P4	4	14	14	13
P5	2	19	19	14
Average			134	9.6

(2) SiF(N nprt emp,i ,,<,)

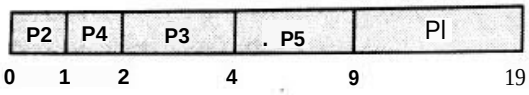
Arrival timefO r allp<«e is 0,



Process	Priority	Finish time	tat	Waiting time
PI	3	19	19	9
P2	1	1	1	0
P3	3	4	4	2
P4	4	2	2	1
P5	2	9	9	4
Average			7	3.2

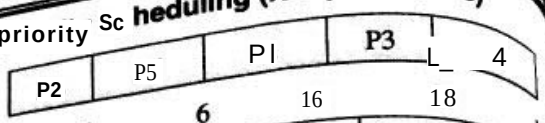
(3) SJF (Preemptive)

Since all processes arrive at same time 0, the solution is same as nonpreemptive SJF.



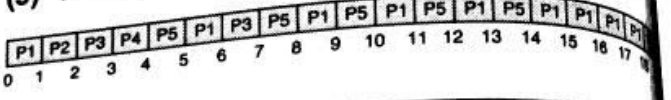
Process	Priority	Finish time	TAT	Waiting time
L	3	19	19	9
P2	1	1	1	0
Lzlj	3	4	4	2
P4	4	2	2	1
P5	2	9	9	4
Average			7	3.2

(*) priority Sc heduling (Nonpreemptive)



proce85	priory	Finish time	TAT	Waiting time
P£	3	16	16	6
P2	1	1	1	r
P3	3	18	18	16
P4	4	19	19	18
P5	2	6	6	1
Average			13.4	8.2

(5) Round Robin (time quantum = 1)



Process	Priority	Finish time	TAT	Waiting
PI	3	19	19	9
P2	1	2	2	1
P3	3	7	7	5
P4	4	4	4	3
P5	2	14	14	9
Average			9.2	5.4

Example 2.17.2

Calculate average waiting time and average turnaroundt for the following situation. Assume quantum of 2 ms.

Process	Burst Time	Priority	Arrival timei
P1	5	1	1
P2	7	3	5
P3	6	2	0

(i) Preemptive SJF

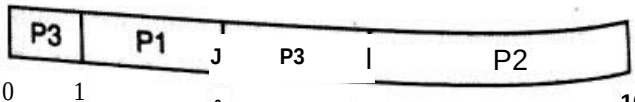
(ii) Preemptive priority

(Hi) RR

Solution :

(0 Preemptive SJF

Following is the Gantt chart



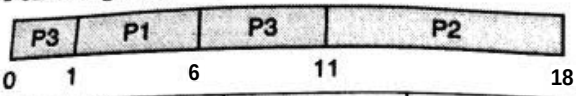
1Process	Priority	TAT	Waiting time
Pl	I	(6 - l) = 5	(1 '0 = 0
P2	3	(18 - 5) = 13	(11 - 5) = 6
P3	2	(11 - 0) = 11	(6 -]) = 5
Average j		9.6	3.66

Average TAT = (5 + 13 + 11)73 = 9.6

Average waiting time = (0 + 6 + 5)73 = 3.66

(ii) Preemptive priority

Following is the Gantt chart



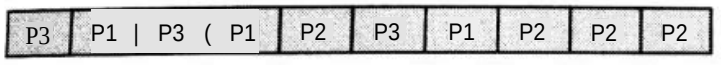
Process	Priority	TAT	Waiting time
p	1	(6-1) = 5	(l'l) = 0
F	3	(18-5) = 13	(11-5) = 6
P3	2	(11-0) = 11	(6- 1) = 5
Average		9.66	3.66

Average TAT = (5 + J3 + 11)/3 = 9.6

Average waiting time = (0 + 6 + 5)/3 = 3.66

(iii) RR

Following is the Gantt chart



Process	Priority	TAT	Waiting time
P1	1	(13-1) = 12	(2- 1) + (6-4) + (12-8) = 7
P2	3	(18-5) = 13	(8-5) + (13-10) = 6
P3	2	(12-0) = 12	(4-2) + (10-6) = 6
Average]		12.33	6.33

Average TAT = (12 + 13 + 12)/3 = 12.33

Average waiting time = (7 + 6 + 6)/3 = 6.33

Example 2.17,3

Consider the four processes P1, P2 t P3 and P4 with length of CPU burst time. Find out average waiting time and average turnaround time for the following algorithm.

(!) FCFS

00 RR (slice = 4 ms)

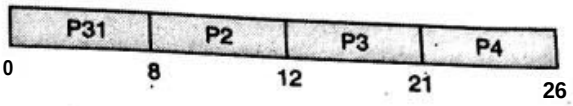
W SJF

Process	Arrival time	Burst time
P1	0	8
P2	1	4
P3	2	9
P4	3	5

Solution :

(0 FCFS

Following is tbc Gantt chart



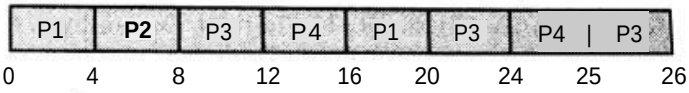
Process	TAT	Watting time 1
P1	(8'0) = 8	0
P2	(i2-1) = U	(8-1) = 7
P3	(21 -2) = 19	(12-2) = 10 1
P4	(26- 3) = 23	(21-3) = 18 j
Average	15.25	8.75 j

Average TAT = (8 + 11 + 19 + 23)/4 = 15,75 ms.

Average waiting time = (0 + 7 + 10 + 18)74 = 8.75 ms.

(ii) RR (slice = 4 ms)

Following is the Gantt chart



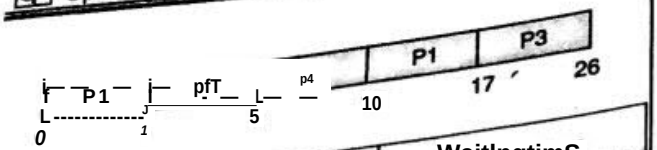
Process	TAT	Waiting time
P1	20	(16-4) = 12
P2	(8-1) = 7	(4-1) = 3
P3	(26-2) = 24	(8- 2) + (20- 12) + (25- 24) = 15
P4	(25-3) = 22	(12-3) + (24-16) = 17
Average	18.25	11.75

Average TAT = (20 + 7 + 24 + 22)74 —18.25ms

Average waiting time = (12 + 3 + 15 + 17)74 ~ 11,75ms

(iii) Preemptive SJF

Following is the Gantt chart



Process	tat	WaitIngtimS
Pl	(17 – 0) = 17	(1 0 - 1) * 9
P2	(5 - 1) = 4	(1 - 0 = 0
P3	(26 – 2) = 24	0 7 2) = 15
P4	(10-3)jL2	<5– 3) = 2
Average	13	6.5

Average iaiT = (17 + 4 + 24 + 7)/4 = 13 nis.

Average waiting time = (9 + 0 + 15 + 2)/4 = 6.5 ms.

Example 2.17.4 for execution at

suppose mat the following process amve

time indicate

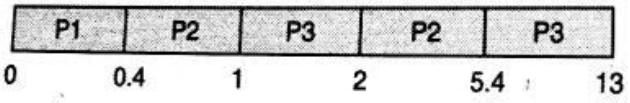
Process	Arrival time	Burst time
Pt	0,0	8
P2	0.4	4
P3	1.0	1

Calculate average waiting time anu average turnaround time using SRTF and SJF .

Solution:

(i) Preemptive SJF is SRTF (Shortest Remaining Time Next)

Following is the Gantt chart



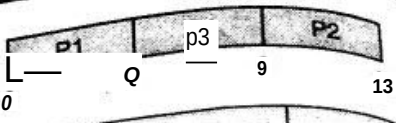
Process	TAT	Waiting time
Pi	(13 – 0) = 13	(5.4 – 0.4) = 5
P2	(5.4 – 0.4) = 5	(2 – 1) = 1
P3	(2 – 1) = 1	(1 – 1) = 0
Average	6.33	2

Average TAT = (13 + 5 + l)/3 - fi „

Average waiting time = (5 + 1 + 0)/3 = 2 “ “ .

W Non - preemptive SJF

Following is the Gantt chart



Process	TAT	Waith ng time
PI	(8 - 0) = 8	0
P2	(I OAI	2.6 (9 0.4) = 8.6
P3	(9 - 1) = 8	(8 – 1) = 7
Average	9.53	5.2

Averag e TAT = (8 + 126 * 8)/3 = 9.53

Average waiting tin* = (0 + 8.6 + 7)/3 = 5.2 ms

Exampl® 2.17.5

consider the foliowing set ot processes having their burst time 22ri— millisecond

Process	CPU Burst time	Arrival tin*
P1	10	0
P2	5	1
P3	2	2

Calculate average waiting time using tolkswin scheduling algorithms.

(D FCFS

(2) SJF

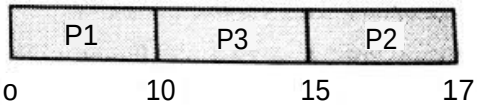
(3) Priority scheduling having priority range from 1 respectively for process P1 = 3, P2 = 2, P3 = 3as

(4) RR (slice = 2)

Solution :

(1) FCFS

Following is the Gantt chart

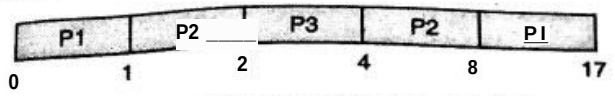


Process	Waiting time
P1	0
P2	(1 0 - 1) = 9
P3	(1 5 - 2) = 13
Average	7.33

Average waiting time = (0 + 9 + 13)/3 = 7.33

(2) Preemptive SJF

Follnu/Jtvrt

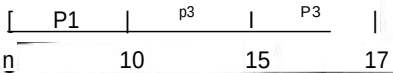


process	Waiting time
P1	(8 - 1) = 7
P2	(4 - 2 - 1) = 1
P3	(2 – 2) = 0
Average	2.67

Average waiting time = (7 + 1 + 0)/3 = 2.67 ms.

Prinritv scheduling having priority range from 1 to 3. respectively for process P1 = 3, P2 = 2, P3 = 3 as given.

Folio wing is the Gantt chart

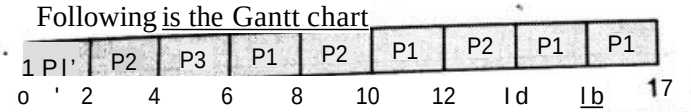


Process	Waiting time
P1	0
P2	(10 - 1) = 9
P3	(15 - 2) = 13
Average	7.33

Average waiting time = (0 + 9 + 13)/3
= 7.33 ms. (Same as FCFS)

(4) RR (slice = 2)

Following is the Gantt chart



Process	Walting time
PI	(6 - 2) + (10 - 8) + (13 - 12) = 7
P2	(2 - 0) + (8 - 4) + (12 - 10) = 7
P3	(4 - 2) = 2
Average	5.33

Average waiting time = (7 + 7 + 2)/3 = 5.33 ms.

Example 2.17.6

Consider the following set of processes.

Process	CPU Burst time	Arrival time
PI	3	0
P2	5	1
P3	2	2
P4	5	3
P5	5	4

Calculate average waiting and turnaround time for each algorithm.

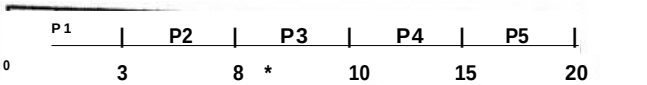
(1) FCFS

(2) SJF

(3) RR (slice - g)

Solution :

(1) FCFS : Following is the Gantt chart

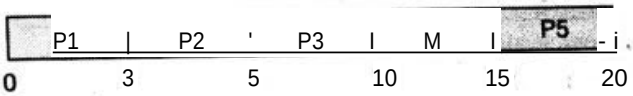


Process	TAT	Waiting time
PI	(3 - 3) = 0	0
P2	(8 - 1) = 7	(3 - 1) = 2
P3	(10 - 2) = 8	(8 – 2) = 6
P4	(15 - 3) = 12	(10 - 3) = 7
P5	(20 - 4) = 16	(15 - 4) = 11
Average	9.2	5.2

Average turnaround time = (0 + 7 + 8 + 12 + 16)/5
= 9.2 ms.

Average waiting time = (0 + 2 + 6 + 6 + 11)/5
= 5.2 ms.

(2) SJF : Following is the Gantt chart



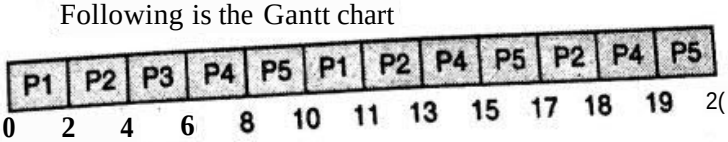
Process	TAT	Waiting time
P1	(3 - 3) = 0	0
P2	(10 - 1) = 9	(5 - 1) = 4
P3	(5 - 2) = 3	(3 - 2) = 1
P4	(15 - 3) = 12	(10 - 3) = 7
P5	(20 - 4) = 16	(15 - 4) = 11
Average	8.6	4.6

Average turnaround time = (0 + 9 + 3 + 12 + 16)/5
= 8.6 ms.

Average waiting time = (0 + 4 + 1 + 7 + 11)/5
= 4.6 ms.

(3) RR (slice = 2)

Following is the Gantt chart



Process	TAT	Waiting time
P1	(11 – 0) = 11	------(io5iL— —_
P2	(18 – 1) = 17	(2 – 1) + (11 – 4) + (17 – 13) = 12
P3	(6 – 2) = 4	(4 – 2) = 2
P4	(19 – 3) = 16	(6 – 1) + (13 – 8) + (18 – 15) = 11
P5	J20-4KJ1	(8 – 4) + (15 – 10) + (19 – 17) = 11
Average	12.8	8.8

Average turnaround time = (11 + 17 + 4 + 16 + 16)/5

= 12.8 ms

Average waiting time = (8 + 12 + 2 + 11 + 11)/5

= 8.8 ms.

Example 2.17.7

Assume that you have following jobs to execute with one processor

Job	CPU Burst time	Arrival time
0	75	0
1	50	10
2	25	10
3	20	80
4	45	85

Suppose system uses round robin with quantum of 15.

(i) Draw Gantt chart

(it) And average wait and tumarou nd time

Soln. : The Gantt chart is,

P1	P4	PC

0 15 30 45 W re M 100 115 130 145 ISO 165 180 185 200 215

Proem	TAT	Wafting time
PO	(215-0) = 215	(45 - 15) + (115 - 60) + (165 - 30) + (200 - 180) = 140
PI	(185-10)-175	(15 - 10) + (60 - 30) + (130 - 75) + (180 - 145) = 125
P2	(85 - 10) = 75	(30 - 10) + (75 - 45) = 50
P3	(150-S0) = 70	(85 -80) +(145-100) = 50
P4	(200 - 85) = 115	(100 -85) + (150- 115) + (1B5 - 165) = 70
Average	130	87

Average turnaround time = (215 + 175 + 75 + 70 + 115)/5

= 130 ms.

Average waiting time = (140 + 125 + 50 + 50 + 70)/5

= 87 ms.

5** Use following Scheduling algorithms *° «**!» J AWT for the following process

(i) Non-preemptive SJF

(ii) p2,p«v eandnor

(iii) Preemptive priority.

jw*!*..	Arrival time	Burst time	Priority
P1	0	8	3
P2	i	1	1
P3	2	3	2
P4	3	2	3
P5	4	6	4

Solution •

(I) Hf°

P1	P2	P3	P4	P5

0 8 9 12 14

Process	Priority	Hnlah time	TAT	Waiting
PI	3	8	j8-0) = B	0
P2	1	9	(9-1) = B	(8-1)=7
P3	2	12	(12-2) = 10	(9-2)=7
P4	3	14	(14-3) = 11	(12-3)=
P5	4	20	(20-4) = 16	(14-4)=
Average			(53/5) = 106	(33/5)=

(ii) SJF(Preemtive)

P1	P2	P3	P4	P5	Pi

0 1 2 5 7 13 20

Process	Priority	Finish time	TAT	Waiting
PI	3	20	(20-0) = 20	(13-1)J
P2	1	2	(2-!) = l	(1-lWJ
P3	2	5	(5-D) = 3	(2-a i
P4	3	7	(7-3) = 4	(5-g |
P5	4	13	(13-4> = 9	(7- d
Average			(37/5) = 7.4	(17/5) |

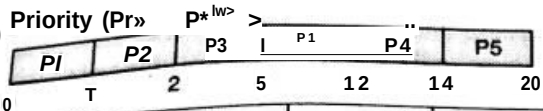
(iii) SJF(Nonpreemtive)

P2	P4	P3	P5

0 8 20

Ptocwm	priority	Finish tin»	TAT	Watting Unw
P1	3	8	(&—0) - B	0
P2	1	9	(9 - i) - a	(8 - 1}*7
P3	2	12	(14 - 2) = 12	(11 - 2) = 9
P4	3	14	(11 - 3) = 8	(9 - 3) = 6
PS	4	20	(20 - 4) = 16	(14 - 4) = 10
Average			(52/5) = 10.4	(32/5) = 6.4

(iv) Priority (Pr» P* lw> >



Proc*®*	Prtorfty	Finish time	TAT	Waiting time
PI	3	20	(12-0) = 12	(5-1) = 4
P2	1	2	(2-1) = 1	(1-1) = 0
P3	2	5	(5-2) = 3	(3-2) = 1
P4	3	7	(14-3) = 11	(12-3) = 9
P5	4	13	(20-4) = 16	(14-4) = 10
Average			(43/5) = 0.6	(23/5) = 4.6

Example 2 J 7.9

Find AWT, ATAT, ART and AWTAT for the following set of processes with CPU burst time in ms. Assume that all processes arrive at time 0.

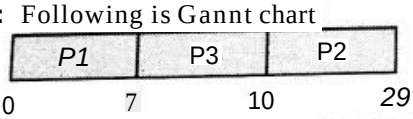
(P1-19), (P2-7), (P3-3)

i) FCFS with order P2,P3, P1

ii) Round Robin (Quantum = 2ms)

Solution :

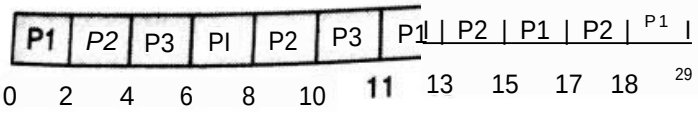
(i) **FCFS :** Following is Gannt chart



I Process	TAT	Waiting time
PI	(29 - 0) = 29	(10-0)3=10
1 P2	(7-0)»7	0
j P3	(10-0) = 10	(7-0) = 7
1 Average	46/3 = 15.33	17/3 = 5.66

(ii) Round Robin (RR) with quantum = 2

Following is Gannt chart



Proc+ca	TAT	Walting Uma
PI	(29-0) 29	(B■2) + (11 - B) +{16 - 13) +{16 - 17) . 10
PZ	(iB-o)-ia	(2 -0) +(6-4) +{13 -10) +(17 -15) =11
P3	(ii - q . it	(4-0) +(10-6)>B
Average	W3 ■ 18.13	1 29/3 =9.66

Example 2.17.10 MU -Dre. 2016. 10 Marks

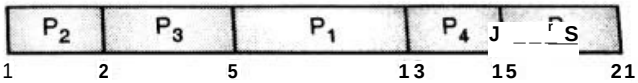
Assume the following processes arrive tor execution at the tune indicated and the length of epu burst time given in msec.

Job	Burst time	Priority	Arrival time
Pi	8	3	3
Pa	1	1	A
Ps	3	2	?
P4	2	3	3
P_5	6	4	*

For the above process parameters, find average waiting times and average turnaround times for the following scheduling algorithms. First Come First Serve, Shortest Job First, non preemptive priority and Round Robin (assume quantum = 2 Units)

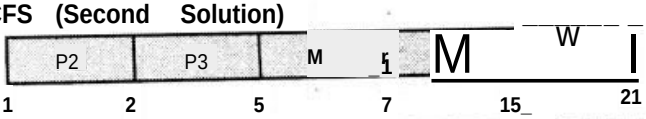
Solution :

FCFS (First Solution)



Job	Priority	Arrival Time	Finish time	TAT	Waiting time
PI	3	3	13	(13=3)= 10	(53) = 2
P2	1	1	2	(2-1)- 1	(1-1) = 0
P3	2	2	5	(5-2)=3	(2-2)=0
P4	3	3	15	(15-3)= 12	(13=3)= 0
P5	4	4	21	(21-4)= 17	(15-4)= 11
Average				(43/5)= 8.6	(23/5)= 4.6

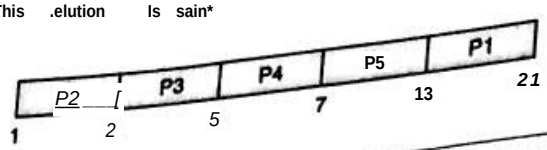
FCFS (Second Solution)



Job	Priority	Arrival Time	Finish time	TAT	Wattingtime
PI	3	3	15	(15-3)=12	P-3) -4
P2	1	1	2	(2-1)=1	(1-1)=0
P3	2	2	5	(5-2)=3	(2-2)=0
P4	3	3	7	(7-3)=4	(5-3)=2
P5	4	4	21	(21-4)=17	(15-4)=1
Average				(37/5)= 7,4	(17/5)=3

preemptive : SJF : for SJF preemptive)

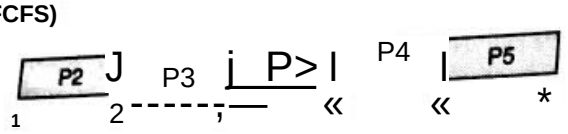
No"
(This .elution Is sain*



Job	Priori*	Anri* TW*	flnW> time	TAT	Waiting time
PI	3	3	21	(2f4) = «	(13-3) ■ 10
P2	1	1	2	(2-1)= 1	J1-D = 0
P3	2	2	5	(S2)-3	(2-2) a 0
P4	3	3	7	(7-31-4	(5-3) = 2
P5	4	4	13	(1M*9	(74) - 3
				ra&S)-?	(15/5J-3

a*w
Nonpmwn PHority: (Samea.fi— of ptive

FCFS)



Job	Priority	Arrival Time	Flniah time	TAT	Waging time
PI	3	3	13	(13-3)= 10	(5-3) - 2
P2	1	1	2	(2-1)= 1	(1-1) - 0
P3	2	2	5	(5-2)= 3	(2-2) = 0
P4	3	3	15	(15-3)= 12	(13-3) = 10
P5	4	4	21	(21-4)= 17	(15-4) = 11
Average				(43/5) =8.6	(23/5) = 4.6

Round Robin - quantum - 2 units

Job	Priority	Arriwl Time	Anieh time	TAT	Waiting time
PI	3	3	21	(21-3) = 18	(4-3) + (11-6) + (15-13)+ (19-15) = 12
P2	1	1	2	(2-1) = 1	(M) 0
P3	2	2	11	(11*2)= 9	(2-2) + (104) = 6
P4	3	3	a	(3-3)= 5	(6-3) = 3
P5	4	4	19	(19-4)= 15	(8-4)) + (13-10)+(17-15) = 9
Average				(48/5)= 9.6	005) =6

Example 2.17.11

Consider followin a set of Passes, with |en 9th or
burst given in millis eoorKfe as follows -

Process	Burst Tim*	Arrival Time	Priority
P1	8	0	3
P2	1	1	1
P3	3	2	2
P4	2	3	3
P5	6	4	4

(i) Draw Gantt chart for FCFS, SJF, Preemptive Pri and RR (a uanturn 2)

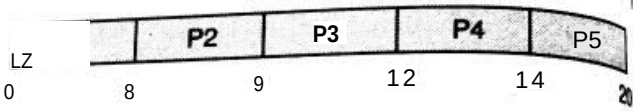
(«) What is the turnaround time of each proceas for abo algorithm'?

(hi) wat is waiting time of each process for eatb dl above algorithms?

(iv) Which algorithm results in minimum ave rage wa time?

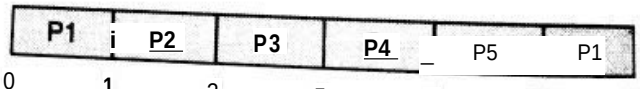
Solution :

FIFO



Process	Priority	Finish time	TAT	Waiting time
P I	3	8	(8-0) = 8	0
P2	1	9	(9-1) = 8	(8-1) = 7
P3	2	12	(12-2) = 1Q	(9-2) = 7
P4	2	14	(14-3) = 11	(12-3) = 9
P5	4	20	(20-4) = 16	(14-4) = 10
Average			(53/5) = 10.6	(33/5) = t6

SJF (Preemtive)



Proce.1	Priority	Finish time	TAT	WaNngta
PI	3	20	(20-0) = 20	(13-1)=12
P2	1	2	(2-11= 1	(1-11=0
P3	2	5	(5-2) = 3	(2-2)=0
P4	3	7	P-3) = 4	(5-3) =2
P5	4	13	(13-4) = 9	(7-4).3
Average				

JjHN nP ,t,Ve)

P1	P2	P4	P3	P5
0	8	9	11	14	20

Process	ftforW	Finish lime	TAT	Waiting time
P1	3	8	(8-0) - 8	0
fe-—	1	9	£9- 1) = 8	(8-1) » 7
P3	2	12	(14-2) = 12	(11-2) = 9
P4	3	14	(11-3H8	[illegible]
Pfi	4	20	(20-4) = 16	(14- 4) = 10
Avsmg®			(52/5) = 104	(32/5) = 6.4

priority (Preempt)

rp*	2	P3	P1	P4	P5
0 1	2	5	12	14	20

Procs*	Priority	Finish time	TAT	Waiting time
PI	3	20	(12-0) = 12	(5 - 1) = 4
P2	1	2	(2-1) = 1	(1-1) = 0
P3	2	5	(5 - 2) = 3	(3 - 2) = 0
P4	3	7	(14 - 3) = 11	(12 - 3) = 9
P5	4	13	(20-4) = 16	(14-4) = 10
Average			(43/5) = 8.6	(23/5) = 4.6

RR (Quantum 2)

P1	P2	P3	P4	P5	P1	P3	P5	P1	P5	P1
0	2	3	5	7	9	11	12	14	16	18	20

Praceu	Priority	Finish time	TAT	Waiting time
P1	3	20	(20-0) = 20	0+(9'2)+(14-1lH18-16) = 12
P2	1	3	(3-1) = 2	(2-0 = 1
P3	2	12	(12- 2) = 10	(3-2M11-5) = 7
P4	3	7	(7-3} = 4	(5-3) = 2
P5	4	18	(18-4) = U	(7-4)+(12-9)+(16-14) = 8
Average			(50/5) = 10	(30/5) = 6

Preemptive SJF g lives minimuia average waiting lieie.

Example 2.17.12 MU - Noy. 2015. 10 Marks

Consider following set of processes with their CPU burst time.

Process	Burst time	Arrival time
P1	10	1
P2	4	2
P3	5	3
P4	3	4

(i) Draw Gantt chart FCFS, SJF preemptive and round robin (quantum w 3), Calculate average waiting time and average turnaround time.

(ii) Explain which scheduling policy adopted by Linux?

Solution :

Process	TAT	Walting time
PI	(23- 1) = 22	(14-2) = 12
P2	(6 -2) = 4	0
P3	(14-3) = 11	(9-3) = 6
P4	(6 - 4) = 2	(6-4) = 2
Average	42/4=10.5	20/4 - 5

(i) FCFS Scheduling

P1	P2	P3	P4
1	11	15	20	23

Process	TAT	Waiting time
Pl	(11- l)=10	0
P2	(15-2) = 13	(11-2) = 9
P3	(20 - 3) = 17	(15-3) = 12
P4	(23-4) = 19	(20-4) = 16
Average	59/4=14.75	37/4=9.25

(ii) SJF Preemptive Scheduling

P1	P2	P4	P3	P1
1	2	6	9	14	23

(iii) RR Scheduling with quantum = 3

P1	P2	P3	P4	P1	P2	P3	P1	P1
1	4	7	10	13	16	17	19	22	23

Process	TAT	Waiting time
P1	(23- 1)=22	(13 – 4) + (19 – 16) = 12
P2	(17-2) = 15	(4 - 2) + (16 – 7) = 11
P3	(19- 3) = 16	(7 - 3) + (17 – 10) = 11
P4	(13-4) = 9	(ro- 4) = 6
Average	62/4 =15. 5	40/4= 10

Syllabus Topic : Thread Scheduling

2.78 Thread Scheduling

2.18.7 Contention Scope

Q. Write note on thread scheduting J

- Scheduling of the user level and kernel level threads is different.
- *If* system implements many to one or many to many model of multithreading then thread library schedules user-level threads to run on an available LWP.
- This scheme of scheduling is called as process-conteiftion scope (PCS) as threads belonging to the same process competes for processor.
- The operating system has to actually schedule the kernel thread on physical processor to run a user thread scheduled by thread library on available LWPs.
- Kernel uses system-contention scope (SCS) to schedule the threads on processor when there are many threads competing for the physical processor.
- Windows XP, Solaris 9; and Linux use one-to-one model and schedule threads using only system-contention scope.
- Usually, PCS is carried out in keeping with pr,ont y- rhe scheduler chooses the ses the 7 runnable thread with u highest *priority* to run.
- Programmers set the priorities nr level threads whicharenotaheredbyrhXa .r some thread libraries may permits the alter the priority of a thread. Prammer to
- The KS Wi « aatura lly preempt the thread Present for a higher-priority thread- On th-- running " there is no assurance of time slid ' --e other hand > equal priority. --ng amon g threads of

2.18.2 pthread Scheduling

Following are the pthread ""«""»■> value,

- PTHREAD_SCOPE_PROCESS is used to threads using PCS scheduling.
- PTHREAD-SCOPEJSYSTEM is Used to threads using SCS scheduling.

The system that implements many to many model PTHREAD_SCOPE„PROCESS to schedule thread available light weight processes. Thread library ma the number of LWPs. Scheduler activations c this purpose. The PTHREAD_SCOPE jy s s and bind an LWP for each user-level thread on many systems, successfully mapping threads by the one-to-one policy.

In order to get and set contention scope Pthread IPC offers following two functions two

- pthread_attr_setscope(pthread_attr_t *attr int
- pthread_attr_getscope(pthread_attr_t *attr t

Syllabus Topic : Multiple Processor Scheduling

2.19 Multiple-Processor Schedmin,

Q. Explain multiprocessor schedulingh f

Here we assume the homogeneous multiorom processors are identical.

2.19.1 Approaches to Multiple-Processor Schedu hng

- In asymmetric multiprocessing, all sd decisions, I/O processing, and other system t are carried out by master processor.
- AH other processors execute in user mode.J system data structures are accessed by single J Pressor leading to minimizing lhe data shanog l
- In symmetric multiprocessing, scheduling is earned' by each processor.

p'«X V h "" e can hc maintained * Processor may have its p queue for ready processes.

- All modern operating system supports symm multiprocessing

2.19.2 Processor Affinity

- Some systems also s "PP° n a* affinity scheduling in Which same thread again and again gets scheduled on same CP U. Tins makes better use of previously cached blocks and increases the cache hits.
- Also, the TLB may also have the right pag es , reducing 7L0 faults. For creation of affinity a two-level scheduling algorithm is used.
- After a ttread 1S created, it is assigned to a lightly j ded CPU which is a top level of the algorithm. It ensures that, each CPU acquires its own collection of threads*
- The bottom level of the algorithm performs real scheduling- It is carried out by each CPU separately, on the basis of priorities or some other means.
- By trying to keep a thread on the same CPU for its entire lifetime, cache affinity is maximized.
- *In case* if a CPU has no threads to run, it takes one from another CPU instead of remaining idle.
- Approximately equal load distribution over the available CPUs, cache affinity and minimizing contention for the ready lists as CPU frequently uses same ready list are three benefits offered by two-level scheduling algorithm.
- Soft affinity means same process will be scheduled on same processor and keep running on it. This can be policy but cannot be guaranteed by OS.
- Process migration on other CPU may happen. In hard affinity process can not be migrated to other processor.

2.19.3 Load Balancing

- The main objective of load balancing in symmetric multiprocessing systems (SMP) is to keep the load balanced among all the processors.
- Load is evenly distributed among all processors. Otherwise some processors may remain idle and other will be heavily loaded.
- In push migration approach of load balancing, a specific process periodically check load of processors.

If it finds the processor with light load then it migrates process from heavily loaded processor to lightly loaded or idle processor.

In pull migration approach, idle processor pu lls the waiting task from heavily loaded processor.

2.19.4 Multicore Processors

- Now days, it is possible to place multiple processor cores on the same physical chip.
- This is called multicore processor, Each of the cores is treat e as separate physical processor by OS as there is register in core to set its architectural state.
- SMP systems using multicore processors consume less power and efficient.
- Scheduling is complex for multicore system. The processor accessing memory has to wait for data as it is not available. So processor spends 50 % of its time in waiting.
- As a solution over this, hardware designer provided two or many hardware threads for each core.
- If one thread has to wait for memory then core switch to other thread. OS treats each hardware thread as logical processor for S/W thread.
- The course grained and fine grained multithreading can be used to multithread a processor.
- Multithreaded multicore processor requires two level of scheduling. OS schedules software thread on hardware thread using any scheduling algorithm.
- Second level of scheduling involves how each core make a decision about which hardware thread to execute.

2.19.5 Virtualization and Scheduling

- With virtualization, single CPU system works Like multiprocessor system.
- The virtualization software offers one or more virtual processors to each of the VMs running on the system and then schedules the use of the real CPUs among the VMs. VMs are created and managed by host operating system. Each VM has guest OS and applications running within it.
- The guest OS scheduling algorithm that expects a convinced amount of progress in a given amount of time will be negatively impacted by virtualization.

2.19.6 Other Multiprocessor Scheduling Approaches

Following are the four general approaches used for scheduling of threads on multiprocessor system.

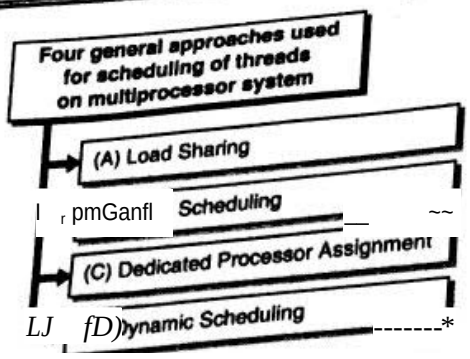


Fig. C2.13: Approaches used for scheduling of threads on multiprocessor system

➜ ***(A) Load Sharing***

All the ready threads are present in single global queue and each processor selects thread from this queue when it is idle.

\+ ***(B)* Gang Scheduling**

A set of related threads is scheduled to execute on a set of processors all at once, on a one-to-one twb.

➜ **(C) Dedicated Processor Assignment**

In this approach, during execution of program the number of processors assigned is equal to number of threads available in program. All the processors returns to *pool of* processors after program complete the execution.

➜ (D) **Dynamic Scheduling**

The number of threads tn process may changes during the execution of that process.

\+ *2.1* 9.6(A) Load Sharing

Advantages

- In this approach, load is equally distributed among all the processors so that no one remains idle.
- There is no need of centralized scheduler The scheduling routine of the operating system executes on available processor to choose the next thread.
- The maintained global queue can be accessed by using schemes used in uniprocessor scheduling such J based schemes or schemes based on execution Z''

Load sharing approaches

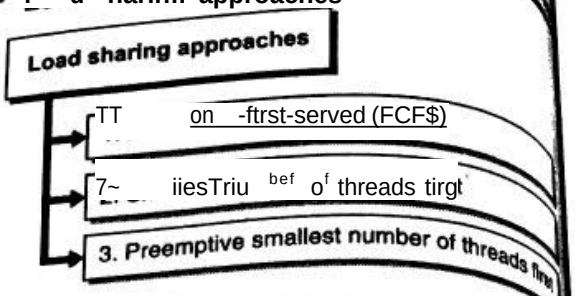


Fig. C2.13 Load sharing approaches

! irs»-« e-n -ser ,,ed (FCFS)

The threads of the newly arrived job are placed consecutively at the end of shared queue processor selects next ready thread and executes it finishes execution or gets blocked.

➜ **2. Smallest number of threads first**

If the jobs are having smallest number of unscheduled threads then highest priority is given to it. The ready queue is like priority queue and equal priority jobs are ordered according to jobs arrived first to FCFS, thread executes until it finishes execution or gets blocked.

➜ **3. Preemptive smallest number of threads first**

Job having smallest number of unscheduled threads gets highest priority. If an arriving job is having smaller number of threads with compare to executing job it preempts the threads of scheduled job.

Disadvantages

- A single shared ready queue may occupy the memory portion and if many processors simultaneously access it then it may become bottleneck.
- Preempted threads may not get same processal resume execution. The caching becomes lessefficJ processors have cache memory.
- It is unlikely to get access to all the threads J program at the same time. If more coordination equired between thread then process switch mayis to poor performance.

■* 2.19.6(B) Gang Scheduling

Advantages

- As synchronization, blocking can be minimized less process switching can be achieved by running related

performance * parallel there id more incltaii

- Scheduling overhead can be minimized as a sinele decision affects a number of processors and processLs at one time.
- Gang scheduling is better for the application whose any part is waiting for other part is ready for execution.
- It is implemented in many Processor operating systems- rn this scheduling process switches are minimized and threads ran in parallel. Hence, performance is good.
- As threads that require coordination run in parallel, they can access file without additional overhead and resource allocation can be done with less overhead.
- In this scheduling processor allocation is required. If N processors and M applications with N or fewer threads are present then each application could be given 1/M of the available time on the N processors, using time slicing-

2.19.6(C) Dedicated Processor Assignment

- In this approach, during execution of program the number of processors assigned is equal to number of threads available in program.
- All the processors returns to pool of processors after program complete the execution. In this approach, if thread of an application is blocked waiting for I/O or for synchronization with other thread then processor remains idle. It can be addressed as following :
  - If tens or hundreds of processors present in system then processor utilization does not matter for performance,
  - As process switching is avoided, it results in better improvement in performance,

2.19.6(D) Dynamic Scheduling

- In some applications, the number of threads changes during execution.
- The operating system then can adjust the load to improve performance. Both OS and application can carry out scheduling task.
- The operating system partitions the processors to allocate to the jobs,
- The runnable task of each job is mapped to threads and is given to the processors for execution at present in its partition,

A proper decision, about which subset to run, in addition to which thread to suspend when a process i preempted, is left to the individual applications (maybe throu&h a set of nm-time library routines).

- This approx may be appropriate for some applications and not for all. Applications can be implemented to take advantage of this feature of operating system.

Operating system is only responsible to processor a ocatinn and performs following tasks when job demands one or more processors.

- Allocate the idle processor to satisfy the request.
- If job is new arrival and need processor then take the processor from currently allocated job to which more than one processor is allocated and allocate to newly amved job.
- In case if jobs request cannot be satisfied then it remains outstanding until processor becomes available or the job withdraws the request.

After release of one or more processors

- Search the queue of requests that are mot satisfied for processors. Allocate a single processor to each job in the list that are waiting new arrivals and currently has no processors.
- Then search the list again, allocating the remaining processors on an FCFS basis.

2.20 Exam Pack (University and Review Questions)

Syllabus Topic : Process Concept

Q. What do you mean by process ?
(Refer section 2.2) **(3 Marks)** **(Dec. 2014)**

Q. Explain the concept of process,
(Refer section 2.2)

Q. What is context switch?
(Refer section 2.3)

Syllabus Topic : Operation on Processes

Q. What are the operations performed on process? *(Refer section 2.4)*

Q. Explain process creation.
(Refer section 2.4. 1)

Q. Explain process termination
(Refer section 2.4.2)

Q. Explain role of process control block.
(Refer *section 2.5)* (5 Marks) (Dec. 201 4)

Q. Explain process control blocn. *(Refer sect fen 2. 5)*

(J. Draw and explain process state tra nsition diagram. *(Refer section 2.6)*

☞ ***Syllabus Topic : Process Scheduling***

Q. *Explain* long-term scheduler *(Refer section 2.8.1(A))*

Q. *Explain short-term scheduler. (Refer section 2.8.1(B))*

Q. ***Explain medium-term scheduler.*** *(Refer section 2.8. 1(C))*

Q. *Compare the functions of different types of schedulers. (Refer section 2.8, 1(D))*

☞ **Syliabus Topic : Multithreading**

Q, Explain user level *and kernel* level threads with advantages and disadvantages.

(Refer section 2. 11)

☞ **Syllabus Topic : Process - Multithreading *Models***

Q. *Explain various multithreading models. (Refer section 2. 12)*

☞ ***Syllabus Topic : Thread* Libraries**

O. Write note on thread libraries.

(Refer section 2. 13)

☞ ***Syllabus Topic : Threading* Issues**

Q- What are different threading issues?

(Refer section 2. 14)

☞ **Syllabus Topic : Process Sch T uling : Basic Concepts**

Q. What is difference between preemptive and preemptive scheduling? *(Refer section 2.15.2)*

☞ **Syllabus Topic : Scheduling c rite*1***

Q. What are the criteria tor eva luatio algorithm performance? fR e fer >I sche ction 2.16.

☞ **Syllabus Topic : Scheduling a10 rl %u**

Q. Explain FIFO scheduling algorithm *(Refer section 2.16.2(A))*

Q. Explain SJF and SRTN sghe duling algorit *(Refer section 2.16.2(B))*

Q. Describe priority scheduling algorithm *(Refer section 2. 16.2(C))*

Q. With the help of example, explain Scheduling algorithm. *(Refer section 2* Round R .16.2(D))

Q. Explain multilevel queue scheduling algo *(Refer section 2. 16.2(E))*

Q. Explain multilevel feedback queue sche algorithm. *(Refer section 2.16.2(F))*

Example 2.17.1 (10 Marks) (Dec

Example 2.17.10 (10 Marks) (De

Example 2.17.12 (10 Marks) (He

☞ **Syllabus Topic : Thread Scheduling**

Q. Write note on thread scheduling. *(Refer section 2.18.1)*

☞ **Syllabus Topic : Multiple Processor Sc**

Q. Explain multiprocessor scheduling in deta *(Refer section 2. 13)*

CHAPTER 3

Process Coordination

[Module in

Syllabus Topics

Synchronization : The critical Section P hi Peterson's Solution, monitors, Atomic trar rization, Methods fo tion, Recovery from Deadlock. S-Z

Syllabus Topic ; **Synchro nization**

3,1 Background

➔ (May 16)

Q. Explain the process synchronization in brief.

MU - May 2016, 5 Marks

- In a single, proccssor multiprogramming system. processes are not executed concurrently. In order to get the appearance of concurrent execution, a fixed time slot is allocated to each process. After utilization of this time slot, CPU gets allocated to other process. Such switching of CPU back and forth between processes is called as context switch.
- At a lime single process gets executed so parallel processing cannot be accomplished. Also there is a definite amount of overhead drawn in switching back and forth between processes. Apart from above limitations, interleaved execution offers major benefits in processing efficiency and in program structuring.
- In a multiple processor system, interleaving and overlapping the execution of multiple processes is achievable. Both interleaving and overlapping correspond to basically diverse modes of execution and present diverse problems. In reality, both interleaving *and* overlapping can be treated as illustration of concurrent processing and both the techniques address the similar problems.
- The comparative speed of execution of P -s depends on activities and behavior of other process . how the operating system handles interrupts, and the scheduling policies of the operating system.

☞ **The difficulties that arise**

1. Global resources sharing. For example, suppose two processes use the same global variable simultaneously and both carry out read and write operations on that variable, then various reads and writes execution ordering is serious.
2. Optimal management of the resources is not easy for the operating system.
3. Programming error tracing is not easy as results arc typically non- deterministic and not reproducible.

Only a single user is supported by single processor multiprogramming system at a lime. While working on one application, user can switch to other application in this system. The keyboard for input purpose and screen for output purpose used by each application is same. The reason is, each application needs to use the procedure echo. As echo procedure is shared by each application, memory can be saved by keeping single copy of procedure in memory portion which is global for all sharing applications.

☞ **Example**

```
void echof)
{
        chinput = gelcharQ;
        choutput = chinput;
        putchar(choutput);
}
```

In order to have efficient and close interaction among processes, sharing memory among processes is useful. Consider the following sequence :

I- initially process P_1 calls the echo procedure. The getchar() returns its value and stores in input variable chinput. Process Pi gets interrupted straight away after returned value is stored in chinput. At this instant, the most recently entered character, m, is stored in variable chinput.

2. After P_1 gets interrupted, there is turn of process P_2 and it call up the echo procedure. Process P_2 inputs and displays the single character now on monitor.

3. now again, process P_1 resumes its execution and the value in chinput holds value n, which is transferred variable choutput and displayed, result in

4. In this way, the first character m is lost and the second character n is displayed two times. All this happens due to shared global variable, chinput. If after updating the shared global variable, one process is interrupted, another process may modify the variable before the interrupted process can use its value. In above example both P_1 and P_2 are allowed to use shared global variable chinput. However, if only one process at a time is allowed to be in procedure, above discussed sequence would result in the following :

(i) Initially process P_1 calls the echo procedure. The getchar() returns its value and stores it in input variable chinput. Process P_1 gets interrupted straight away after returned value gets stored in chinput At this instant, the most recently entered character, m, is stored in variable chinput.

(ii) After P_1 gets interrupted, there is turn of process P_2 and it call up the echo procedure. At this situation, process P_1 is in suspended state and it is still inside the echo procedure. The process P_2 is not allowed to enter inside the echo() procedure. Therefore, P_2 is suspended until P_1 comes out of the echo procedure.

(iii) Later on, process P_1 is resumes and finishes the execution of echo procedure. The output displayed is the character m.

(iv) When P_1 comes out of echo() procedure P_2 becomes active. Now process P_2 can successfully call the echo() procedure.

Therefore, if is necessary to use shared by only one process at a time.

3.2 Interprocess Communication

a Explain the inter-process communication in brief. MU - May 2016, 5 Marks

Communication and communication are two Requirements should be satisfied when processes communicate with each other. Synchronization passes is required to achieve the mutual exclusion Independent processes do not communicate with other but cooperating processes may need to exchange information. Cooperative processes communicates through shared memory » message passing.

Cooperating processes require an Interprocess Communication (IPC) mechanism that will allow them to exchange data and information. There are two fundamental models of interprocess communication :

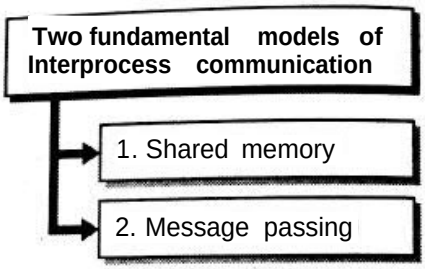


Fig. C3.1 : Fundamental models of interprocess communication

1. Shared memory

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

In the message passing model, communication takes place by means of messages exchanged bet## cooperating processes.

1 Message Passing

Message passing provides both functions. Message passing has the further benefit that it lends itself to implementation in distributed systems as well as shared-memory multiprocessor and uniprocessor systems.

- Follow, ng are the two primitives used in messa- passing :
  - send (destination, message)
  - receive (source, message)

- This is the minimum two operations required for processes to send and receive the messages. A process sends data in the form of a message to another process indicated by a destination. A process receives data by executing the receive primitive, indicating the source and the message.
- Communication by sending and receiving messages require synchronization. The receiver cannot receive a message until it has been sent by another process.
- The sending process is blocked until the message is received, or it is not after the send primitive executed by process. Similarly, when a process issues a receive primitive, there are two possibilities :
  - Previously sent message is received and execution continues.
  - If there is no waiting message, then either (a) the process is blocked until a message arrives, or (b) the process continues to execute, abandoning the attempt to receive.
- Thus, both the sender and receiver can be blocking or nonblocking. Three combinations are common, although any particular system will usually have only one or two combinations implemented :
  - **The blocking send and blocking receive :** Until the message is handed over, the sender and receiver both gets blocked.
  - **The nonblocking send and blocking receive :** After sending the message, the sender continues its work but receiver remains blocked until message is arrived to it. This permits a process to send one or more messages to a multiple destinations as quickly as possible. Here receiver is in need of message so that it can resume the execution . So it gets blocked until message arrives.
  - **Nonblocking send, nonblocking receive :** Neither party is required to wait.
  - **The nonblocking send and nonblocking receive:** Both sender and receiver wait will continue the work.

Message passing system should give guarantee that messages will be correctly received by receiver. Receiver sends acknowledgement to sender after receiving the message, if acknowledgement not received in defined time [1] en sender resend the message. It also offers authentication service.

3.3 Race Condition

Q- What Is race condition? Explain with example.

A race condition takes place when more than one process write data items so that the final result depends on the order of execution of instructions in the multiple processes. Consider two processes, P_s and Pj, which share the global variable "x". At the time of execution, process P_t updates x to the value 1 and at the time of execution of another process, P_2 updates "x" to the value 2. Thus, the two tasks are in a race to write variable "x". In this example the process that modifies x value lastly decides the final value of"x".

Role of Operating System

1. The operating system should be capable of keeping track of the different processes.
2. Allocation and reclamation of various resources should be done by operating system.
3. Each process's data and physical resources should be protected by operating system against unintentional interfering by other processes.
4. The operations performed by the process and the output that it generates should not depend on the speed of execution compare to the speed of other concurrent processes.

Definition of process interaction

Process Interaction can be defined as :

- The processes not aware of each other
- The processes in a straight way or indirectly aware of each other

The conflict occurs between the concurrently executing processes. if these processes compete for the use of the same resource. More than one process may require the same resource while they are executing. The existence of the particular process is not known to other process. The competing processes do not pass the information to each other.

Syllabus Topic : The Critical Section Problem

3.4> TheCritJcal SectJonProb 161,

(Dec. 14, June 15)

Q. al section problem. MU - Dc c. 2

Q. Explain cndca) section problem with its different solutions. MU - June on15 10 Marks

As discussed previously, it i5 necessary <o «"<' ou , some means *to disallow more* th® one process from reading and writing the shared data at the same time. The po n,on of the program where the shared memory is accessed is referred as the *Critical Section*. In order to avoid race conditions and ««>«. «* mus be able to reco nize the codes in *Critical Sections* in each thread. The type p mX of the code that comprises *Criticai Section* am as follows:

- These codes reference one or more variables in a “re - update-write" way. At the same time, any of those variables maybe changed by another thread.
- These codes modify one or more variables that can be referenced in “read- update-write” fashion by another thread.
- Any part of data structure used by the codes can be modified by another thread.
- Codes modify any part of a data structure that is currently in use by another thread.

When one process is currently executing shared modifiable data in its critical section, no other process is to be permitted to execute in its critical section. Hence, the execution of critical sections by the processes is mutually exclusive in time.

In a code called as critical section, only one process executes at a time. In a critical section problem an algorithm needs to be designed which permits at most one process tnto the cntical section at a time, with no deadlock

" ciidcal section's problem solution must obey mutual exclusion, progress and bounded waiting. Any process Erectly cannot enter in critical section. First process has to obtain permission for its entry in its critical section. The segment of code which implements this appeal of proce ss f s the entry section. When process conies out of the critical section after completing its execution there it has to execute exit section. The rest of the code is the remainder section.

Any solution to the critical-section problem satisfy the following three necessary conditions :

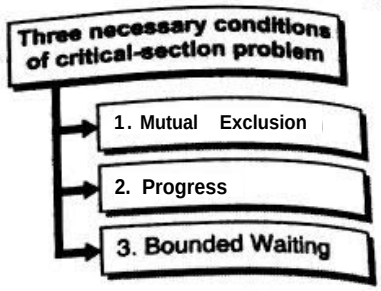


Fig . C3-2 : Conditions of critical section proble

i. Mutual exclusion

At a time, single process should be executing in section (CS). K currently Process P_5 i$_a$ e xecutin CS then other processes should not execute critical sections.

2. Progress

If process's CS is free, means it is not exe 1 Th CS. In this situation suppose some processes J process go into their critical sections for execution, foj executi take the decision about who will now entern j with e only processes that are not executing in their paralk sections are allowed to take part to make this dec This decision should be taken in some definite and should not delayed indefinitely,

3. Bounded waiting

A limit should be set on the number of times lbw processes are permitted to go into their critical sc after a process has made a request to go into its c section and before that request is approved.

Semaphore and monitor are the solutions to i mutual exclusion.

A synchronization variable taking positive ina values is called semaphore. Binary semaphore only values 0 or L Hardware does not supply the semaiJ Semaphore offer the solution to critical section ptoM semaphore variable takes value greater than 1 then W called as counting semaphore. Like semaphore, a also solves critical section problem. It is a component which contains one or more procedut l initialization sequence and local data. Following ri components of monitors :

- Shared data declaration
- Shared data initialization
- Operations on shared data
- Synchronization statement

3,5 Mutual Exclusion

➔ (June 15, Nov, 15, May 16)

q What is mutual exclusion ?
MU - June 2U15- Nov 2015. 5 Rfiarks

Q. What is mutual exclusion ? Explain its significance. MU - May 2016, 5 Marks

At the same time as one process executes the shared variable, all remaining processes wishing to accomplish so at the same instant should be kept waiting; when that process has over executing the shared variable, one of the processes waiting to perform so should be permitted to cany on . In this manner, each process executing the shared data (variables) keep outs all others from doing so at the same time. Only one process should be allowed to execute in critical section. This is called Mutual Exclusion.

The mutual exclusion is put into effect only if processes access shared changeable data - If during execution, the operations of the processes do not conflict with each other, they should be permitted to proceed in parallel.

<r Requirement of Mutual Exclusion

- It is obligatory to enforce a mutual exclusion: If many processes want to execute in critical section then only one process should be allowed to execute in that critical section among all processes that have critical sections for the same resource or shared object.
- If the process halts in its remainder section (other than critical section) then it is allowed to do so without interfering with other processes.
- If process is waiting to enter in critical section then it should not be delayed for an indefinite period: that is it should not lead to deadlock or starvation.
- If the critical section is empty (any process not executing in critical section), then if any process that requests entry to its critical section must be allowed to enter without delay.
- Assumptions about relative process speeds or number of processors are not made.
- Any process will not remain inside its critical section for indefinite time. It should stay t ere or a finite time only.

Mutual Exclusion Conditions

A race condition can be avoided if we disallow to two processes in their critical section at the same J me .

Following 4 conditions should hold to obtain a excellent solution for the critical section problem (mutual exclusion) :

1. Two processes should not be inside their critical sections at the same time.
2. Comparative speeds of processes or number of CPUs in the systems are not assumed.
3. Process outside its critical section should not block other processes.
4. Any process should not wait for longer amount of time arbitarily to enter its critical section.

Syllabus Topic : Petarson's Solution

3.6 Peterson's Solution

Q. Explain Peterson solution to mutual exclusion.
Q. Give software approach for mutual exclusion.

- Peterson's solution involves only two processes. It gives a good means of algorithmic explanation of work out the critical-section problem and exemplifies some of the difficulties concerned in designing software that addresses the requirements of mutual exclusion, progress, and bounded waiting.
- The two processes *Pi and Pj* perform alternate execution between their critical sections and remainder sections (remaining code of the process). Petersons solution requires the two processes to share two data items :

```
int x;
boolean y{2];
```

- The variable x decides who will enter in its critical section, if x=i, then process *Pi* is permissible to execute in its critical section. The y array is used to specify whether process w *ready* to enter its critical section. For example, if y[i] is true, this value indicates that *Pi* is ready to enter its critical section.
- In order to enter the critical section, process ft first sets yli] equal to tree and then sets variable x to the value J. Pi x =j because, if the other *(Pj)* process desires to enter the critical section, it can do so. If both processes attempt to enter simultaneously, x will be set to both t and j at approximately the same time. Only one of these assignments will last; the other will occur but will be overwritten straight away.

```
do {
    y fO= TRUE;
    x= i:
    while (ylfl && x== j);

critical Berben
    y[i] = FALSE;

remainder section
} while (TRUE*
```

The final value of x decides which of the two processes is permi... to cn(er its critical section first. We can prove that solution is correct by showing:

- Mutual exclusion condition is satisfied.
- The progress requirement is satisfied.
- Tbe bounded-warring requirement also achieved.

Mutual exclusion

- Every *Pi go into its* critical section if and only if either y(j] = false or x=i. Also note that, if both Pt and ft runs in critical sections simultaneously, then y[0] — y [1J =tme. It implies that *Pi* and *Pj* could not have successfully executed their while statements at about the same time, since the value of variable x can be either i or j but cannot be both.
- Hence, one of the processes say, *Pi* must have successfully executed the while statement, whereas *Pj;* had *to* execute at least one additional statement ("x== j").
- However, at that time, y [jJ = true and x == j, and this condition will continue as Jong as *Pi* is in its critical section. Therefore we can conclude that mutual exclusion is preserved.

Progress and bounded waiting

- Process *Pi* can be prohibited from entering the critical section only if it gets stuck in the while loop with the condition y Q] =tnte and x= j; this)oo p is the onl y one possible.

— If Pi is false, and *P,* can go t0 the critical section into ils eritjcaj section. If *Pj* has set *y[j] to* true and is also ninn- ...ng in its while statement, then either variable x — :

Depending on the value of x either P, ... enter in *the* critical section.

— i then pi * al entCr cri,ic «?\l

If X ft will enter the critical section.

fts critical section, it win res M e *' owjng ft to enter its critical section j

If *pj* resets y LU to tru 6 ' it must alsft

A ft does not change the value of A while executing the while statement.

Critical section (progress) after at most (bounded waiting).

Syllabus Topic = SynchronSg

3.7 synchronization Hardware]

— T iThardware ut*011 to mutual exclusio

- The simple hardware instructions that are ava many systems can be used successfully ih sol critical section problem. Hardware features ca any programming task easier and get better efficiency.
- If we restrict the interrupts from taking 1 shared variable was being modified, in J environment the critical -section problcmj solved.
- Many systems have special instructions called] Set Lock called as TSL instruction. It is having) "TSL ACC, X". In this instruction ACC is accu register and X is symbolic name of memory loc holds character to be used as flag.
- TSL is indivisible instruction means that it c interrupted in between. After instruction getsi following action take place.
  - o Content of X is copied to ACC
  - o Content of X becomes N

During execution of above two steps, TSL instr is not interrupted. If we assume value of X=N initiii meaning of N is critical region is not free and F w free. Consider the following steps to enter in critical re

1. TSL ACC, X (Content of X is copied to AC Content of X becomes N)
2. CMP ACC, "F* (See if critical region is ft*1
3. BU 1 (If critical region not free the back.)
4. Return (Return to caller and eriff)

Operatin

Following are

1. MOVX
2. Return

Considt

- Asquint

So stq execut

- *Pj* gets the Cl the ct

3. So

- Cons Ever sche and
- Aft- to e
- If be by
- Si C 1

progn requii busy

3.8

Q.

Following are the steps for exiting from critical region :

1. MOV X, "F* (F gets copied to X)
2. Return (Return to caller)

Consider the two processes *Pi* and *Pj*.

- Assume initially scheduler schedules *Pi* and X = "F*. So step 1 makes ACC= F and X= N. Here Pl after executing step 4, prepare to enter CR (critical region).
- *Pj* gets scheduled due to context switch before *Pi* enters the CR* *Pj* executes step 1 and X = N. In step 2, *Pj* fails the comparison and loop backs. It cannot execute step 3. So *Pj* cannot enter in CR as *Pi* is already in its CR.
- Consider *Pi* is again scheduled and executing in its CR. Even though context switch occurs and *Pj* gets scheduled at this moment it cannot enter in CR as ACC and X values are N (busy waiting).
- After completion of execution in CR, *Pi* executes step 5 to exit the CR ans X —**F.
- If *Pj* scheduled after this, it execute step I and ACC becomes F as X was F in above step. Now due to step I by *Pj* ACC = "F" and X= "F".
- Step 2 is executed by *Pj* and succeeds and it enters the CR.

This solution satisfies all conditions mutual exclusion, progress and bounded wailing. Since special hardware is required cannot be generalized to all machines. Also due to busy waiting, it is not efficient solution.

Syllabus Topic : Semaphores

3.8 Semaphores

Q, What is semaphore?

- E.W. Dijkstra (1965) abstracted the key notion of mutual exclusion in his concepts of semaphores.
- The solutions of the critical section problem represented in the section are not easy to generalize to more complex problems.
- To avoid this complicatedness, we can use a synchronization tool call a semaphore. A semaphore S is an integer variable that, apart from initialization, is accessed only through two standard atomic operations: wait and signal. These operations were firstly termed P (for wait) and V (for signal).

Definition

- **A semaphore S is integer variable whose value can be accessed and changed only by two operations wait (P or sleep or down) and Signal (V or wakeup or up). Wait and Signal are atomic operations.**
- **Binary Semaphores does not assume all the integer values. It assumes only two values 0 and 1. On the contrary, counting semaphores (general semaphores) can assume only nonnegative values.**
- **The wait operation on semaphores S, written as wait(S) or P(S), operates as follows :**

```
wait(S): IF S > 0
         THEN S: = S —1
         E15E (wait on S)
```

The signal operation on semaphore S, written as signal(S) or V(S), operates as follows :

```
aignal(S): IF (one or more process are waiting on S)
           THEN (let one of these processes proceed)
           ELSE S: = S + 1
```

- The two operations, wait and signal are done as single indivisible atomic operation. It means, once a semaphore operation has initiated, no other process can access the semaphore until operation has finished. Mutual exclusion on the semaphore S is enforced within **wait(S)** and **signal(S).**
- If many processes attempt a wait(S) at the same time, only one process will be permitted to proceed. The other processes will be kept waiting in queue. The implementation of wait and signal promises that processes will not undergo indefinite delay. Semaphores solve the lost-wakeup problem.

cr Disadvantages of Semaphore

1. Semaphores are fundamentally shared global variables.
2. From any location in a program it can be accessed during course of execution.
3. A lack of command over it or assurance of proper usage.
4. A lack of proper linking between the semaphore and the data to which the semaphore controls access.
5. They provide two functions, mutual exclusion and scheduling constraints

Mutex is the concept related to binary semaphore. A key difference between the two is that the process that locks the mutex (sets the value to zero) must be the one to unlock

Operating extern ir"?

i, me value to 1). In contrast, it is possihie for one for another to process to lock a binary «m.pb»« unlock it.

Syllabus Topic
Classic Problems of Synchronization

« -SS£ ,“

3.9.1 Producer Consumer Problem

➔ (Dec. 16)

Q. Explain an for producer-consumer U- Pff. 2016. 10 Marks

- inpnxlu cer-consu roblem as... T Tuial buffer is r p means that there is a finite bounded buffer. W available in , buffcr . While numbers of slots a" is full the producer producing the items, if .be buffer « process should be suspe • jBms from in the same ' ' “Ztsumer should be buffer, if it X -nsurttbatonlyone suspended. It necessary process at a time manipulates bUffer “ avoid race conditions or Jostupdates.

- *Slee* *UP system* calls is used in this situation to avoid race condition. Producer process should go to sleep when buffer is lull and consumer process should wale up when producer will put data in buffer. In the same way, consumer goes to sleep until the producer puts some data in buffer and wakes up. to consume this data.

- Here two processes, producer and consumer share a common, fixed-size (bounded) buffer. The producer puts data into the buffer and the consumer takes data out.

- Difficulty arises when the producer wants to put a new data in the buffer, but there is no space in buffer. That is, n is already full. Way out to this problem is that, producer goes to sleep and to be awakened when the consumer has removed data.

- It may also happen that, consumer wants to take out data from the buffer but buffer is already empty Way *to this* problem is that, consumer goes to sleep unt J the producer puts some data in buffer and take consumer up. wakes

3-8 Process C...

The code for the producer process can be m... follows :

```
while (true)
{
/* produce an item in nextProduced variable */
while (counter == ARRAY SIZE)
; /* do nothing */
array[in] = nextP«»diiced;
in = (in + 1) % ARRAY-SIZE;
counter++;
}
```

The code for the consumer process can be mo... follows :

```
while (true)
{
while (counter = - 0)
/* do nothing */
nextConsumed = array [out];
out = (out + 1)%AHRAY_SIZE ;
counter—;
/* consume the item in nextConsumed variable*/
```

- Producer and consumer functions are accu... considered separately, they may not J when executed concurrently.
- Race condition also occurs in this approach justi... came across in previous approaches.
- As constraint is not imposed on accessing the co... variable, the race condition is occurred.

3.9.2 Producer Consumer Problem Usin... Semaphore

➔ (Jun...

Q. Explain how Readers/writers problem cm... solved with semaphores ?

MU - June 2015. 10lto

- In previous section, we have seen to prob... consumer problem. The Solution to producer cons... problem uses three semaphores namely, full, empty mutex. 1

- The semaphore 'fuff is used for counting J SOts in the buffer that are full. The 'empty' counting the number of slots that are empty semaphore 'mutex' to make sure that the produc...

consumer do not access modifiable shared section of me buffer simultaneously.

☞ **Initialization**

- Set full buffer slots to 0.
  i.e. semaphore Full = 0.
- Set empty buffer slots to N.
  i.e. semaphore empty = N.
- For control access to critical section set mutex to I
  i.e. semaphore mutex = 1.

```
Producer ( )
WHILE (true)
    produce-item ( );
    P (empty);
    P (mutex);
    enter-item ( )
    V (mutex)
    V(fuil);
Consumer ( )
WHILE (true)
    P(fiill)
    P (mutex);
    remove-item ( );
    V (mutex);
    V (empty);
    conaume-Item (Item)
```

☞ **Busy Waiting**

■> **(Nov. 15, Dec. 16)**

Q. What do you mean by busy waiting? What is wrong with it ?

MU - Nov. 2015, Dec, 2016, 5 Marks

Busy waiting is the limitation of semaphore definition.

If critical section is not free (process is already executing in it) then other process trying to enter in critical section **should loop continuously in entry code. This busy waiting** (continual looping) is the problem as far as multiprogramming environment is considered. In this case a single CPU is shared among many processes. This busy waiting waste CPU cycles which would have been used by other processed effectively.

3.9.3 Readers/Writers Problem

Q. Explain reader-writer problem.

- **While defining the reader/writers problem, it is assumed that, many processes only read the file (readers) and many write to the file (writers). File is shared among a many number of processes. The conditions that must be satisfied are as follows :**
  - **Simultaneously reading of the file is allowed to many readers.**
  - **Writing to the Hie is allowed to only one writer at the same time.**
  - **Readers are not allowed to read the file while writer are writing to the file,**
- In **this solution, the first reader accesses the fde by** performing a down operation on the semaphore file. Other readers only increment a counter, read count When readers finish the reading, counter is decremented.
- When last one ends by performing an up on the semaphore, permitting a blocked writer, if there is one, to write. Suppose that while a reader is reading file, another reader comes along. Since having two readers at the same time is not a trouble, the second and later readers can also be allowed if they come down.
- After this assumes that a writer wants to perform write on the file. The writer cannot be allowed to write the file, since writers must have restricted access, so the writer is suspended. The writer will suspended until no reader is reading the file. If a new reader arrives continuously with very short interval and perform reading, the writer will never obtain the access of the file.
- To stop this circumstance, the program is written in a different way: when a reader comes and at the same time a writer is waiting, the reader is suspended instead of being allowed reading immediately.
- Now writer will wait for readers that were reading and about to finish but does not have to wait for readers that came along after it. The drawback of this solution is that it achieves less concurrency and thus lower performance. Following is the solution given

```
typedef int semaphore;
semaphore mutex = 1 * controls access to 'readcount'*/
semaphore file = 1;/* controls access to file ♦/
int readcount = 0;
void reader(void)
{
```

```
while (TRUK) 1 P rrpea. evermore */
down(&mu ir xl: /* get exclusivh m'ccbh «
readcountc = readcountc + 1;/* one reader more now */
first
if(rc . d™, nl = = l ) down(&fiH=/' if "..... .
reader... */
, . .u'reudeount'*/
up{*ntu<«)y* rrleiue exclude acces
r»d fik()/*nMd the file*/
downW) exclusive to ' coun */
ree.koi.nt = readreunt - ! ; / • « * fewer **
last
if (readout = = 0) upf&iW ia ,he
reader... */
upfOrmutextV release exclusive access to 'readcount' */
uee-datA-rradtV* noncritical region */
}
}
void wrilerf void)
(
while (TRUE) {/* repeat evermore */
(hink up data(); /* noncritical region */
down(&file);/* get exclusive access */
write_fileQ; /* write the file */
upl&file); /• release exclusive access */
}
}
```

3.9.4 Dining Philosopher Problem

-> **(Dec. 14, Nov. 15)**

Q- Explain dining philosopher problem and solution to it **MU - Dec 2014. Nov. 2015. 10 Marks**

Q. Explain Dining philosopher problem using semaphores.

- The problem is stated as : The work of five philosophers is thinking jinri patina a trtU« i *□ for them and all agreed that only food that contributed to their thinking efforts was spaghetti. Each philosopher requires two forks to eat spaghetti.
- The provision for eatin® d tounu table with a big •serning pot of spaghetti, five piates, and five fork. One plate is given to one phitosonher a -k: philosopher sits at assigned place at the table. He has to use two forts on either side of the plate. He takes and spaghetti-
- Algorithm should be designed w philosophers to eat. The algori thm must satisfy exclusion (two philosophers should fork at the same time) while av oidi S starvation. This problem exemplifies basic deadlock and starvation.
- The dini n 8 philosopher's problem c be example of problems related to coordination of resources, which may occur whei) includes concurrent threads of executj * n way, <h« Problem is a typical 'wtca X approaches to synchronization. *t

Solution using semaphores

- In the solution using semaphores, each philosopher pick and choose the fork on the left side <t[]ri fork on the right side. Once the Philos opher ends eating, the hvo forks are replaced rm the table
- This solution, unfortunately, leads to de i the philosophers are hungry simul taileo pick up the fork on their left, and right si de fork not be there. In this unseemly position, all philoso starve.
- Additional five forks (a more clean solution the problem or use of just one fork to eat the can be a solution. Following program shJ solution.

```
/• program dining philosophers ♦/
semaphore fork [5] = {1};
int i;
void philosopher (int i)
{
while (true) {
thiuk();
wait (fork[il);
wait (fork [(i+1) mod 5]);
eat();
signalffork [(i+1} m<K j 5 |(.
signal(fork(i]);}
}
void main()
```

```
{
    parbe  n (philosopher  (0), philosopher  (1),
    philosopher  (2), philosopher  (3),
    philosopher  (4));
}
```

– In another strategy consider the helper who only allows four philosophers at a time into the dining room. With at most four seated philosophers, at least one philosopher will have access to two forks. Following is rhe solution using semaphore.

```
/* program dining philosopher* */
semaphore fork[5] = {l};
semaphore room = {4};
int i;
void philosopher (ini i)
{
    white (true) {
    thinkQ;
    wait (room);
    wait (forkfij);
    wait (fork [(i+1) mod 5]);
    eat();
    signal (fork [(i+1) mod 5]);
    signal (forkfij);
    signal (room); }
}
void mainO
{
    parbegin (philosopher (0), philosopher (1),
    philosopher (2), philosopher (3),
    philosopher (4));
}
```

Syllabus Topic : Monitors

3.10 Monitors

Q. What is monitor? How it is used to achieve mutual exclusion? Explain.

– If «maphnre s fITC uwd incorrectly, k efln pnjducc >*»ninge ™ that arc hard to detect. These errors occur only (f some pabular execution sequences results and icsc sequences do not always occur. In order to handle such errors, researchers have developed high-level language constructs called as monitor.

– A monitor is a set of procedures, variables, and data structures that arc all grouped together in a particular type of module or package. Processes may call the procedures in a monitor if required, but direct access to the monitor's internal data structures from procedures declared outside the monitor is restricted to the processes. Following is the example of the monitor.

```
monitor example
integer i ;
condition e ;
procedure producer ( );
end ;
procedure consumer ( );
end;
end monitor;
```

– Monitors can achieve the mutual exclusion: only one process can be active in a monitor at a time. As monitors are a programming language construct, the compiler manages the calls to monitor procedures differently from other procedure calls.

– Normally, when a process calls a monitor procedure, if any other process is currently executing within the monitor, it gets checked. If so, the calling process will be blocked until the other process has left the monitor. If no other process is using the monitor, the calling process may enter.

☞ Characteristics of a monitor

– Only the monitor's procedures can access the local data variables. External procedures cannot access it.

– A process enters the monitor by calling one of its procedures.

– Only one process may be running in the monitor at a time; any other process that has called the monitor is blocked, waiting for the monitor to become available.

Conditional variables are contained within monitors. These conditional variables can be accessed only within the monitors and used to achieve the synchronization. Functions cwait () and csignal() carry out the operation on conditional variable and these are created as special data type in monitors.

execution nf the caJW

!. **emdt (e) :** On * c the other process can BO W process gets poised. Afte access the monitor.

2. **csignal (0 » B ? e " d P s resumes it,** blocked. Now s will be selected to resume execution. Only one processes, were blocked. No action will be taken If there is no such process.

the execution in case it ...any such

Monitor wait and signal operations are dissimilar from those for the semaphore. If a process in a signals and no task is waiting on the condition variable, the signal is lost-

- When one process is executing in monitor, precesses tat trying to enter the monitor join a queue processes blocked waiting for monitor avmlabthty.
- Within the monitor process may provisionally block itself on condition x by issuing cwait (x); After blocking, it then waits in a queue of processes to enter again the monitor when the condition alters, and resume execution at the point in its program following the cwait (x) call.
- If the execution of process going on in the monitor and during execution, it notices alter in condition variable x, it issues csignal (x), which signals the related condition queue that the condition has altered.
- It is possible for producer to put in characters to the buffer only by way of using procedure append within the monitor; the producer does not have straight access to buffer.
- This procedure initially makes sure the condition *nor* M to conclude if there is space existing in the buffer if space i, »« available then the process executing the monitor is blocked on that condition.

Syllabus Topic : Atomic Transacti 0fl3

3 J2_Atomlc Transact lone

Q. What is atomic transactions? Explain,

- Critica' ... means These
- Concurrent execution of two critical sections gives the result equivalent to their ... unknown order. For data base syste'T ended sLZ Xu XXT' " SOI » e

... important along with its storage and retrieval. ... consistency.

3.11.1 System Model of Atomic Transac

Wh.le processing the transaction its ato... be preserved although there » poss TX ... there in ... Transaction is a program which ... and updates data within file on disk.

- So transaction is treated as sequence of read and ... operations that can be terminated by comm... operation. Commit operation indicates 1... is completed and terminated its execution succes...
- Whereas abort operation indicates that ransac... terminated unsuccessfully due to some er...
- As aborted transaction modifies the d ... termination, the state of the date must be stored ... state what was before transaction started.
- This roll back of transaction must be en... system. To ensure the atomicity, it is ncce ... devices that store the data. Transactions acc... from these devices.

Volatile storage

- Examples are main memory or cache memory system crash, information on this storage can... survived.

Nonvolatile storage

- Examples are disk and magnetic tapes. Ifq crashes then information on this storage sunhn storage is slower than volatile storage.

Stable storage

- Rephcate the information on other disk to surw

3.11.2 Log-Based Recovery

- The atomicity is ensured by recorrW modifications to the data by transaction. Thisraa is done on stable storage. Write-ahead logging' method to achieve it.
- In this method, a log is maintained by system 011* storage and each log record described single of ®* by transaction write. Following are the fid* * record.

o **Transaction name :** This name of «•* ... ®* doo ' write operation is unique. 1

o ... is wri,!*"" "*"* : This name of data item 1 *S Wn ,en 's also unique.

o **°M value** ■ The ... before

- o New value : Value of data operation after write operation is carried out.

– Other log records are also there to record the significant events during processing of the transactions. These are for example, start of transaction, commit or abort of transactions. Using these logs the data is easily recovered in case of failures. Recovery algorithm uses following two procedures:

- o undo(Ti) : The data items updated by transaction Ti is restored to old value by undo(Ti) procedure.
- o rtdofTi) : This procedure set the value of data items to new value. These data items are updated by transaction Ti.

3.11.3 Checkpoints

Searching of log is carried out in case of failure. It is necessary to search the log to know the transactions to be redone and undone after failure. This searching is time consuming and modifications on updated data will cause recovery to get longer. Checkpointing is the solution over this problem. Along with maintaining the write-ahead log, system time to time takes checkpoints which involve the following sequence of actions to take place :

- Write all log records at present residing in main memory onto stable storage.
- Write all modified data residing in volatile storage to the stable storage.
- Write a log record <checkpoint> onto stable storage.

As <checkpoint> record is present in the log, it permits the system to make its recovery procedure more efficient.

3.11.4 Concurrent Atomic Transactions

Each transaction is atomic. There can be many transactions executing concurrently. This concurrent execution is equivalent to serial executions of the transactions in any of the order. This is called as serializability property. This property can be maintained by just executing each transaction within a critical section.

3.11.4(A) Serializability

Q. Write note on Serializability

– Let X and Y are two data items that can be read and written by two transactions TO and TI. Both transactions executes atomically in - the . order followed by TI. This sequence of execution is called as schedule. In serial schedule each transaction is executed atomically. A serial schedule contains a sequence of instructions from different transactions in which the instructions of a particular transaction appear together.

– If two transactions overlap their execution then the resultant schedule is no longer serial. A nonserial schedule does not essentially mean an incorrect execution. The Fig, 3.11.1 shows the serial schedule in which TO is followed by TL The Fig. 3.1f.2 shows the concurrent serializable schedule.

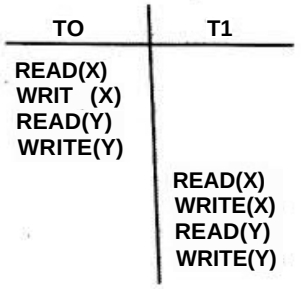


Fig. 3.11.1

- Fig. 3.11.1 shows that serial schedule in which TO is followed by T k

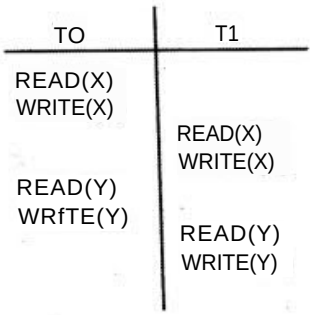


Fig. 3.11.2: A concurrent serializable schedule

Let OP_0 and $OP_($ are the two operations from transactions TO and TI respectively. If OP_0 and OP> accesses the same data item and at least one of the operation is write then OP_0 and OP_t conflicts.

3.11.4(B) Locking Protocol

Q. Explain two-phase locking protocol.

— A lock is associated with each data item to ensure the serializability. In this case it is necessary by each transaction to follow the locking protocol that defines how locks are acquired and released. Following two models can be used.

- o **Shared.** If a transaction Ti has acquired a shared mode lock on data item A, then Ti can read this item but cannot write A.

- **Exclusive.** If a transaction T_j has acquired an exclusive-mode lock on data item A, then T_i can both read and write A.

- Two-phase locking protocol also ensures the serializability. In this protocol each transaction should issue lock and unlock requests in two phases:
  - **Growing phase,** in this phase, a transaction may acquire locks but may not release any locks.
  - **Shrinking phase,** in this phase a transaction may release locks but may not acquire any new locks.

- At the start a transaction is in the growing phase. The transaction acquires locks as required. After releasing a lock, it enters the shrinking phase, and no more lock requests can be issued.

- The conflict serializability is ensured by two-phase locking protocol but does not ensure freedom from deadlock. Beside this, it may happen that, for group of transactions, there are conflict-serializable schedules that cannot be obtained by using the two-phase locking protocol. To get better performance over two-phase locking, we must either need added information about the transactions or to impose some structure or ordering on the set of data.

3.11.4(C) Timestamp-Based Protocols

- In above two-phase locking protocol, the order followed by pairs of conflicting transactions is decided at execution time. Other method selects the order in advance to determine serializability order.

- For this time-stamp ordering scheme is used. System assigns a unique fined timestamp to each transaction before it starts the execution. If TO transaction has $T_{S(TO)}$ and later T1 transaction enter in system having timestamp TS(T1). Following two methods are used to implement this approach.
  - The value of the system clock can be used as the timestamp. In this method, the timestamp of transaction is equal to the value of the clock when the transaction arrives in the system.
  - A logical counter is used as timestamp. In this method, the timestamp of transaction is equal to the value of the counter when the transaction arrives in the system. The counter j, increments after a new timestamp is assigned.

- Transactions occur on separate System Processor, not sharing any clock, this method will not work.

- Serializability order is determined by timestamp system protocol TM_Z » BTS<TO,<TS<T ,)to equivalent schedule to serial schedule b w' appears before T1.

Syllabus Topic : Deadlock s

3.12 Deadlocks

■> (June 15, Nov. 15, May 16, Dec.

Q. What is deadlock ?

MU- June 2015, Nov. 2015, May 201 Dec. 2016, 2 Mark

Deadlock occurs in system when process waiting resources never gets resources as these are held by processes. So in this scenario, several processes change state from waiting to other state.

Syllabus Topic : System Model

3.12.1 System Model of Deadlocks

We know that processes needs different resources! order to complete the execution. So in a multiprograra environment, many processes may compete for a multy number of resources. In a system, resources are finite.S with finite number of resources, it is not possible to Hi the resource request of all processes.

- When a process requests a resource and if the resori IS „Ot available at that time, the process enters a wai state. In a multiprogramming environment it may happen with many processes. There is chance** Waiting Processes will remain in same state and will never again change state.

- It is because the resources have requested are held by other waiting processes. When such type of situation occurs then it is called as deadlock.

- If there are 5 same type of resources avail * system then we say that there 5 instances of resource type C

- Suppose the process request the instance of resource. If any instance of the resource type is allocated to it, then request will be fulfilled. If it is not possible to fulfill the request, then the instances are not identical, and the resource type classes have not been defined appropriately.
- As a rule, first process must request a resource and then use it. After the use of this resource, it should release it. A process may request any number of resources which are necessary to complete its execution.

Following are the sequence wfien a process may utilize a resource :

1, **Request :** If it is not possible to fulfill the request instantly, then the requesting process must wait until it can get the resource.

2, Use : The process use the resource to accomplish the task.

3, **Release :** The process free the resource after use

Syllabus Topic : Deadlock Characterization

3.13 Deadlock Characterization

In deadlock, processes never finish execution as resources are held up by other processes. Due to this unavailability new processes are prevented from starting the execution. Following are the features that characterize the deadlock.

3.13.1 Conditions

In deadlock, processes never finish execution as resources are held up by other processes. Due to this unavailability new processes are prevented from starting the execution.

Necessary Conditions

(June 15, Nov. 15, May 16, Dec. 16)

a. Eti.-=- necessary and sufficient condition deadlock to occur.

yMtpTjune 2015. Nov. 2015. Fay 2016 Dec■201g. gJVjjgkS,

If the following four conditions hold simultaneously in a system then a deadlock situation can occur in the system. These are shown in Fig. C3.3.

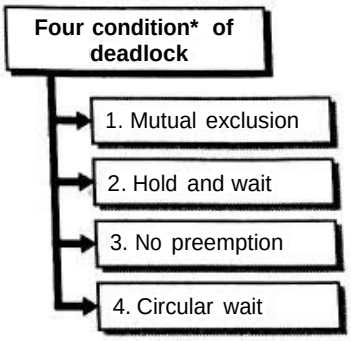


Fig. C33 : Necessary conditions of deadlock

1. Mutual exclusion

This condition ensure that, resource should be used by single process at a time and it remains with using process in non-sharable mode. If same resource is needed by another process then the requesting process must be postponed to use this resource. Requesting process will get the resource after released by holding process,

2. Hold and wait

The process holds minimum one resource and waits to obtain remaining needed resources that are at present being held by other processes.

3, No preemption

Preemption of the resource will not be done; Process holding the resource, releases it on its own after the completion of the its chosen task,

4. Circular wait

In this condition, a collection of waiting processes say|P0, Pl, ..., Pn) exist such that P0 is waiting for a resource that is held by Pl, Pl is waiting for a resource that is held by P2, ... Pn-1 is waiting for a resource that is held by Pn, and Pn is waiting for a resource that is held by P0.

3.13.2 Resource Allocation Graphs

-> **(Dec. 16)**

q write note on : Resource Allocation Graph.

MU - Dec. 2016. 5 Marks

- A directed graph called a system resource-allocation graph can explain the deadlock precisely.
- In this graph the set of vertices V is partitioned into two different types of nodes. These are,

...esses in the system, say P = { P1, P2, ... Pn}

The set consisting of all the active pre—

The set consisung ... in the system,

say R = 1 R ... wi tb d-rechon fro* ... p .hen it mean

m this graph If there is ... typ e R ... n(jW

process P to instance of resource ... that P has made request for an instance of R and ... waiting for that resource. If there is edge with direction from resource type R to process P then it means that an ... R has been allocated to instance of resource type ... process P

- So, edge fhrmf rocess P to resource ... and edge from resource R to process Vis'X ... edge. Circle nodes in graph represent processes and square represents resources.
- If resource allocation graph contains cycle then ... may exist in the system. If it does not contain cycle, then processes are nor deadlocked.

If system contains single instance of all the res *"" ■ 2s then all processes involved rn cycle is deadlocked. So in this case, a cycle in rewurce allocation graph is both necessary and sufficient condition for existence of deadlock. Dot inside square represents single instance of that resource.

- Cycle in following graph shows the deadlock situation when resource instances are single. PI has requested the resource held by P2 and P2 request the resource held by PI. P3 has requested resource R3.

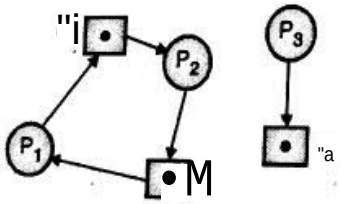


Fig. 113.1 : Resource Allocation Graph

- If mor. than one instances of each resource type is present in resource allocation graph then a cycle indicates that a deadlock has occurred. In this case a cycle « *a necessary* condition for the existence of d" ... SUffieieW f o r ~e of
- The reason behind cycle not being the snffi ■ condition is that, the precess can release the held resource although cycle exists. This released ... then can be obtained h eased resource ... requesting process.

Conserve y cycle can be broken. He, ce, if c ■ ... present in resource allocation graph the... free from deadlock state. In case if cycle K ... system may or may not be in a deadlocked state.

Syllabus Topic : Deadlock Prevention

3.14 peadfockf!?*® tion

→ (June 15, Nov. ...

Q. Explain deadlock prevention. MU - June 20ic; 2 Mark...

Q. Whal i8 difference between deadlock avoidanc... and deadlock prevention? MU - Nov. 2015. 10 Mark...

THfecribed above, for a deadlock to occ$_{Ur}$, each ... , te four necessary-conditions: mutual exclusion. hold ... X 2 preemption and circular wait should ... simultaneously in the system. By ensurtng that at least ... one these conditions cannot hold, we can pn ent ... occurrence of a deadlock. So deadlock prevention can ... achieved by denying the holding of at least one of, ... fol lowing four conditions.

1. Mutual exclusion	2. Hold and wait
3. No preemption	4. Circular wait

The approach is explained as bel o w.

1. Mutual exclusion

- Non-sharable resources cannot be shared by mas processes simultaneously. If resources are iw sharable then mutual -exclusion condition ma hold for them.
- For example, It is not possible to use printer h many processes at the same time. Printer is no... sharable but read only file is sharable resource ... do not require mutually exclusive access.
- Therefore sharable resource such as read only ... cannot be involved in a deadlock. We cannot ... prevent deadlock by denying mutual exclusion as ... some resources are inherently non-sharable such as printer.

2. Holil and wait

- Here we have to see that hold and wait will never occur. The * should be an assurance I* ... — P 2 hould requests a resource only if it does ...

not hold any other resource. Following two protocols can be used.

- Allocate all the resources needed by process before it starts the execution. This protocol will keep the allocated resources unused. The reason is that, even though some of the resources are needed by process in the last stage of execution, it will hold it from start of execution.
- Second protocol allows the process to request the resources only when it has released the previously allocated resources. Again the disadvantage of first protocol appears here. Suppose the important resource for which many processes compete is needed only at the start and end of the execution by this process. If initially process release resource and at the end request for the same, then it may not possible to get the same resource immediately. So we must request all resources at the beginning.
- Apart from the limitation of the above two protocols, other limitation can be starvation. In this, processes will keep significant and popular resources with them forcing to wait other processes indefinitely which require these resources.

3, No Preemption

- We have to ensure that no preemption condition does not hold. Following two protocols can be used.
- If a process at present keep some resources with it and requests another resource that cannot be immediately allocated to it, then all resources at present being held should be preempted.
- The list of the resources for which process is waiting is maintained. The preempted resources from the process are included in this list. The process now restart the execution provided that it can get back its old preempted resources, in addition to the new ones that it is requesting.
- If the needed resources by the process are available, then allocate to the process requesting for it. On the other hand if they are not available, then check whether it is allocated to the process which is further waiting for additional resources. If it is the case then withdraw the resources from waiting process and allot them to requesting process.
- The process which requests the resources should wait in case If resources are neither available nor held by waiting process. In this situation, if other process make a request for the resource held by waiting process, then it should be preempted from waiting process.

4. Circular waft

- To prevent the occurring of circular-wait condition inflict a total ordering of all resource types. In this case each process should request the resources in an increasing order of assigned numbers to the resources.
- Consider the collection of resource types R = {RI, R2, ..., Rn). To this each resource type, a unique integer number is assigned. As resources are now recognized by their numbers, it is easy to compare them. It is also easy to know if one resource precedes the other in our ordering. Assignment of numbers to resources is a one-to-one function p ; r _+ N, where N is the set of natural numbers. In order to prevent the deadlock, following two protocols can be used.
  - o Each process requests resources in an increasing order of assigned numbers to the resources.
  - o Whenever process request instance of resource type Rj it should release any resource Ri such that F(Ri) > = F(Rj) .

If these two protocols used, circular wait condition cannot hold. If process P, waits for resource Ri which is held by process Pi+I we get F(Ri) < F(Ri+l) for all i. It implies that F(R0) < F(R1) <F(R2) < ..... F(Rn) < F(RO). By transitive relation, we get contradictory statement F(R0) < F(R0). circular wait does not hold

" ""s nTbusTopicTpeadlockAvoidance ~

3d5 Deadlock Avoidance

➔ **(June 15, Nov. 15)**

Q. Explain deadlock avoidance.
MU • June 2015. 2 Marks

Q. What is difference between deadlock avoidance and deadlock prevention?
MU - Nov. 2015. 10 Marks

Deadlock avoidance requires that the system has some UU<MW«t>'*available up front. Initially every process d which AvUres the nw'htwm number of resources nee to

- An ltZn«»>« method for avoiding deadlocks is are require additional information about ess wjl| to be requested. If the sequence in which pr« n request the resource and will release it toow it Jk.on.Hy, then we can decide for each reque. whether or not the process should wait.
- To avoid the circular wait condition completely, a deadlock-avoidance algorithm inspects the resour « allocation slate in dynamic way. Following are factors which define and describe resource allocation state.
  - o The number of available resources in the system.
  - o The number of allocated resources in the system.
  - o The maximum demands of the processes.
- Dijkstra's Banker's algorithm is used for deadlock avoidance. The algorithm requires prior knowledge of number of resources each process needs.
- After that, the OS act like a sincere small-town banker. OS will allocate the resources to processes only it has sufficient number of resources available to fulfill possible demand. This is considered as a safe state.

Safe State

- If the system is able to allocate resources to each process up to its maximum need in some order and still avoid a deadlock then the system is in safe state.
- If the safe sequence of all processes present in system the system is considered to be in safe state: Sequence < P1, P2, ... Pn > is considered as a safe
- sequence for the present allocation state if, for each process Px, the resources which Px can still request can be fulfilled by,
  - ° The at present available resources in the system plus
  - o The resources held by all of the processes Py's, where y < x.
- If process Pi cannot get all the needed resources as they are not available, it should wait until Pj Complete the execution and frees the resources
- If the system is a e state, there can no deadlock. If the systein P°ssib ®y of dead]ock unsafe state, there is the

cr Example

Consider a system with 10 disk drjy es and 3 (P0, P1, and P2) :

Process	Maximum needs	Current
PO	8	4
P I	4	2
P2	7	2

- \~il71O disk drives are available in the L 4PJ. P0. P2> is safe sequence M ''‘“fres 8 disk drives, process P1 requi 'M P2 may utre up to 7 disk drives. A4 holding disk drives ieTs PO « holding 2 disk drives and prtrcess PO J drives. (Thus in the system 2 disk dnves are fr
- At time to, the system is in safe state. 1 <pi PO, P2> satisfies safety condition,*! immediately be allocated its disk drives 1 them after completion. Now total 4 disk J available. 1
- PO gets all the disk drives. After completion them and now total 8 disk drives are avails ! system. Finally P2 can be allocated all the ne J drives and returns them after completion. (The J will have now all 10 disk drives aJ

Fig. 3.15.1 shows safe, unsafe and deadlock spaces.

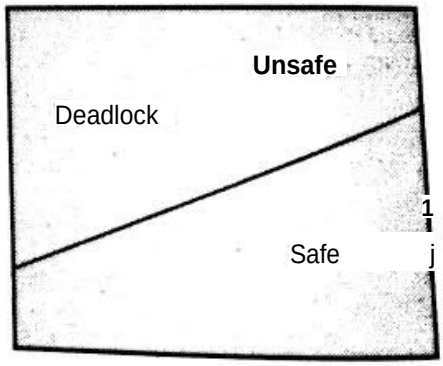


Fig 3.15.1 : Safe, unsafe, and deadlock states?

- **At** time t_h if any process requests the resource say disk drives in our example and if allocation other process could not be allocated resource type due in unsaf e state. t0 unava *tability , then system

3.1

5-1 Deadlock Avoidance Algorithm#

(Mayifc*

Following algorithms are used in the avoidance of deadlock.

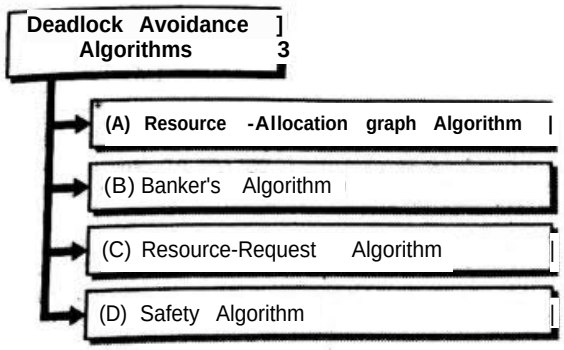


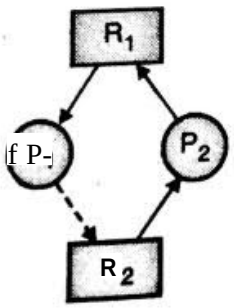
Fig. C3.4 : Deadlock avoidance algorithm

4 3.15.1(A) Resource-Allocation Graph Algorithm

- If system has resource allocation system with only one instance of each resource type, then only this algorithm is applicable.
- Future request edge in resource allocation graph is represented by claim edge (dotted edge).
- The claim edge is transformed to a request edge when a process requests a resource.
- The assignment edge is changed to a claim edge after a process free the resource.
- When requested resource is allocated to the process then request edge is converted to assignment edge. If assignment leads to cycle in resource allocation graph then system is in unsafe state. If not so then system is in safe state.
- After assignment of resource, if cycle detection algorithm detects cycle, then system is in unsafe state.

The unsafe state in resource allocation graph is shown in Fig. 3.15.2.

- Cycle detection : Ofn^2)



Fig, 3.15.2 ; Rewuree-allocation graph for deadlock avoidance

3.15.1(B) Banker's Algorithm

(Dec. 16)

Q. Explain the banker's algorithm In detail.
MU - Dec. 2016, 10 Marks

- It is applicable to the resource allocation system with multiple instances of each resource type.
- It is less efficient than resource-allocation graph algorithm.
- Newly entered process should declare maximum number of instances of each resource type which it may require.
- The request should not be more than total number of resources in the system.
- System checks if allocation of requested resources will leave the system in safe stale. If it will the requested resources are allocated.
- If system determines that resources cannot be allocated as it will go in unsafe state, the requesting process should wait until other process free the resources.

Data structures used In Banker's algorithm

(June 15)

Q. Explain data structures used in bankaris algorithm.
MU ■June 2015, 5 Marks

Following Data structures are used in Banker's algorithm

- A[m] : Array A of size m shows the number of available resources.
- **M[n][m]** : Two dimensional array M shows maximum requirement of the resources by each process. **M[i][j]** = k indicates process Pi can request at the most k instances of resource type Rj.
- **C[n][m]** : Two dimensional array C shows cunent allocation status of resources to each process. **C[i][j] = k indicates process Pi allocated k instances of** resource type Rj.
- **N[n][m]** : Two dimensional array N shows the remaining possible need of each process. i.e., N[iJGl = M[i][j] * C[i][j]). N[i][j] = k indicates process Pi may need additional k instances of resource type Rj to accomplish the execution.

3.15.1(C) Resource-Request Algorithm

- This algorithm decides whether requests of the resources can be safely granted so as to system will remain in safe state,

- Let R[n][ml be the two dimensional array, describe the current outstanding requests of all processes. R[i][j] ~ k indicates process Pi want k instances of resource type Rj. to accomplish the execution.

1 ■ H R[i][j] <= N[i](j], to step 2; otherwise, raise a® error condition as process is requesting mor resources than it needs.

2L If R[i](j] > A(jL then the process should wait as needed resources are not available.

J, Otherwise, go to step 4

4, Allocate the requested resources t< Pi. The st®1® is altered as

A|jl = Aoi-mn

C[il[j] = C[ilO) + R[i][j]

N(ilU) - N[i]lj] - R 1'1W

- Once the resources are allocated, check to - if the system state is safe. If unsafe, the — must wait L the old resource-allocated states restored

➜ 3.15.1(D) Safety Algorithm

- This algorithm find out whether system is ■" state or not.

1. U, w be an integer array of length m, initialized to array A (Available number of resource ;). Let F be a boolean array of length n. initialled to false (determines if process finished or not).

2. Find i such that both:
   F[il = false
   N[i]<=W
   If DOsuch i exists, go to step 4

3. W = W + C(i];
   Ffi] = true;
   Go to step *2*

4. If F[i) = true for all i, then the system is in a safe state, otherwise unsafe.

Kun-time complexity : 0(m x n²)

- To demonstrate the working of banker's algorithm, consider the following snapshot of the system at time to. Processes are numbered P0 to P4 (total 5 processes).
- Resources are P, Q. and R (total 3 resources). Resource type P has 12 instances, resource type Q has 9 instances, and resource type R has 6 instances.

Process	CfAlK***	M(Maximum rec uirenwnt)	N Remaining need)
	P, Q, R	P, Q, R	P, Q, R
P0	3. 0. 1	5, 2,2	2, 2, 0
P1	0, 2, 1	3, 2, £	3. 0.1
P2	2. 0. <1	3, 2, 2	2 a-2
P3	2, 1, 0	7, 3, 1	5, 2, 1
P4	4, 0, 2	5, 0. 3	1, 0, 1

„ currently with sequence <pj , P1, P

The 9yS * state- The sequence <P2, P1 , P0, P

P3> is in safe safety criteria as shown.

satisfy the

s P 2 :

1. pO ,,, (| 2. 2) <= ('• 6, 2) is true
   N <= \ (i, 6. 2) - <1. 2. 2) = (0, M
   A = A ~ C (0. 4. 0) + <3 - 2. 2) =13,5,
   available in the system after P2 finishes.

2. **process P1 :**
   N <= A (3, 0, D<=(3. 6, 2) is tree I
   A = A - N (3, 6> 2) _(3, 0. D = (0,6,1)
   A = A + C '(0,6, 1) + (3. 2, 2) = (3,8,
   available in the system after P1 finishes.

3. **process P0 J**
   n <= A (2, 2,0) <=(3,8.3) fctrae
   A = A - N (3, 8, 3) -(2, 2,0) = (1,6,3)
   A = A + C (1,6,3) + (5,2, 1) = (6,8,4)a*
   in the system after POfinishes.

4. **Process P4 :**
   N <= A (1,0, 1) <=(6,8, 4) is true
   A = A - N (6, 8, 4) - (1, 0, 1) = (5, 8,3)
   A = A + C (5, 8, 3) + (5, 0, 3) = (10.
   available in the system after PO finishes.

5. **Process P3 :**
   N <= A (5, 2, 1) <= (10, 8, 6) is true
   A = A - N (10,8, 6)- (5,2, 1) = (5,6.5)
   A = A + C (5, 6, 5) + (7, 3, 1) = (12A
   available in the system after P0 finishes.

So the required resources can be allocated» processes in the sequence <P2, p(, PO, P4, P3> *1 j

SHSE Suppose at time H.P? renuest one

jnstsnce of resource type P and two instances of resource type Q, so Request R2 - (1, 2, 0) and it is not greater than available resources A.

So step 4 of the resource request algorithm indicates that request can be allocated. After allocation of these resources, now safety algorithm is executed to determine whether system remains in safe state. If any such sequence exists, then system will be in safe state.

3 16 Solved Problems

Example 3.1 **8.1 fa [*11** Dec. 2014 **It-I'll** Marks

rrtnsidor the following snapshot of a s'/stem :

	Allocation	Max	Available
	A B C D	A B C D	A B C D
P0	0 0 1 2	0 0 1 2	1 5 2 0
P1	1 0 0 0	1 7 5 0	
P2	1 3 5 4	2 3 5 6	
P3	0 6 3 2	0 6 5 2	
P4	0 0 1 4	0 6 5 6	

With reference to banker's algorithm

(a) Find need matrix

(b) Is the system in a safe state?

(c) If a request from process P1 arrives for (0, 4, 2, 0), can the request be granted immediately ?

Solution :

(a) Need matrix is,

	Need			
	A	B	C	D
P0	0	0	0	0
P1	0	7	5	0
P2	1	0	0	2
P3	0	0	2	0
P4	0	6	4	2

(b) System is in safe state because many safe sequences exist such as (TO; P2; P1; P3; P4)- As avadab'e tt, < 1.5. 2. 0). either process P0 or « can complete the execution. Once process P3 finishes, tt release ° resources which permit all other existing processe. execute.

(c) Yes, request can be granted immediately* After granting this, Avaflabte will become (I_t 1, 0, 0).One of the safe sequence is (P0, P2, P3, *P1,* and P4). Allocation for P I becomes (I_t 4, 2, 0) and Need for P I becomes (0, 3, 3, 0),

Example 3.16,2

Consider the following snapshot of the system.

Process	Allocation A B C	Maximum Requirement A B C
P0	0 0 1	1 1 1
P1	2 0 0	3 2 3
P2	1 3 2	4 3 1
P3	1 0 1	0 0 1
P4	0 0 1	3 2 1

Let ABC be the resources with instances of A is 7, B is 3 and C has 6 instances.

(I) What is the content of matrix need ?

(ii) Is the system in a safe state? Prove it.

Solution :

(i) Need = Max - Allocation

A	B	C
1	1	0
1	2	3
3	0	1
- 1	0	0
3	2	0

(ii) A = 7, B = 3 and C = 6 are total instances of resources. By adding the Columns of Allocation matrix

We get, A = 4, B = 3 and C = 5,

Now Available number of resources = Total - Allocation

- Resource A available = 7 - 4 = 3

  Resource B available = 3 - 3 = 0

  Resource C available = 6 - 5 - 1

By using above available number of resources and applying bankers algorithm, sequence

<P2, P3, P4t PO, P1> is safe sequence.

Hence, system is in safe state.

Example 3.16.3

Consider the system with 5 processes, pO to p4 and resource type A, B, C. An rthe following questions.

Process	Allocation A B C	Request A 0 C	Available A B C
P0	0 1 0	0 0 0	0 0 0
P1	2 0 0	2 0 2	
P2	3 0 3	0 0 0	
P3	2 1 1	1 0 0	
P4	0 0 2	0 0 2	

Let ABCbe the resources with instar** 5 of A is 7, B is 3 and C has 6 instances.

(i) Is the system in a safe state?

(ii) If at time t_1, P2 makes additional request for an instance of type C, Check whether system is in deadlock or Not at time*1

Solution :

(i) Request rpQi<= Available (0 0 0 <— qqo)

Hence request of P0 is satisfied. P0 free the resources (0 / «>..« Available becomes (0 1 0 1

.yto P0, Let the P2 request for resources.

Request (P0)< "Available (0 0 0 < = 0 1 0)

Hence request of P2 is satisfied. P2 free the resources (3 0 3) and Available becomes (3 1 3).

Available - Available + Allocation {P2|.

In this way we can find that P3, P1 and finally P4 gets all their requested resources and safe sequence exist < P0, P2, P3, P I, P4>. Therefore system is in safe state.

(ii) If at time t_t P2 makes additional request for an instance of type C then Request [F0]< = Available (0 0 1 < = 0 1 0) is false and resource C cannot be granted to P2. System will be in deadlock state.

Example 3,164

It is proposed to use bankers algorithm for handling deadlock Totai number of resources available for allocation *fe 7,* 7 and 10 respectively. The vocation state is shown as below. current resource allocation

Process	Allocation Ri R2 R3	Max R1 R2 R3	Avail R1 R2
P1	2 2 3	3 6 b	7 7
P2	2 0 3	4 3 3	
P3	1 2 4	3 4 4	

(i) 19 the CU following request be flT * nter in stat®7

(ill wou *

* P 1 requ®5 , (1 , 1, 0;

Sotut* on :

n c e c P h ? 2 ,**3 *

0) “: d : , x — is 4, 10). It is adding AH° caUon column ,

RequesUPU = MaxlPU ' """"Wil =

Now available in system are (7, 7. 10)

The request of P2 satisfied as av ailable in syst

(7, 7, *10)* and then of P3 is satisfied, the refore is in safe state.

(il) Request of P1 is satisfied as (1, 1, 0) <= (7,7,

Example 3 -1 6 5

Consider follow* ng snapshot of the system

Using Banker's algorithm answer the following questio

(i) How many resources of type A B C D are there?

(ii) What are the content of need matrix ?

pil) Find if system is in safe state? If it is, find sequence.

Process	Max A B C D	Allocation A B C D	Available A B C
PO	6 0 1 2	4 0 0 1	3 2 1
P1	1 7 5 0	1 1 0 0	
P2	2 3 5 6	1 2 5 4	
P3	1 6 5 3	0 6 3 3	
P4	1 6 5 6	0 2 1 2	

Solution :

«) A-9; B-13; C-10 ; D-ll

(H) Need[i, j] = Max[ij] - Allocation jl 50 c

Need matrix is

	A	B	C	D
PO	2	0	1	1
P1-	0	6	5	0
P2	I	1	0	2
P3	1	0	2	0
P4	1	4	4	4

(jii) The system is in a safe state as the processes can be finished in the sequence PO, P2, P4, PI and P3,

Example 3.16.6 MU - June 2015. Nov. 2015. 10 Marks

Consider the following snapshot of th© system

	Allocation	Max	Available
	A B C	A B C	A B C
P0	0 1 0	7 5 3	3 3 2
Pt	*2 0 0	3 2 2	
P2	3 0 CM	tt o cv	
P3	2 1 1	2 2 2	
P4	0 0 2	4 3 3	

Answer following questions using banker's algorithm

(a) What is the content of need matrix?

(b) Is the system in a safe state?

(c) If a request from process P1 arrives for (1, 0, 2), can the request be granted immediately ?

Solution :

(a) Need = Max- Allocation

	A	B	C
PO	7	4	3
P1	1	2	2
P2	6	0	0
P3	0	1	1
P4	4	3	1

(b) System is in safe state because safe sequence exist and it is (P1; P3; P4; P2;PO)-

(c) Yes, request can be granted immediately The reason is Request <= Available that is (1, 0,) <- (3, 3, 2)

Example 3.16.7 MU -Nov. 2015.10 Maiks

Consider the snap of the system.

Process	Allocation A B C D	Max a b c d	Available A B C D
P0	0 2 1 2	0 3 2 2	2 5 3 2
P1	1 1 0 2	2 7 5 2	
P2	2 2 5 4	2 3 7 6	
P3	0 3 1 2	1 6 4 2	
P4	2 4 1 4	J _ 3 6 5 8	

Answer the following questions using bankafs algorithm.

(I) What is the content of Matrix *need?*

(ii) Is the system in safe state?

(iii) If a request from process PI arrives for (1.3,2,1) can the request be granted immediately ?

Solution :

(I) Needfi, j] = Max[i jl - Allocation!. j]

So content of Need matrix is

	A	B	C	D
P0	0	I	1	0
P1	I	6	5	o
P2	0	I	2	2
P3	1	3	3	0
P4	1	2	4	4

(ii) Yes, System is safe and safe sequence (PO, P2, P1, P4, P3) exist.

(iii) As (3, 3, 4, 8) remains available in system, P1 can be granted (1,3,2,1) immediately

Sylhbus°Top7c : Deadlock Detection

3.17 Deadlock Detection

➔ **(June 15)**

Q. Explain deadlock detection.
MU - June 2015. 2 Marks

If the system does not make use of deadlock prevention or deadlock detection algorithm, then system will observe the deadlock situation. In this situation, it is necessary that, system should have .

Algorithm which examines the state of the system to decide whether a deadlock has occurred.

- Algorithm to recover from deadlock if it is occurred.
- Both the algorithms requires the overhead including maintaining necessary information and executing the detection algorithm and potential losses in recovering from deadlock.

Single instance of each resource type

- This uses a wait-for graph. Wait for graph which is resource allocation graph is used by algorithm if all resources have only single instance.
- The wait for graph is obtained from the resource-allocation graph by eliminating the nodes of type resource and collapsing the suitable edges. This is shown in resource allocation graph Fig. 3.17.1 which shows that, process P_1 is requesting resource r_2 which is held by process P2.

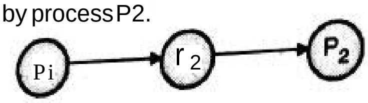


Fig. 3.17.1 : Resource allocation graph

- Above situation in resource allocation graph is represented as shown in wait-for graph Fig. 3.17.2.

Fig. 3.17.2 : Wait for graph of above resource allocation graph

- In this way system maintains the wait-for graph and if it contains the cycle, the deadlock exists in the system. By invoking cycle detection algorithm on wait-for graph, the deadlock can be detected. If n is number of vertices in wait-for graph, n^2 operations are required to detect the cycle.

Many instances of each resource type

The wait-for graph discussed above is not applicable to multiple instances of a resource type. In this case algorithm similar to Banker's, which investigates every possible allocation sequence for processes which remain to be completed. The structures used are as follows.

- A[m] ; Array A of length m shows the number of available resources.
- C[n][m] : Two dimensional array C shows current allocation status of resources to each process.
- R[n][m] : the two dimensional array, denotes outstanding request of all processes. Pi want k instances of resource type Rj for completion of execution.

Algorithm

Q. Write note on Deadlock Detection algorithm.

1. Let W be an integer array of length m initialized to A (Available number of resources). F is array of length n, utilized to know whether process finished or not.
   For all i, if C[i] ≠ 0 then F[i] = false; else F[i] = true.
2. Find an i such that both
   F[i] = false // Pi is currently not involved in deadlock
   R[i] <= W
   If no such i exists, go to step 4
3. // reclaim the resources of process Pi
   W = W + C[i]; F[i] = true; Go to step 2
4. If F[i] = false for some i,
   then the system is in a deadlocked state. Moreover if F[i] = false, then process Pi is deadlocked.

Run-time complexity : O(m × n²) :

Invocation of detection algorithm depends on:

- Frequency of occurring the deadlock. If it is frequently then algorithm will be called frequently.
- The number of processes that will be influenced by deadlock when it occurs.
- If the request of any process cannot be fulfilled, there are chances of deadlocks to occur.
- The detection algorithm is invoked in following cases.
- **Extreme :** Invoke the algorithm every time a request is denied.
- **Alternative :** Invoke the algorithm at less frequent intervals :
  Once per hour
  Whenever CPU utilization < 40%
- **Disadvantage :** Cannot determine exactly which process 'caused' the deadlock.

Syllabus Topic: Recovery from Deadl ~

3.18 Deadlock Recovery

Q. Explain deadlock recovery in detail.

- It needs to recover from deadlock when it is detected. Several options exist after detection algorithm detects that a deadlock exists in the system. One possibility is to inform operator and let them decide how to deal with ft manually.
- The second option is to allow the system to recover from the deadlock on its own (automatically). There are two solutions exist to break a deadlock. One solution is just to abort one or more processes so that circular wait will be broken. The second option includes preemption of some of the resources from one or more of the processes which are involved in deadlock.

3,18J Process Termination (Kill a process)

When deadlock occurs, the operating system decides to kill one or more processes. This will reclaim the resources acquired by that killed processes. The deadlock detection algorithm detects whether deadlock exist or not after killing the process. Following two methods are used for killing the process :

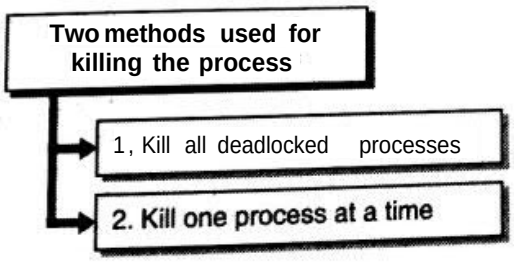


Fig. C3.5 : Two methods used for killing process

- F l . Kill all deadlocked processes

- This method is simple and effective to eliminate the deadlock and clearly will break the deadlock cycle, but at a huge cost.
- If all these killed processes have been performed die computation for longer period of time, then the part y computed result would be wasted. Recomputation will be done and is veiy costly.

➜ 2. Kill one process at a time

In this method, one process is killed at a time and check whether detection algorithm is in v o e t0 deadlock is still exist or not in die system

- Invocation of the detection algorithm after killing each process is considerable overhead. We must re-run algorithm after each kill.
- Killing a process is not easy. We should kill only those processes whose killing will incur minimum cost.
- Different parameters that need to be consider to ensure the minimum cost arc :
  - o Priority of the process.
  - o How much computation process has finished and how much more time does it need to finish the execution.
  - o The number of the resources process has used.
  - o The type of resources the process has used.
  - o The number of the resources the process will need to complete the execution.
  - o The number of processes needs to be killed.
  - o Whether the process is interactive or batch.

3.18.2 Resource Preemption

Incrementally preempt the resources from the process and reallocate resources to other processes until the circular wait is broken.

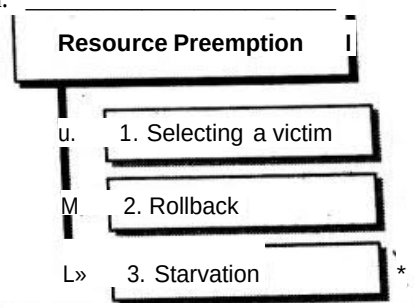


Fig. C3.6 : Resource preemption

➜ 1. Selecting a victim

Which process and which resources should be selected for preemption to minimize the cost? The process which has finished almost all computation and only few amount of computation is remained should not be selected as a victim.

➜ 2, Rollback

After preempting the resources from the particular process, it cannot continue the execution as needed resources are preempted. It is required to rollback the process to safe state and restarts the execution from that state. As safe state is difficult to define, total rollback will be the simple solution.

3. Synchronization

It may W ' jesouttes- times io p««P‘ the

3.19 Exam[pa]. an dRey!?*3 u - '

(Univers...)

Synchronization

svHabutToP ' ...nization in brief. (May 2016)

Q. SIES 5 *

Q. Explain the inter-process communication in brief. [...] 2016) (Refer section 3.2) (5 Marks)

Q. with example-

Q. „ . — — «“ (Refer section 3.3)

Syllabus <°P'ic : The Critical Section Problem

a Explain critical section probl em. (Dec. 2014) (Rg fer section 3.4) (5 Marks)

a. Explain critical section probintri With its different solutions. (Refer section 3.4)(10 Marks) (June 2015)

a. What is mutual exclusion? Marks) <»*-*“ 3.9(5 (June 2015, Nov. 2015)

Q. What is mutual exclusion ? Explain its significance. (Refer section 3.5) (5 Martis) (May 201 6)

Syllabus Topic : Peterson's Solution

Q. Explain Peterson solution to mutual exclusion. (Refer section 3.6)

Q. Give software approache for mutual exclusion. (Refer section 3.6)

Syllabus Topic : Synchronization Hardware

Q. Explain hardware solution to mutual exclusion. (Refer section 37)

Syllabus Topic : Semaphores

° WhatBsemaphorelflefersectionS

*■ Syllabus Topic ; Classic Prnh teffl . of 0 ems of Synchronization

Q. * Explain an algorithm for arorll ,„ ...er-consumer problem.(Refer section 3.9.1)(10 Marks)(Dec. 2016)

Q. Explain how Realtor ers problem can be solved with semaphores ? (Refer section 3.9.2) (10 Marks)

n«20l5)

Q. What do you mean by busy waiting? What ... with it? (Refer section 3.9.2) (5 Marks) (Nov. 2016

Explain «ad riterproWem - S

Q. (Refer Refer section 3.9.3)

Q. explain < 9 philosopher problem and ... it. (Rsfer section MW Marfa)

<DeCJ0 ,\ 1

Q. Explain 0Wn9 philosopher Prob... semaphores. (Refer sectron 3.9.4) *1

Toole : Monitors

Syllabus Topi

Q. YYtrat is monitor? How it is used to ach ... exclusion? Explain. (Refer section 3. l0) >

Syllabus Topic : Atomic Transaction,

Q, What is atomic transactions? Explain. (Refer section 3,11)

Q. Write note on Serializability. (Refer section 3.11.4(A))

a. Explain two-phase locking protocol. (Refer section 3.11.4(B))

Syllabus Topic : Deadlocks

a. What is deadlock ? (Refer section 3. 12) (2 Ma... (June 2015, Nov. 2015, May 2016,Dec.?

Syllabus Topic : Deadlock Characterization

Q. Explain necessary and sufficient condh deadlock to occur. (Refer section 3.13,1) (2 Ma... (June 2015, Nov. 2015, May 2016,Dec3

Q* Write note on : Resource Allocation Graph. (Refer section 3. 13.2) (5 Marks) (Dec.U

Syllabus Topic : Deadlock Prevention

Q. Explain deadlock prevention. (Refer section 3. 14) (2 Marks) (June*

Q. What is difference between deadlock avoids deadlock prevention? (Refer section 3.14) (WMarks) (W*

, ,,bU,T opic : O«k»k»«*!.«.

c1 Ew „„ a ~<ta.»ad.„c..

(Refer section ...

Q. What is difference between deadlock avoidance and deadlock prevention?
(Refer section 3.15) {IQ Marks) (Nov. 2015)

Q. Suggest techniques to avoid deadlock.
(Refer section 3.15.1) (1 Mark)
(May 2016, Dec. 2016)

Q. Explain the banker's algorithm in detail.
(Refer section 3. 15. 1(B)) (10 Marks) (Dec. 2016)

q. Explain data structures used in banker's algorithm.
(Refer section 3.15.1(B)) (5 Marks) (June 2015)

Example 3.16.1 (10 Marks) **(Dec. 2014)**

Example 3.16.6 (10 Marks) (June 2015, Nov. 2015)

Example 3.16.7 (10 Marks) (Nov. 2015)

☛ **Syllabus Topic : Deadlock Detection**

Q. Explain deadlock detection.
(Refer section 3. 17) (2 Marks) (June 2017)

Q. Write note on "Deadlock Detection Algorithms".
(Refer section 3. 17))

☛ **Syllabus Topic : Recovery from Deadlock**

a. Explain deadlock recovery in detail
(Refer section 3.18) ---------□qq'

CHAPTER 4

Memory Management

Syllabus Topics

Memory Management strategies : Background, Swapping, Memory Allocation, Structure of the Page Table, Segmentation; Virtual Memory Management: Demand Paging, Write, Page Replacement, Allocation of Frames, Thrashing, Memory-Mapped Files, Allocating Kernel Memory, Other Considerations.

Syllabus Topic : Memory Management Strategies

4.1 Memory Management Strategies

Background

Q. What are different requirements for memory management ? Explain.

- Memory is an important resource of the computer system that needs to be managed by the operating system.
- To execute the program, user needs to keep rhe program in main memory. The main memory is volatile.
- Therefore a user needs to store his program in some secondary storage which is non volatile.
- It is required to have process code, stack, heap (dynamically-allocated structures), and data (variables) to be in primary memory.
- Therefore memory must be allocated to every process. The management of main memory is required to support for multiprogramming.
- Many executables processes exist in main memory at any given time.
- Different processes in main memory have different address space.
- Memory manager is the module of the operating system responsible for managing memory.
- Programs after completion of execution move out of main memory or processes suspended for completion of I/O can be swapped out on secondary storage to free memory for other processes.
- New processes are required to be loaded in main memory.
- If available main memory is not sufficient to hold all processes, swapping between main memory and secondary memory is done.
- Memory managers move processes back and forth between memory and disk during execution.
- So it is required that operating system should have some strategy for the management of memory.

4.1.1 Monoprogramming

- In early computer systems and early microcomputa operating systems, mono-programming concept was used.
- Only one program was residing in main memory al given point of time.
- Operating system was residing in some portion of memory and other portion was fully devoted to li* single executing process.
- This system was simple to design but was supporting multiprogramming. In this appro relocation was needed as program could be dwa loaded at the same location.

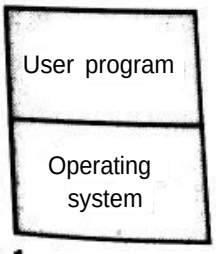


Fig. 4.1.1 : Monoprogramming

41,2 Multiprogramming

- Multiprogramming is required to support multiple processes simultaneously. Since multiple processes are resident in memory at the same time, it increases processor utilization if the processes are I/O bound, CPU does not remain idle.
- CPU utilization increases as it executes multiple processes on time sharing basis.
- Multiprogramming gives illusion of running multiple processes at once and provides users with interactive response to processes.

4.1.3 Dynamic Loading

- Dynamic loading ensure the better memory space utilization. Unless and until called, routine is not loaded in main memory in dynamic loading.
- All routines remain stored on disk in a relocatable load format.
- The main program is loaded into main memory and is executed.
- The advantage of dynamic loading is that, memory space is utilized only for the routines which are currently need to be executed and other unused routines reside on disk.
- If user designs the program which uses many library routines then operating system will itself support for dynamic loading. No special support require from operating system.

4.14 Overlays

- In the early days, as Large programs were too large to fit into their partitions, programmers created overlays to solve this problem.
- Rather than loading an entire program into memory at once, we can divide it into those modules that are always required and those that are required only at certain times during program execution.
- Overlays involve moving data and program segments in and out of memory which helps for reusing the area in main memory.
- Overlays were physical divisions of the program that werc established by the programmer. To complete the execution, it is required chat, entire program and data of a Process must be in physical memory.
- It is required to have process in main memory for execution and hence physical memory should accommodate the process.
- The process size can be larger Chan the amount of memory allocated to it.
- The overlays allows to have needed modules and data in main memory at any given time.
- The data and modules which are not required for execution at that time are swapped out of memory.
- Next time, the needed modules are loaded into space which is freed now by swapping the modules not required for execution.
- An overlay handler exists in another area of memory and is responsible for swapping overlay pages or overlay segments.
- Overlays are implemented by user; no special support needed from operating system, programming design of overlay structure is complex.
- Consider the example of two pass assembler. As we know that pass 2 is executed after pass 1, it is not necessary to keep both modules of pass 1 and pass 2 in memory simultaneously.
- During pass 1 symbol table is constructed and pass 2 generates machine-language code.
- Symbol table and common routines are required in both passes. Consider the memory requirement of pass 1 module, pass 2 module, symbol table and common routines.

Pass, 1 Module	100 K
Pass 2 Module	90 K
Symbol Table	30 K
Common Routines	30 K
Total	**250 K**

- Total 250 K memory is needed to run the assembler. If only 200 K memory is available then overlays can be defined as overlay A and overlay B.
- Overlay A contains pass 1 module, symbol table and common routines. Overlay B contains pass 2 module, symbol table and common routines.
- Overlay driver of 20 K is loaded in memory. Initially overlay A is loaded in memory which requires 160 K of memory.

- Once overlay A finishes execution, overlay driver loads overlay B in pace of overwriting overlay B. Overlay B requires IM K of memory.

4.1.5 Relocation

a — Write short note on Relocation.

Consider the pactions of memory as follows.

1. First partition- 100K (Operating system)
2. Secondpartition: 100K
3. Third partition: 20dK
4. Founh partition: 300K

- Consider veiy first instn file created by absolute address 100 within the binary
- UlXioads the program in first partition at address KJOK. then instruction will find OS at absolute address 100.
- If instead of first partition, loader loads the program in second partition, the instruction will jump to address 100K+100.
- For third partition instruction will jump to address 200K+100 and so on. This problem is called as relocation problem.
- The solution to this problem is to in fact modify the instructions as the loading of program is done in memory.
- If programs are loaded in first partition then 100K is added to each address, programs loaded into second have 200K added to addresses, and so on.
- To carry out relocation at the time of loading like this, the binary program must have linker included list or bitmap indicating which addresses are to be relocated and which are opcodes, constants, or other items not needed to relocate.

4.1.6 Logical and Physical Address Space

Q. With the help of exempts, dearly differentiate between logical and physical address space.

- An address generated by CPU is called logical address ot virtual address and an address generated by Memory Management Unit (MMU) is called physical address.
- If size of program written by user is 1000 K and loaded in main memory from address 5000 to 6000.

and logical adtte collection of ogical 4 address JP®* program and a physical address J*' generated ny physical addresses correspond collection ot an k r

the8el o bmdirtg 1 f instruct ,ons " d to "tens.1

- Xe t at rfe time or load addresses art same. If *the* same b lrw «curs at run time then logical and physreal add are different.
- ™ location register (base register) contains
- IX) Physical address (5200) can be calcul * Zws. Consider logical address 200 of program.

Physical address (5200) = logical address (offtet) 1 + content of base register (5000)

- Manning from virtual to physical addresses is dorei, the memory management unit (MMU) at nin time. Th user program deals with logical addresses. Tbt memory-mapping hardware converts logical address into physical addressed logical addresses.
- Memory protection is done by using two registers, usually a base and a limit, as shown in Fig. 4.1.2. The base register holds the smallest legal physical memory address (in our example 5000). The limit register gives the size of the range of physical addresses. For example, if the base register holds 5000 and the limit register is 1000, then the program can legally access all addresses from 5000 through 6000.

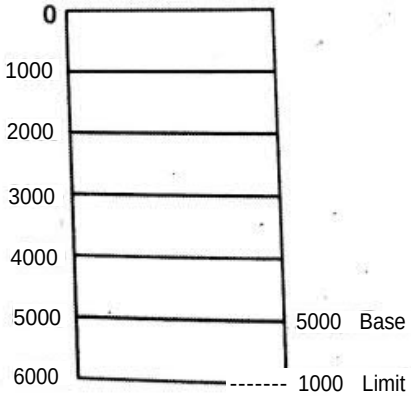


Fig. 4.1.2 : Memory Protection with base and limit registers

Syllabus Topic : Swapping

4.1.7 Swapping

- Systems in which it js possible mov e entire Presses between disk and main memory during are called swapping systems.

In a time sharing operating system, system's memory is allocated to multiple processes.

In order to have sufficient memory free, it will be required to swap data between memory and secondary storage (disk).

If we use round robin CPU scheduling algorithm then after expire of time quantum memory manager will swaps out the finished process and swaps in the another process.

It increases degree of multiprogramming ensures effective management of memory among many processes.

If scheduling is priority based, then after arrival of high priority process low priority process is swapped out on disk.

After completion of high priority process again a low priority process gets swapped in memory by memory manager.

Swapping needs a backing store to store a swapped data. The backing store is usually a fast disk and having capacity enough to store copies of all memory images for all users.

This backing store must offer direct access to these memory images.

The operating system maintains a ready queue for the processes which are ready to execute.

These processes have their memory images on the backing store or in memory.

The CPU scheduler calls the dispatcher program, when it decides to run the process. If dispatcher does not find next process in the queue in main memory, it swaps in the desired process.

if memory region is not free to bring the process from hacking store, dispatcher swaps out current process from the memory.

It then reloads registers and transfers control to the selected process. The context-switch time in such a swapping system is quite high

Syllabus Topic : Cnrti-unua Memory Allocation

4.2 Contiguous Memory Allocation

4.2.1 Multiprogramming with Fixed and Variable Partitions

Q. Explain memory management with Fixed and Variable Partitions

Q. What is internal fragmentation and external fragmentation? Explain with example

In this scheme memory is divided into a number of fixed-sized partitions. Each partition may contain exactly one process.

The number of partitions and its size would be defined by the system administrator when the system starts up. The degree of multiprogramming will be high if the number of partitions created is more.

Always a selected process from the input queue is placed into the free partition. When the executing process finishes the execution and terminates, the partition becomes free for another process.

The operating system maintain a table to keeps track on free and occupied partitions by processes.

Partitions can be of fixed size or of variable size. Fixed sized partitions are relatively simple to implement. The problems with the fixed size partitions are :

If program size is larger than the partition size, program cannot be stored in partition.

The second problem is **internal fragmentation.** If partition size is larger than program size, some space of the partition will be unused within that partition. Fig. 4,2.1 shows fixed size partitions.

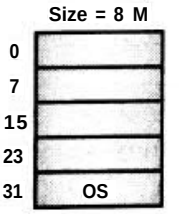


Fig. 4.2.1: Fixed size partitions

As a name suggest, in a variable- sized partition, partitions of different sizes are available. Memory is **partitioned in partitions of different sizes.**

The best-fit strategy is used to allocate the partitions to the processes. In best-fit, that partition is chosen to fit the process so that less amount of memory will be wasted.

It means smallest partition big enough to accommodate the process is chosen,

A queue is organized for each size of the partition where processes wait for that partition.

Partitions are allocated to the processes with best-fit policy Best fit ensures to allocate the partition that is big enough for the process to reside.

Therefore variable partitions minimize internal fragmentation.

- However, such an allocation may reduce the performance as processes would queue up in best fit queue even though other partitions are free.
- For example, it may have many processes queued up in a queue of partition of size 8 , and no processes are queued up in a queue for the parution size 16M.
- In both fixed and variable size partitions it may happen that, partition can remain empty even though processes are waiting in other partitions queue leading to external fragmentation. Fig. 4.2.2 Shows variable size partitions.

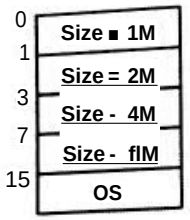


Fig. 4.2.2 : Variable size partitions

☛ **Difference between Internal fragmentation and external fragmentation**

Q. Compare internal and external fragmentation.]

Sr. No.	Parameter	Internal Fragmentation	External Fragmentation
1.	Partitioning technique	Internal fragmentation takes place in fixed partitioning technique	External fragmentation takes place in variable partitioning technique
2.	Definition	If partition size is larger than program size, some space of the partition will be unused within that partition called as internal fragmentation	in both fixed and variable size partitions it may happen that, partition can remain empty even though processes are waiting in other partitions queue leading to external fragmentation.
3.	Drawback	Contiguous memory allocation reduce the internal fragmentation	Paging and segmentation technique is tht solution over externa] fragmentation problem.

4.2.2 Dynamic Partition Technique

- Dynamic partition technique can reduce the internal and external fragmentations.
- In this technique basically a variable partitioning , variable number of partitions determined dynamical!,. Initially, all memory is available for user processes, and is considered as one large block, of available menm,, a hole.
- When a process arrives and needs memory, it search for a hole large enough to fit for this process.
- This technique ensures the efficient use of main memory. There is no internal fragmentation.
- However, if the many small holes are scattered, it can't be allocated to one large size process. Consider the following example of Fig. 4.2.3.
- Initially 64 MB main memory is available. 8 MB is allocated to operating system as shown in Fig- 4.2.3(a). For user processes 56 MB memory is available and considered as one large single hole.
- Processes P1, P2 and P3 are loaded and to tiiesc processes sufficient memory space can be allocated, creating the hole of size 4 MB at the end of menwr? (Figs. 4.2.3(b), (c) and (d)).
- At some point of time none of the allocated process is are ready.
- Operating system swaps out P2 and in its pl*# new process P4 of size 18 MB is swapped in Fig- 4.2-3(c)-
- The hole of 2 MB is created. Again after some tfo* since none of the processes in main memory is res ' and only P2 is in ready suspend state, is available.
- To swap in P2, there is no sufficient space is avail' 1,1, So operating system swaps P1 out and swaps P2 back »i as shown in Fig. 4.2.3(f).
- ShX plesbowsthat,,hereareman y scattered small

These holes cannot be allocated to the P ess o f Jarger CaJJed extern al fragmentation.

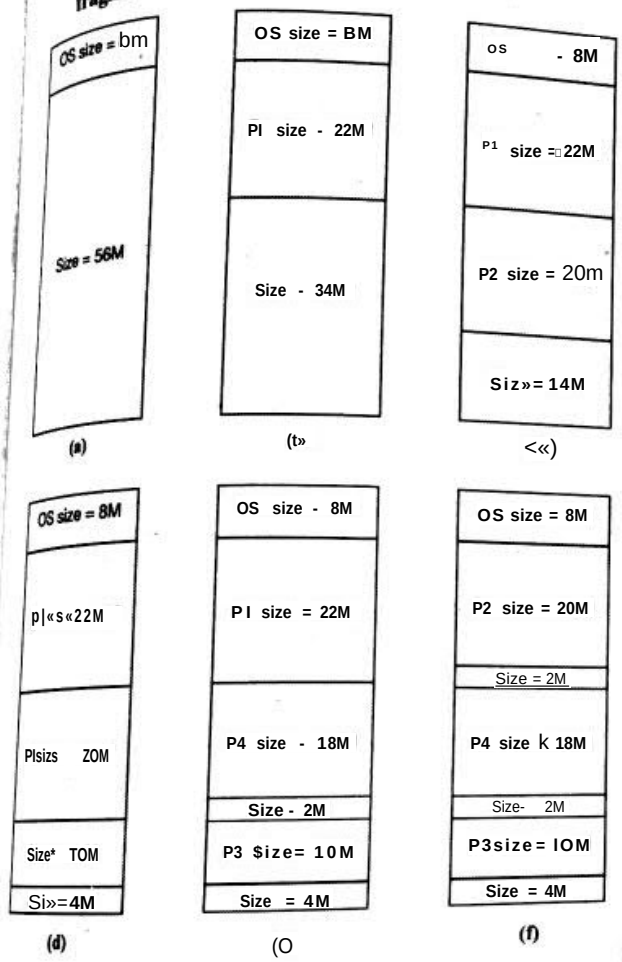


Fig.W : Creation of scattered holes in dynamic partition technique

<2.3 Compaction

Q. What is compaction? Explain with example.

Solution to above discussed problem is compaction

Compaction is a technique to covert many small s' ze Altered holes into one large size continuous hole . It is location of the various processes so as to accumulate all the free space in one place.

Due Io compaction, there is efficient use of process r. Fig. 4.2.3 shows the memory after compaction

4.2.4(a) shows rhe scattered small size holes. If processes P4 and P3 moved up. ®el on , fb) hole « the end of memory as shown in Fig-

- If processes P2, P4 and P3 niOve d downward we get one 1-rg size hole of 8 MB at the start of user memory area as shown in Fig. 4.2.4(c).
- However in this, 48 MB processes need to be moved compared to 28 MB shown in Fig. 4.2.4(b). ^t 4.2.4(d) shows the 28 MB shifting of processes.
- C mpaction should ensure the moving of processes hav ibg minimum total size. Compaction increases the degree of multiprogramming.
- It is because processes can be loaded in the memory which is available by combining many scattered holes. If relocation is done at compile time (static), compaction is not possible.
- II is possible only if relocation is dynamic and done at execution time.

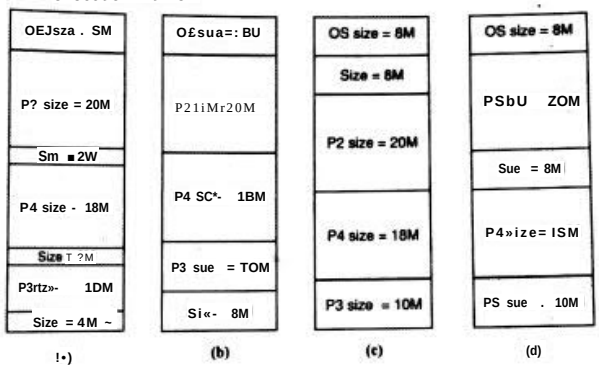


Fig. 4,2.4 : Compaction

4.2.4 Memory Allocation Strategies

-> **(May 201 6)**

q. Discuss partition selection algorithm in brief. **(MU ~ May 2016, 4 Maries)**

Q. Explain different memory allocation strategies.

If process requests the free hole, then question is which free hole should be allocated to this process. Following are the strategies used to allocate the free hole to the requesting process.

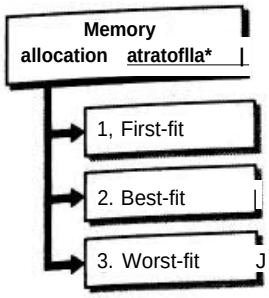


Fig. C4.1 = Memory allocation strategies

* tototXltegyof memory allocation, the available hole that is big enough to occupy an mcoX process allocated to that process. There are two options available to start the search

Searching can either initiate from the location when! previously search was stopped or at the start of d-e set of holes. Searching end * hen free hole that 1S big enough to hold the process is local ed.

lx :gy of memory allocation, the smallest enough to occupy an incoming process get allocated to that process.

– If the list of holes is not kept in sorted order of size. complete list needs to be searched. This approacn leads to create the smallest

<+ 3, **Worst-fit**

– b worst-fit strategy of meminy allocation, always the largest bote gets allocated to incoming process. Ag .

, If the list of botes is not kept in sorted order of size then the complete list needs to be searched Tins approach leads to create the largest leftover hole which can be used again to allocate the next incoming process

Example 4.2.1

Given memory partition of 100 K, 500 K, 200 K, 300 K, and 600 K in order, How would each of the First-fit, Best-fit and Worst-fit algorithms place the processes of 212 K, 417 K, 112 K and 426 K in order? Which algorithm makes the most efficient use of memory?

Solution :

First-fit

- 212 K is put in 500 K partition.
- 417 K is put in 600 K partition.
- 112 K is put in 288 K partition (new partition 288 K = 5OOK-212K).
- 426 K must wait.

Best-fit

- 212 K is put in 300 K partition.
- 417 K is put in 500 K partition.
- 112 K is put in 200 K partition.
- 426 K is pot in 600 K partition.

W'>rs,fit 600 K partition.

,12 K is pi* "> 600 . . ion.

' L 7 K 1S PU t * 500KPartl

: ; 1 2Kispu.to Kparti,iOT(6ooK - 2 i 2 K) -

- 426 K must wait ' in e ¡e. B «t-mto2 sout l bethelit5t

Example 4.2.2 MU - May 2016. 5 Marks

d18CO6S parton -oTselection algorithm in brief. Given memory ZQOk 300k and 550k (In orte r), how 500k. parttfc>n ° h «f the W fit. beat fit and worst lit algorithm place would 220k, 430k, 110k , 425k (in order), Which X'ZXkes the moat effident use of memory?

Solution :

First-fit

2 2OK is put m 500 K partition.

- 430 K is put in 550 K partition.
-]10 K is pul in 150 k partition.
- 425K must wait.

Best-fit

- 220 K is put in 300 K partition.
- 430 K is put in 500 K partition.
- 110 K is put in 150 K partition.
- 425 K is put in 550 K partition.

Worst-fit

- 220 K is put in 550 K partition.
- 430 K is put in 500 K partition.
- 110 K is put in 330 K partition (550 K - 220 K).
- 425 K mu'st wait.

In this example, Best-fit turns out to be the best.

Syllabus Topic : Paging

4.3 Paging

→ (June 15)

Q. Whaf is paging? MU - June 2015. 5 Marks

– Paging is memory management technique which allows Physical address space of the process * noncontiguous.

– In SCnsc , pa 8 in 8 mechanism is similar to nr to a book. When we read a book we only see and open the cunent toread

other pa.es are not le to Us and remains do**[1]

ta tbe san* manner, we can say even ...L

' L have a large program available, the orn ".....ssor only We gasman set of instructions to execute at any time

In fact, aH these instructions which the processor needs ' pjiecute are within a small proximity of each other

>s is like a page which contains all the statements whic h w e are currently reading.

in this way paging allows to keep just the parts of g process that we re using in memory and the rest on the disk.

4.3.1 Basic Operation

Q. 55haTisPage Table? Explain~Sr conversion o[p] Virtual Address to Physical Address in Paging with example.

- Ik problem of external fragmentation is avoided by using paging. Paging is implemented by integrating the paging hardware and operating system.
- Idthis technique, logical memory is divided into blocks oflie same size called pages.
- The physical memory is divided into fixed-sized blocks called frames (size is power of 2, between 512 bytes and 8192 bytes, also larger sizes possible in practice).
- The page size and frame size is equal. Initially all pages rtmains on secondary storage (backing store).
- When a process is needed to be executed, its pages are loaded into any available memory frames from the seco odary storage.

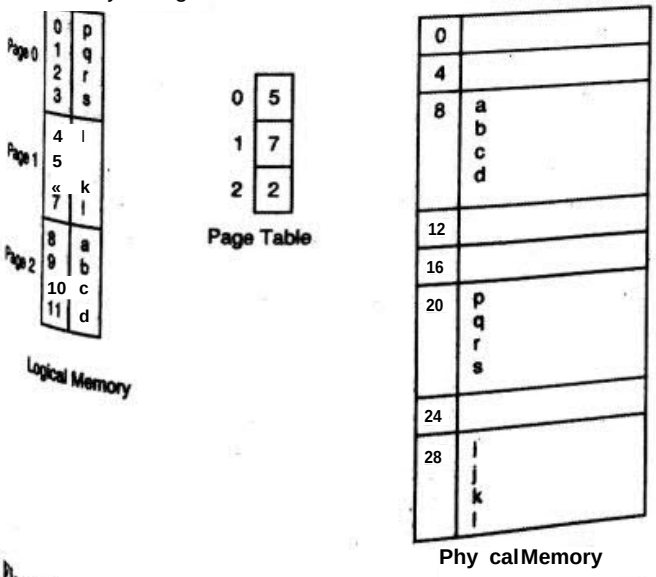


Fig. 4.3.1 : Paging model of physical w1 d l o c«" 1,<®,on ,

Fig. 4 3 1 ... the paging logical memo ws model of physical and Rowing basic operations is done in paging : '

two parts: d *o cal address and it is divided into Page number (p) and a page offset (d).

Th * Pafie numbe r is used as an index into a page table.

The base address of each page in physical memory is maintained by page table.

Jhe combination of base address with page offset mes the physical memory address that is sent to the memory unit.

Th e physical location of the data in memory is therefore at offset *d* in page frame f.

Because of the paging the user program sights the memory as one single contiguous space, it gr ves the illusion that memory contains only one program.

But the user program is spread throughout main memory (physical memory). The logical addresses are translated into physical addresses.

Fig, 4.3.2 shows the operation of the paging hardware.

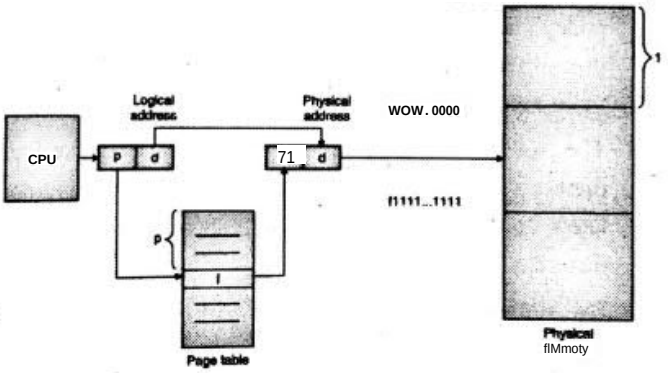


Fig. 43.2 : Paging hardware

Consider the following example of Fig. 4.3.3 Let page size is of 4 K and memory available is 32 K.

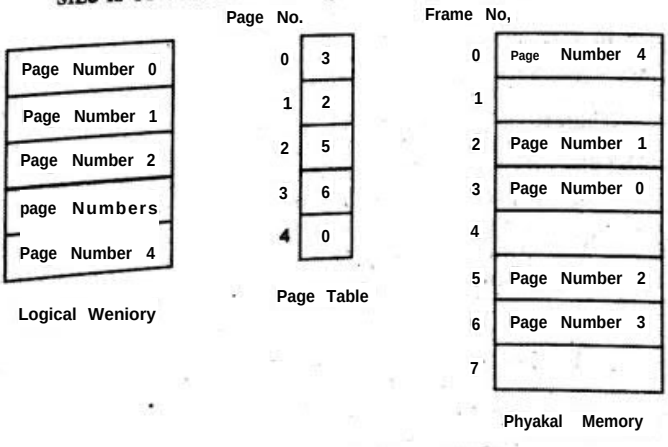


Fig. 433 : Paging example for a 32 K memory with 4K pages

FTOperating ------- ------- r
"TZoil frame nZbeTTpage 1 is in frame numbe

7 PageOisin frame number!

Logical address 4 is in page 1 and offset is 0. Page >*
mapped to frame *7*. *So* phy ical address 28 = ((7 × 4) +
offset 0). Logical address 10 Bin page an d offset 15 2,
Page 2 is mapped to frame 2.
So physical address 10 = ((2x 4) + offset).

4.3.2 Memory Protection and Sharing

- the frame number. So for the reference of the page, page table is verified to get correct frame number.
- Pforectiou bit is associated with each page to denote whether page is read-write or read only. oes to to write read only page, a hardware P g operating system.
- A legal page remains in logical address space of the process and in page table this page is marked with vahd (v) bit. Otherwise page is illegal and marked with invalid (i) bit
- Paging also provides advantages of sharing of common code. The re-entrant code cannot be modified. If many users share this code, only one copy needs to be kept in main memory.
- The page table of different user points to same physical memory (frames). In this case data used by different users can be different so data pages are mapped io different frames.
- By simply having page table entries for different processes point to the page frame, the operating system can create regions of memory that are shared by two or more processes.

4.3.3 Translation Look aside Buffer

In paging, in order to access a byte first page-table entry needed to be accessed and then byte is accessed.

Therefore, two memory accesses are needed to be performed. It results in slowing down memory access by a factor of 2. This delay would be unbearable under most situations.

- The above problem can be resolved by using translation look-aside buffer which is fast searching hardware cache. The TLB is associative, high-speed memory.
- The every entry in TLB is divided in two parts one is key and other is value.
- memory is presented with an item is The associauve m keys at same time. If there jj compared with E in g value field is returned. match
- Tme hardwure is expensive, the search ing Although tn rted , n thjs way n; very faR mechanism suppo nufflber rf entries. often Normally. TL» nolds small ranging the numbers tie tween M Md 1,024.

4 3 1 Effect of Page Size on Performance

➔ **(June 15, Nov. 15, Dec. 16)**

Q. Explain effect of page size on performance. 5 Marks
Z?i - June 2015. Nov.jOl DeoW

—7- portant factor to decide the perform ance.
Xe size is small then the any program will be 2dL in many numbers of pages leading to have a large page table.

- On some machine, it is required to ioad the page table in hardware register every time when context occurs. Hence more time will be required to load the larger sue page table due to small size of the pages.
- More number of transfers to and from disk will be required due to more number of page faults.
- Hence more time will be elapsed in seek and rotational delay. If page size is large then last page will remain somewhat empty leading to internal fragmentation. Due to large size of pages, most of the unused part of program can remain in memory.
- The size of the memory divided by the page size is equal to the number of frames. If the size of page is increased then the number of frame is decreases.
- Having a fewer frames will increase the number of page faults because of the lower freedom y replacement choice.
- Large pages would also waste space by Intemw Fragmentation.
- On the other hand, a larger page-size would draw in more memory per fault; so the number of fault tnay decrease if there is limited contention.
- Larger pages also reduce the number of TLB misses-

4-3.5 Hardware Support for Paging

➔ (De<>-16)

—plain the hardware support for paging.
MU - Dec. 2016, 10 Marks

Each operating system has its own methods for storing page tables.

Most allocate a page table for each process. A pointer to the pa£e ublc 1S stored wi tb the other register values (like the instruction counter) in the process contral block.

When the dispatcher is told to start a process, it must reload the user registers and define the correct hardware page-tabie values from the stored user page table.

The hardware implementation of the page table can be done in several ways.

In the simplest case, the page table is implemented as a set of dedicated registers. These registers should be built with very high-speed logic to make the paging address translation efficient.

Every access to memory must go through the paging map, so efficiency is a major consideration.

The CPU dispatcher reloads these registers, just as it reloads the other registers. Instructions i<£ load or modify the page-tabk registers are, of course, privileged, so that only the

The use of registers for the page table is satisfactory if the page table is reasonably small (for example, 256 entries).

Most contemporary computers, however, allow the page table to be very large (for example, 1 million entries).

For these machines, the use of fast registers to implement the page table is not feasible. Rather, the P®ge table is kept in main memory, and a page-table base register (PTBR) points to the page table.

Changing page tables requires changing only this one register, substantially reducing context- switch time. The Problem with this approach is the time required to access a user memory location. If we want to access Nation /, we must first index into the page table, using 016 value in the PTBR offset by the page number

Ef*4A1**

J'*' Paged system TLB hit ratio is 0.9. Let the RAM access ** t b «20ns. and TLB access time T be 100 ns. Find out

(1) Effective memory **access** with TLB

(2) Effective memory access without TLB

(3) Reduction in effective access time.

Solution :

TLB hit ratio, RAM access time t and TLB access time T is given in problem.

(1) Effective memory access with TLB (Ea)

Ea = {(TLB hit ratio) x (T + t)) + (1 - TLB hit ratio) x (2 T + t)

= (0.9 x (100 + 20) + (1 - 0.9) x ((2x100) + 20)

= 130 ns.

(2) Effective memory access without TLB (Ewt)

Ewt . = 2T = 200 ns

(3) Reduction in effective access time*

= (Ewt - Ea) x (T/Ewt)

= (200 - 130) x (100/200) = 35 %

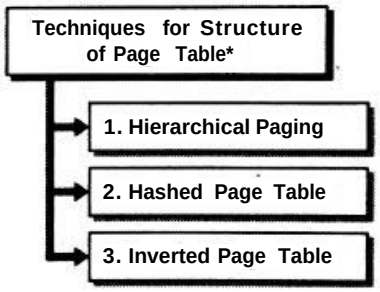
Syllabus Topic : Structure of the Page Table

4.4 Structure of Page Tables

➔ **(Dec. 14)**

Q. Discuss various techniques for structuring page tables. MU - Dec. 2014, 10 Marks

Following are the most common techniques for structuring the page table.



Fig, C4.2 : Techniques for structuring page tables

➔ 4.4.1 Hierarchical Paging

If the logical address space is very large such as 2^{33} or 2^{m}. the page table itself becomes extremely large.

In this case each process may require may large physical address space for the page table alone.

So it would not be advantageous to allocate the page cable contiguously in main memory. Solution to this problem is to split the page table into smaller pieces.

One method to achieve this is to use a two-level paging algorithm, in which the page table itself is also paged. Consider the system with 32 bit logical address space with page size 4 KB. A 32 bit logical address is divided in :

o 20 bit page number and

o 12 bit page offset

– Since paging of the page tables are done, a page number is again subdivide bove 20 bit a m :

O 10 bit page number and

o 10 bit page offset

– Thus the 32 bit logical address is divided as folio*[5] :

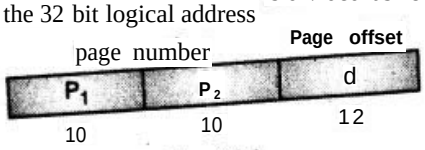


Fig. 4.4.1

where.

P, is on index into the outer page table and

P_2 is the displacement within the page of the outer page table.

– The address-translation scheme for this architecture is shown in following Hg. 4.5.2. As address translation works from the outer page table inward, this scheme is also known as a forward-mapped page table.

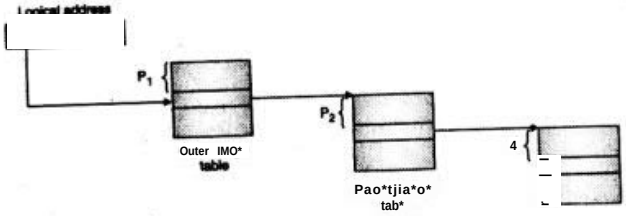


Fig. 4.4.2 : Address translation for a two-level 32-bit paging architecture

– Fig. 4.4.3 shows a two-level page-table scheme.

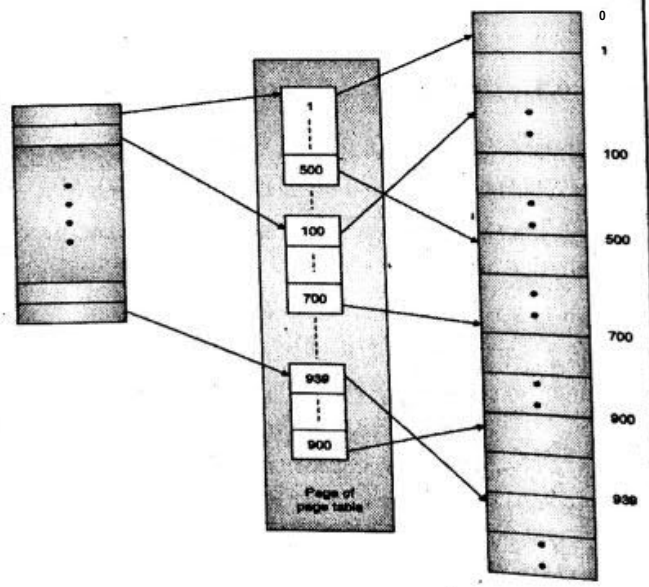


Fig. 4.43 : A two-level page-table scheme

→ 4 4 7T<ashe« PaS* Table

' ,hc address spaces are larger than 32 bits then % virtual page table is used in which hash value „ used w page number-

The elements which are hashed at same loe * V a M is maintained at each entry in the ha* table, Jt is necc3«"j to aV o id "* C0UiSi o n , E** elem... w-u consistsof three fields:

1. The virtual page number
2. The value of the mapped page frame
3. A pointer to the next element in the linl ed list.

– The working of the algorithm is as follows :

– Virtual address contains the virtual page number.-ft, virtual page number is hashed into the hash table.

– The comparison of virtual page number and the field| in the first element in the linked list is performed

– If match occurs then the corresponding page fim (field 2) is used to form the desired physical address.

– If there is no match then succeeding entries in de linked list are searched for a matching virtual w number. This scheme is shown in Fig. 4.4.4.

- A variant of this method that is good for 64-bit addra spaces has been proposed.

– In this, each entry in the hash table refers to a number of pages (such as 16) rather than a single page just Jilt hashed page tables.

– Due to this, a single page-table entry can store tv mappings for many physical-page frames. A gnW page tables are mainly useful for address spaces, wbeff memory references are non-contiguous and scatic.J throughout the address space.

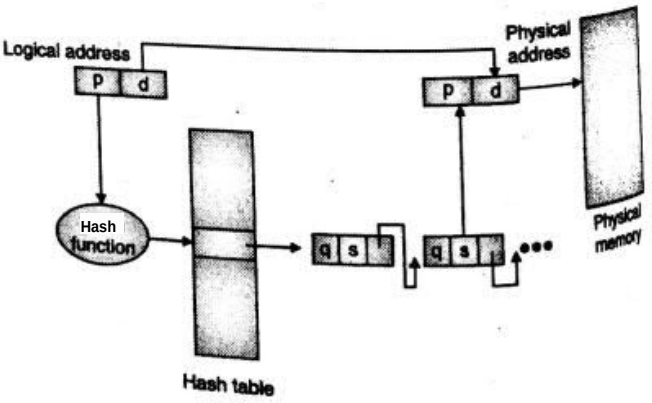
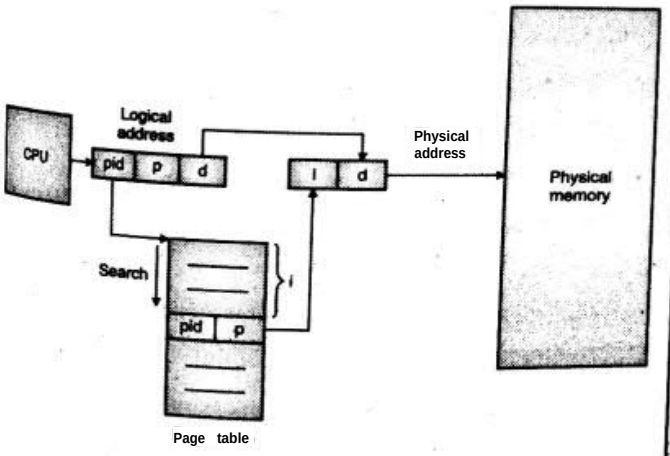


Fig. 4-4.4 : Hashed Page Table

4.4.3 Inverted Page Table

- Generally the operating system converts the Page mfercnce into a physical memory address
- Because the tabie is sorted by virtual address, the epemting system rs capable of computing ZT' able the associated physical address entrv I. i and to use that value directly. the ,ocated
- Tbe disadvantages of this scheme are that each page table may contain millions of entries.
- These tables may usc large amounts of physical memory. This problem is resolved with inverted page tables.
- There is one entry for each real page (or frame) i nory in inverted page table . h this entry the virtual aMess of the page stored in that real metnory loca(ion is present.
- Along with this virtual address, entry also contains the uformauon about the process containing this page.
- B se o f this onl y one Pago table is in the system, and it has only one entry for each page of physical memory. Fig. 4.4.5 shows the operation of an inverted Page table.



F<g- 4.4.5 : Inverted Page Table

- In °tder to explain this approach, example of the inverted page table used in the IBM RT is de ttfonstraied.
- £Vifiy virtuaj addre ss in the system is composed of a trip«. ' < Pr °cess-id f page-number, offset>.
- As shown in above Fig. 4.5.5 inverted page-table entry contains a pair <process-id. page-number>.
- The process-id cany out the work as a address-space identifier.
- At the ti... virtual adlZ" 1'?' rcfeienCe p,ace - P"t of the """tbens. is given to°X. IStU18 o f <proce "' id ' P a 8 e " of the inv e 2 sub «™- Searching pa * e tab 'e 's performed for a match.
- If there is < *offset* 'na<Ch ently i ' tflen "* P h X sicai address illegal address PrOdUCed. If there is access has been tried. " - * »
- In this sche m """ of "*■"*> «S- each p... to sea g h the table eissaved l »'t the amount of time needed when a page reference occur increases.
- Th ' el «irc table might need to be searched to get a C ' s search would consume more time.
- To ease this problem, a hash can be used limit the search to one-or at most few-page-table entries.

Syllabus Topic : Segmentation

4.5 Segmentation

Q- What is segmentation? Explain ft with example.

- In segmentation, a user program and its associated data can be subdivided into a number of segments. Different segments of the programs can have different length.
- Although there is a maximum segment length, the length depends on purpose of the segment in the program.
- Segments are arbitrarily-sized contiguous ranges of the address space.
- They are a tool for structuring the application. As such, the programmer or compiler may decide to arrange the address space using multiple segments.
- For example, this may be used to create some data segments that are shared with other processes, while other segments are not.
- Compiler cart create the different segment for global variables, code, thread stack, library etc.
- In segmentation, a program may occupy more than one partition, and these partitions need not be contiguous.
- Internal fragmentation is eliminated in segmentation but, like dynamic partitioning, it suffers from external fragmentation. In segmentation :
  - o Logical address it is divided into two parts: a segment number (s) and offset (d) into that segment.

noting System (MU - Sem 4 1 m ====g==aBSE ==

o Each entry of segnient table contains se8mentbase and segment limit.

Se nt base indicate stamngp cal address of

o nt base the segment m main memory and segme

denotes length of the segment.

Se nt number is used as index tq the segme

o

o X (d) always between 0 and segment limit Xwie « will occur indicating a«em P tmg accessbeyond the end of the segment Fig. . . indicates 4 segments of the program having different sizes. Segment table contains base an limit of each segment.

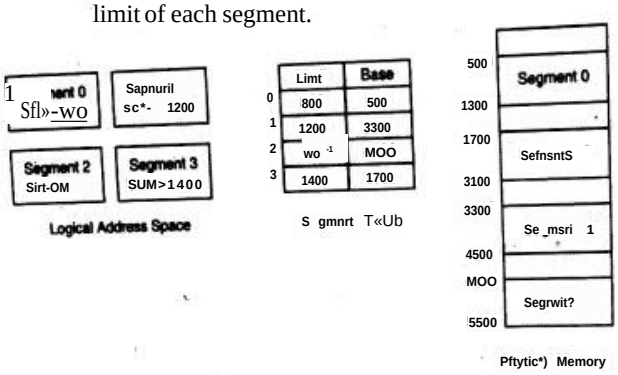


Fig. 45.1 : Segmentation example

Segment 3 is of 1400 bytes long and its base is 1700. Reference to byte 89 of sgment 3 is calculated as 1700 + 89 = 1789.

Example 4.5.1

On a system using simple segmentation, compute the physical address for each of following the logical address If address generates a segment fault than indicate so.

Segment	Base	Length
0	330	124
1	676	125
2	111	99
3	498	302

(I) 0,99 (II) 2, 7B (Hi)1,268 (h/) 3, 222 (v)o;i11

Solution :

Length given is limit of the segment.

(i) 0 f 99 : Segment 0 is present in "Smeot table. N 0wwe have to check (offset < limit) Orn ' » « « Kt is less than limit then physical address = ba » + offset

99<124 therefore physical address ~ 33O+ 99 '428

(ii) 2, 78 : Segment 2 is present in segment table. 78<99 therefore physical address = 111 + 78 = 189

(iii) 1 268 s segment lis present in segment table. 268<125 X) therefore segment fault and trap to Os. Not possible to calculate physical address

(Iv) ¥ **3 222** ■ Se 8ment 3 iS PrcSen ' ta SegnBnt * 222*302 therefore physical address = 498 + 222 = 720

(v) 0 111 : Segment 0 is present in segment table. [11<124 therefore physical address = 330 + 1[] - 44

Example 4.5 J

Consider the following segment table

Segment	Base	Length
0	219	600
1	2300	14
3	1327	580
4	1952	96

What are physical addresses of following logical addresses.

(i) 0. 430 (ii) 1,10 (iii) 2, 500 (iv) 3, 4000 (v) *4*, 112

Solution :

(i) 0, **430 :** Segment 0 is present in segment table. No* we have to check (offset < limit) or not If offset is less than limit then physical address — base + offset

430<600 therefore physical address ~ 2J4 + 430 - 6 0

(«) 1, 10 : Segment 1 is present in segment table. KK[4, Therefore physical address = 2300 + 10 = 2310

(0 2, 500 , Segment 2 is present in segment table. 5®< 5 8O (false) therefore segment fault and trap to OS, Not Possible to calculate physical address

(iv) - an/ 00 1 Se Bm «nt 3 is present in segment table. 400 8 o fore physical address = 1329 + = 1729

(v) <96 (fJ Scgmcnt 4 is Present in segment table- I[13 Not therefore segment faubanduaptoOS' pos S,WetOC alculate physical address.

4.5.1 Difference between Paging and Segmentation

Q. Explain the difference between paging and Segmentation. → (Dec. 0) Mu . Dec. 2016. 5 Marks

Sr-Ko- parameters	Paging	Segmentation
Logical address	C P(J lo « lcal address and 11 is divided into two parts: a page number (p j and a page offset (d).	Logical address it is divided into tWo Pans: a segment number (s) and offset (d) into that segment.
2. Table index	The Page number is used as an index into a p a2e table.	Segment number " us ed as index t0 the segment table.
1 Base address	The uaac aoaress of each page in Physical memory is maintained by page table.	Segment base indicate starting physical address of the segment in main memory and segment base denotes length of the segment.
4. Physical address	'The combination of base address with page offset defines the physical memory address that is sent to the memory unit.	Segment base indicate starting physical address of the segment in main memory and segment base denotes length of the segment
5 Fragmentation	Suffer through internal fragmentation	Suffer through external fragmentation

4.6 Segmentation with Paging

Q. Explain segmentation with paging.

- In segmenta
- Paging and seemem'* P , g " g , advantages of both mah! « me effect, ' e o n TM ple,nen,e<l t°8«l*r >0
- In paging pag protection also provided inTg "eXr " d Sharing
- In this tech
- multiple pages dT 11 of the 5eglwnls " div , d « i >n f~— - ■- - ths· wgnl e re t 1S 3,50 P'ge lable maintained
- Segment offset part of the logical address (containing offset and r— «gment number) is further 8plit " page number and page offset.
- Evcr y segment table (:ntry segment limit and inducdS segmajt basc - — ; ess of page table of that particular segment.
- An
- comnn portents 2 * F , 8 4 6 1 1 logical add ress has three Segment Number (SN)
- Page Number (PN) and page offset.
- Memory , management unit uses segment number as an -ndex into segment table in order to locate the address of page table.
- Then PN belonging to logical address is attached to high order end of address of page table and it is used as an index to page table to locate entry in page table.
- Frame number in page table entry is used to calculate physical address.
- This frame number is attached to high order end of page offset to obtain physical address.

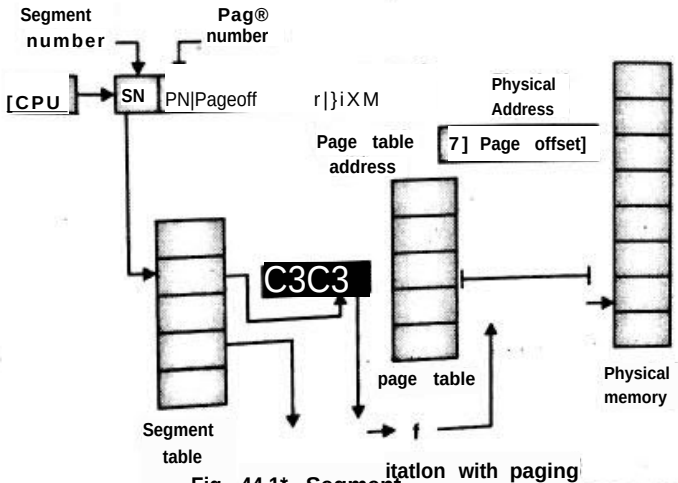


Fig. 44.1* Segment itatlon with paging

Syllabus T. J - ij.iliiitl t[1r ~'' >rv Ma n a 9° ttiet1 —

4.7 virtual Memory Management-.

4.7.1 Virtual Memory -> (May 16)

Q. Write short note on Vtrtt-l T..J 7 ,72016. 5 Mark!

- There are many eases where entire program not needed in main memory at a time.
- In many cases even though entire program is needed, it may not all be needed at (he same time.
- Application programs always gel the feel,ng availability of contiguous working address space due the idea of virtual memory.
- Actually, this working memory can be physically fragmented and may even overflow on to disk storage.
- This technique makes programming of large applications easier and use real physical memory more efficiently than those without virtual memory.
- Although executing process is not entirely present in memory, it can complete its execution due to virtual memory technique,
- Virtual memory permits execution of larger size programs although smaller size physical memory is available.
- It means larger size programs than available physical memory can complete the execution.
- Virtual memory concept separates the user logical memory from actual physical memory.
- This separation offers very large virtual memory to programmers although actual physical

Syllabus TopIcTpeman agin """

4.7.2 Demand Paging

→ (Dec. 14, Dec. 16)

Q. Write short note on Demand Paging,

MJ. Dec. 2014. PeT 2016. 5 Marks

Q, What is demand paging?

- A demand paging is paging system with Wll « where pages are brought in main nrem a fr o m secondary storage on demand.

- W \ ' s wapit fl * nWry ' , naM J f swapping "" cn,ire bto process memory, tas,ead ° Pajl ing *ll° ws t0 SWaP " o n ,y demand , pag for ej[ecut i On at that time. those pages are req

- WUdl failure in accessing the page, if it j, not The,e n dy Xped to a frame. This wiU cause 4 page fau lt which is a special type of interrupt.
- T , handle this interrupt, disk operation is initi „ O S handler to read the required page in memory.
- a»« this mapping of page to free frame is carried M During all this operation the process remains bl ed.
- As soon as c nage becomes available, the blocked process during disk operation is now unblocked, « placed on the ready queue.
- When scheduler schedules this process, it restarts m execution with the instruction that caused the w fault.
- The execution will continue until all the instructions are in memory This approach of fetching pages as they an needed for execution is called demand paging.
- Before swapping in the process in memory, the pager program makes sure about which pages will be used before the process is swapped out again.
- The complete process is not swapped in. Instead, tht pager brings only those required pages into memory-
- This would restrict the swapping in of pages in tnemon which are not needed for execution.

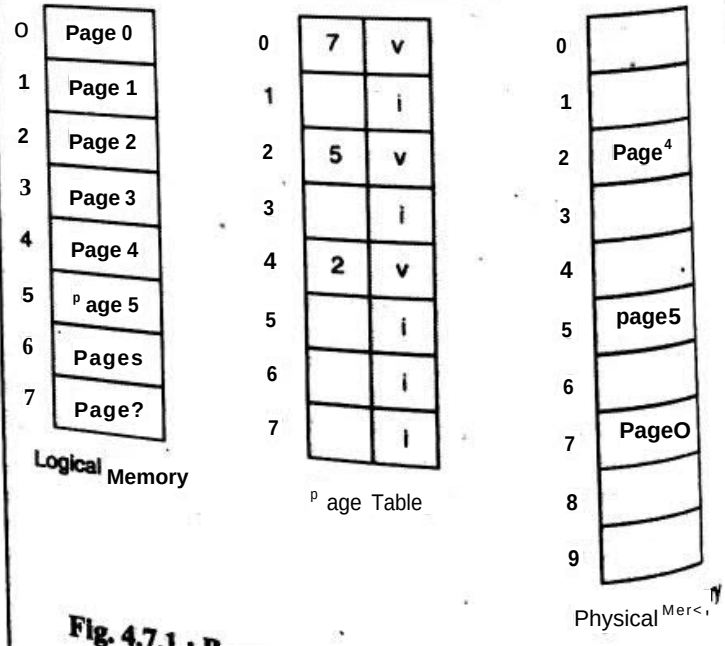


Fig. 4.7.1 : P " B * tab le with valid and invalid P ap *

- It saves swap time and the amount needed.
- in demand paging, valid and invalid bits are used to differentiate between those pages that are in memory and those pages that are on the disk
- Demand paging keep more processes in memory than the sum of their memory need
- utilization as high as possible. CPU

Fig. 4.7.1 shows page table with valid and invalid pages. Those pages which are in main memory are marked as valid. Pages on secondary storage are marked as invalid.

- Page 0, Page 2 and Page 4 are in main memory. So it is marked with valid (v) bit.
- Page 1 and page 3 are not in main memory so marked with invalid (i) bit.
- Page 1, page 3, Page 5, page 6, and page 7 are on secondary storage,
- Continuous allocation of memory to the pages is not necessary. The reason is, My page can be placed in any available page frame.
- The page table maintains the information regarding mapping of page to page frame. Due to this, it offers the impression of having one contiguous block of memory.

Advantages of Demand Paging

Q. List advantages of demand paging.

- Provides large size virtual memory.
- Memory is utilized more efficiently.
- Support for degree of multiprogramming is very good.

Disadvantage of Demand Paging

Q. List disadvantage of demand paging.

Page interrupts requires more number of table handling and Processor overhead with compare to simple paged management techniques.

4.7.3 Page Fault and Instruction Restarts

If the page is in secondary storage and not mapped to a page frame in main memory, a page fault occurs. (It is a trap to the operating system)

When page fault occurs operating system is responsible for :

- Either killing the process in the case of an invalid memory reference, or
- Locating the page into an instruction.
- Following is the procedure done by operating system for handling the page fault for any process.
- Internal table which is with the PCB of the process is verified for valid or invalid reference.
- For invalid reference process is terminated. For valid reference, if page yet is not present in main memory from secondary storage
- Find free frame from free frame list
- If disk is busy operating system has to wait. If not busy search the page on disk and read it into main memory.
- When reading of the page from secondary storage completes, internal table and page table of the process gets updated indicating page is now in main memory
- Restart the execution of interrupted instruction from newly transferred page.

Syllabus Topic : Copy-on-Write

4.8 Copy-on-Write

- The fork() system call is used to create child process which is a duplicate of its parent. Usually, fork() system call worked by creating a copy of the parent's address space for the child.
- While doing so it also duplicates the pages belonging to the parent. On the other hand, if we assume, many child processes call upon the exec() system call straight away after creation, the copying of the parent's address space may be needless.
- Copy-on-Write allows parent and child to share the same pages. If parent or child writes to shared pages then it is marked as copy-on-write and its copy is created.
- All unmodified pages can be shared by the parent and child processes.
- Pages that cannot be modified such as pages containing executable code can be shared by the parent and child. This technique used by Windows XP, Linux, and Solaris.
- As soon as it is known that a page is going to be duplicated using copy-on-write, it is significant to make

a note of the location from which the free page will be allocated. Several operating systems provide a pool of free pages for such request.

Syllabus : Page Replacement

4.9 Page Replacement Strategies

→ (Dec. 14)

- Q. Write short note on various page replacement policies. MU - Dec. 2014, 5 Marks
- Q. Explain the various page replacement strategies.

- After page fault handling, [illegible] to accommodate the new page [illegible] will remain always in main memory.
- For that purpose there are many page replacement [illegible] g it on a [illegible] expected.
- [illegible] number of page faults should be minimum to get the performance.
- Reference string is the string of memory references.
- The generation of the reference string is carried out artificially.
- The reference string can also be generated by listing the address of each memory reference by tracing the system.
- The important factor to determine number of page faults is the number of page frames available.
- These numbers of page faults are for a particular reference string and for page replacement algorithm for given available number of page frames.
- The number of frames available is inversely proportional to number of page faults.
- It means that if more frames are available less number of faults will occur.

4.9.1 FIFO Algorithm

→ (May 16)

- Q. Write short note on Belady's anomaly. MU - May 2016, 5 Marks
- Q. Compare Optimal, LRU and FIFO page replacement algorithms with illustration.
- Q. Compare FIFO and LRU page replacement algorithm.

- The simple page replacement algorithm and works on the basis of first in first out (FIFO). It throws out the pages in the order in which they were brought in. [illegible] associated with each page when it was [illegible]
- This algorithm always chooses oldest page for replacement.
- Since replacement is FIFO, old all the pages in [illegible]
- This algorithm doesn't care about which pages are accessed frequently and which are not. However, it is [illegible] in [illegible]
- Consider the reference strings. 5, 0, 2, 3, 0, 1, 3, 4, 5, 4, 2, 0, 3, 4, 3 and consider 3 available frames.

Reference string

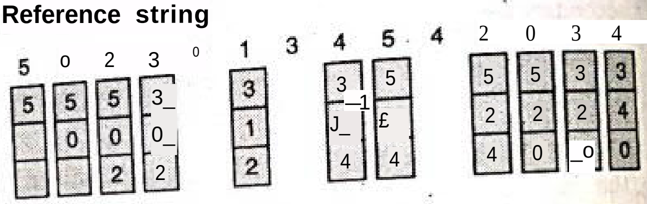


Fig. 4.9.1 : FIFO page replacement algorithm

- Since first three pages were initially not in main memory, references (5, 0, 2) causes page faults and brought into these 3 free frames.
- Reference to page 3 again causes the fault. To bring page 3 in memory as per FIFO policy, page 5 is replaced. Next reference is to page 0 and it is already in memory.
- To bring page 1 in memory again as per FIFO policy page 0 is replaced. Page 3 is already in memory. Reference to page 4 again causes the fault.
- To bring page 4 in memory as per FIFO policy, page 2 is replaced. This process continues as shown in Fig. 4.9.1.
- Whenever faults occurred, it is shown that which pages [illegible] together. There are in total 11 faults occur.
- This algorithm sometimes [illegible] result called as Belady's anomaly.
- That is [illegible] number of [illegible] rate [illegible] increase even though frames increases.

4.9.2 [illegible]

- Q. Compare [illegible] page replacement algorithms with illustration.

cw npare to all other page-replacement algorithms, m, gjgorithni has the lowest page-fault rate.

This algorithm replaces that page which will not be ujed for the longest period of time .

This delays the unavoidable page fault as much as possible, and thus decreases the total number of page faults.

As algorithm replace the page on the basis of future use of page, it is impossible to implement.

The optimal algorithm is unrealizable because it is impossible to determine the number of instructions that will be executed in the future before the page will be referenced.

- This algorithm does not suffer through Belady's anomaly.

, Again consider the same reference string 5, 0, 2, 3, 0, I 3, 4, 5, 4, 2, 0, 3, 4, 3

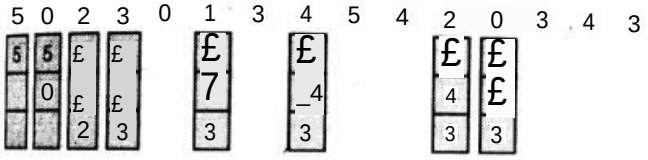


Fig. **4.9.2: Optimal page replacement algorithm**

- Since first three pages were initially not in main memory, references (5, 0, 2) causes page faults and brought into these 3 free frames.

Reference to page 3 again causes the fault. To bring P®ge 3 in memory as per optimal replacement policy, P[a]ge 2 is replaced because it is the page which will not be used for longest period of time.

Next reference is to page 0 and it is already in memory. To bring page 1 in memory again as per optimal [re]placement policy, page 0 is replaced.

Next reference is to page 0 and it is already in memory.

[Re]fetence to page 4 again causes the fault. Since page 1 *ill no longer be referenced, it is replaced. This process continues as shown in Fig. 4.9,2 causing in total 8 page faults.

4.9.3 Least Recently Used Page Replacement Algorithm (LRU)

Q. [C<M]We Optimal, LRU and HFO £*9® q. [re] placenwnt algorithms with Illustration.

Compare FIFO and LRU pafiP replacement algorithm.

- The time of page's last use is associated with each Page.
- When a page must be replaced, LRU chooses that page that was used farthest back in the past.
- LRU is a good approximation to the optimal algorithm.
- This algorithm looks backward in time while optimal replacement algorithm Looks forward in time.
- This policy suggests that replace a page whose last usage is farthest from current time.
- This algorithm can be implemented with some hardware support and is considered to be a good solution for page replacement.
- This algorithm does not suffer through Belady's anomaly.

We will again consider the same reference string 5, 0, 2, 3, 0, I, 3, 4, 5, 4, 2, 0, 3, 4, 3. Following is the result of applying LRU page replacement algorithm.

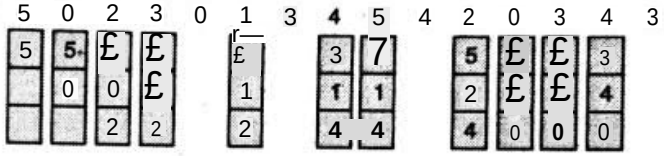


Fig. 4.9,3 : Least Recently Used page replacement algorithm

- Since first three pages were initially not in main memory, references (5, 0, ***2)*** causes page faults and brought into these 3 free frames. Reference to page 3 again causes the fault.
- To bring page 3 in memory as per optimal replacement policy, page 5 is replaced because it is the page which is used least recently.
- Next reference is to page 0 and it is already in memory.
- To bring page 1 in memory again as per LRU replacement policy, page 2 is replaced. Next reference is to page 3 and it is already in memory. This process continues as shown in Fig. 4.9.3 causing in total 11 page faults.

<r Implementation of LRU

Using Stack

- Initially keep all the page numbers on the stack.
- Remove the page from stack whenever it is referenced and place it on the top of the stack.
- Any time top of stack shows latest page number that is referenced and bottom shows the page which is not used for longest period of time.

Using Counters

- In a page table add time of use field with each page.
- k is incremented.
- Copy clock register content to time of use fiel - D
- Time of use field will show the time of last reference I the page.
- Replace the page having smallest time value.

4.9.4 LRU-Approximation Page Replacement

q, Explain LRU approximation approach. 7ZZ]

- Not all the system provides hardware for LRU replacement
- If hardware support not available then FIFO algorithm is used.
- The reference bit is associated with each page and stored in page table. If page is referenced for read or write then this bit is set. Initially this bit is set to 0 by OS, Once page is referenced then set to 1.
- This information is used for LRU-Approximation Page Replacement.

4.9.4(A) Additional-Reference-Bits Algorithm

- In this algorithm, 8 bit byte is associated with page and maintained in table in memory.
- After regular interval, OS is, interrupted which then shifts the reference bit for each page into the high-order bit of its 8-bit byte.
- Other bits are shifted right by I and low order bit is discarded. This 8 bit shift register shows the history of page use m last 8 time period. For example, following values of shift register shows the use of page as shown below.
  - o 00000000- Not used for last 8 time period
  - o 11111111 - Used at least once in each period
  - o 11000100 - This page is uscd more recently than one with a shift register value of 01101111
- The least recently used page is the lowest number when this shift register value is considered as unsiLS integer. 8ne<1
- The number of bits used is hardware pendent. If single bit used then it is second chance replacement.

4.9.4(B) Second-Chance Algorithm

- The FIFO algorithm is the basic algorithm of second replacement algorithm.
- replacement, its reference bit B cheeked
- If the reference bit is 0 (old unused then page is replaced.

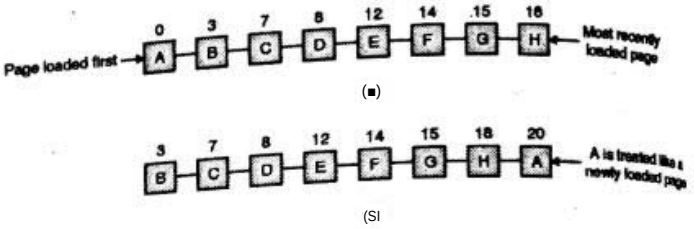


Fig. 4.9.4 : Second Chance Replacement

- If ihe reference bit is set to 1, a second chance is given to the page and then preceded on to select the «n FIFO page.
- After a page gets a second chance, its reference bit h cleared. Now this page is considered as arrived currently and its arrival time is reset to the current time.
- Because of this, page that got a second chance will not be replaced until all other pages have been replaced(or given second chances).
- In addition, if a page is used often enough to keep its reference bit set, it will never be replaced.
- Page A was initially loaded at time 0 and given i second chance. Its arrival time is now set to current time 20.

4.9.4(C) Enhanced Second-Chance Algorithm

- This algorithm considers pair of reference bit and modify bit. We get 4 combinations with two bits as:
  - (0, 0) – It is the best page for replacement which is neither recently used nor modified.
  - (0, !) _ H jn(i icates page is not g00 d foe replacement as it is not referenced but modified-fl need to be written out before replacement*
  - (1, 0) This page may be used again as it sb0*5 recently usedbutnotmodificd
  - THrL ThIS Page is rc ently used and modify page may be used again and hence need to be written out to disk before it is replaced.

4.9.4(D) Clock Page Replacement

Clock l»« e replacement algorithm maintains all ,k, frames on a circular list in the form of a clock.

The clock hand points to the oldest page.

After the occurrence of page fault, the page being pointed to by the hand is checked. If its reference bh * 0, the page is replaced and the new page is inserted into tjje Jock in its place.

After this insertion of new page, the hand is advanced one position. If reference bit is 1, it is cleared and the hand is advanced to the next page.

This process is repeated until a page is found with reference bit equal to 1.

4,9.5 Counting-Based Page Replacement

4.9.5(A) Not Frequently Used or Least Frequently Used Page Replacement Algorithm (NFU or LFU)

- This policy assumes that, those pages which are referenced for more number of times will be needed again.
- A counter is used for each page and incremented with each memory reference.

Suniv? Se with less COunter value should be replaced. Process P has not made use of page 2 and other pages have a count of usage like 3, 4 or even 5 times.

basic ar neJd d as compared to the page at 2. So page 2 should be replaced which is having less count.

Sutnent is that these pages may still be

pie limitation of the algorithm is that, if initially page is used frequently and then there is no reference to this Pa ge, still it will remain in memory.

4.9.5(B) Most Frequently Used Page Replacement Algorithm (MFU)

It is based on the assumption that, page having less count just arrived in memory and it is yet to be used.

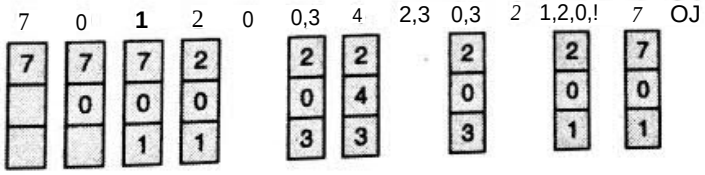
4.10 Examples on Page Replacement Algorithms

Example 4.10.1

For the page reference string 7. 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1. 2, 0. 1. 7, 0, 1. Calculate the Page Faults applying (i) Optimal (ii) LRU and (iii) FIFO Page Replacement Algorithms for a Memory with three frames.

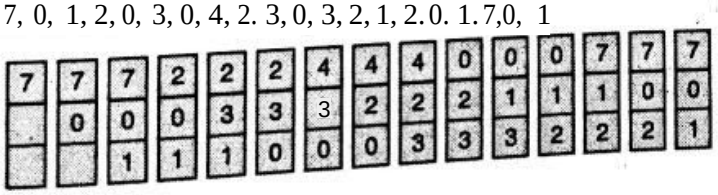
Solution:

Optimal Algorithm



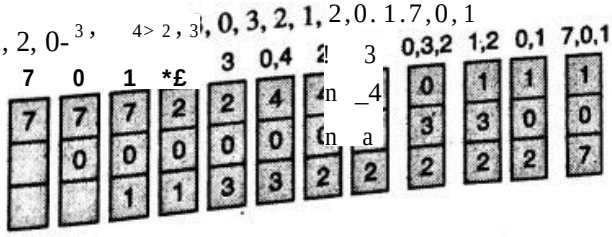
Tat alPage faults = 09

nF0 Algorithm: String = 7, 0, 1, 2, 0, 3, 0, 4, 2. 3, 0, 3, 2, 1, 2. 0. 1. 7, 0, 1



Total Page faulta = 15

LRU A1 8orithm: String = 7, 0J , 2, 0- 3 , 4> 2 , 3 , 0, 3, 2, 1, 2, 0. 1. 7, 0, 1



Total Page faults = 12

Example 4.10.2

Consider the following page reference string: 1, 2, 3. 4, 2, 1, s. "• 2, 1, 2, ' 7 6 3. 2. 1, 2, 3, 6.How many page faults Mr all frames are initially empty, so * I frames? Re.... you, Inst occur tor the following replacement algorithms, assuming tour unique pages wilt all cost one fault each.

(1) LRU replacement (2) FIFO replacement

Solution :

(1) LRU

String 1 2 3 4 2 1 5 6 2 1 2 3 6 3 2 1 2 3 6.

	1	1	1	1	1	1	1	1	6	6							
		2	2	2	2	2	2	2	2	2							
			3	3	5	5	3	3	3	3							
				4	4	6	6	7	7	1							

Page faults for LRU = 10

(2) FIFO

String 1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6.

	1	1	1	1	5	5	5	5	3	3	3	3	1	1			
		2	2	2	2	6	6	6	6	7	7	7	7	3			
			3	3	3	3	2	2	2	2	6	6	6	6			
				4	4.	4	4	1	1	1	1	2	2	2			

Page faults for FIFO = 14

Example 4.10 J

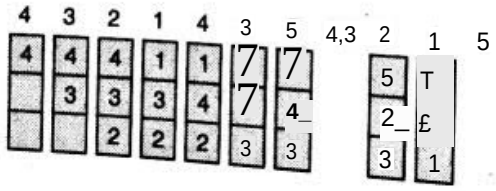
Consider following page reference string

4.3,2,1,4AM, 3,2.1,5

Assume frame size = 3, How many page faults would occur for FIFO, Optimal, LRU algorithm ?

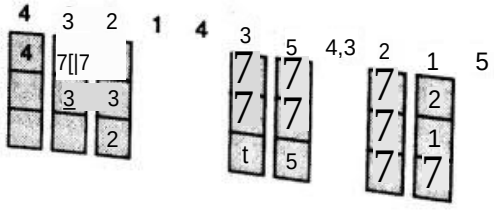
Solution :

(1) **FIFO page replacement algorithm**



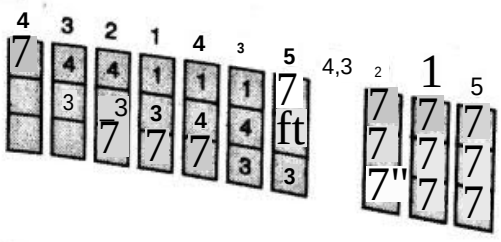
Total number of page faults - 9

(2) Optimal page replacement algorithm



Total number of page faults = 7

(3) LRU page replacement algorithm



Total number of page faults = 10

Example 4.10.4 MU - Nov. 2014. 10 M 3*?

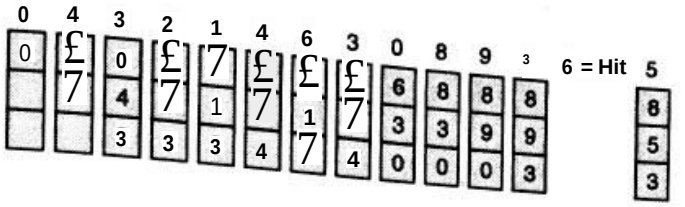
Calculate Hit and Miss using (flu, optimal, FIFO page raptao *'W Policies for , he following sequence.

p8ge(ramesi zeis3.0,4,3 >2,M,6,3,0.8, 9 ,3i8>5

Solution :

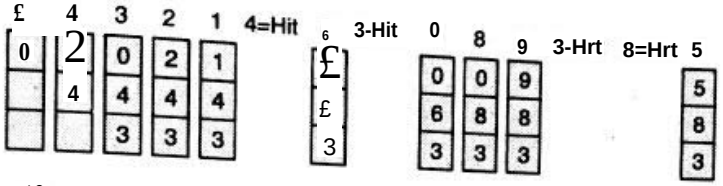
Consider frame size = 3

(1) tB0



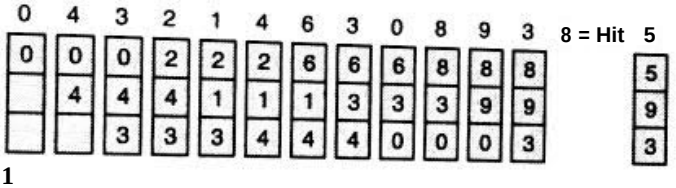
Number of hits = 01, Miss -]3

(2) Optimal



Number of hits = 04, Miss = 10

(3) FIFO



Number of hits = 01, Miss = 1

Ewnple 4.10.5 MU - May 2016, 10 Marks

Cafctrlate hit and miss percentage for the following string using page replacement policies FIFO, LRU and Optimal. Compare for the frame size 3 and 4. Page string 2,0,3,0,4, 2,3,0,3,2,7,2,0,7,5,0,7,5,7,0

Solution:

FrameSize = 3

FIFO Page Replacement

Number of Hits - 8 Miss _ 12

Frame	2	0	3	0	4	2	3	0	3	2	7	2	0	7	5	0	7	5	7	0
0	2	2	2	2	4	4	4	4	3	3	3	3	0	0	0	0	0	0	0	0
1		0	0	0	0	2	2	2	2	2	7	7	7	7	5	5	5	5	5	5
2			3	3	3	3	3	0	0	0	0	2	2	2	2	2	7	7	7	7
PF	Y	Y	Y	-	Y	Y	-	Y	Y	-	Y	Y	Y	-	Y	-	Y	-	-	-

Opti M Page Replacement

Nu n er £ Hits = j3 d Miss = 7

Frame	2	0	3	0	4	2	3	0	3	2	7	2	0	7	5	0	7	5	7	0
0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	5	5	5	5	5	5
i		0	0	£	4	4	4	0	0	0	0	0	0	0	0	0	0	0	0	0
2			3	3	3	3	3	3	3	3	7	7	7	7	7	7	7	7	7	7
PF	Y	Y	Y	-	Y	-	-	Y	-	-	Y	-	-	-	Y	-	-	-	-	-

Repl.ecment

Number of Hits = HJaruf Miss® fn

L Frame	1 2	0	3	©	4	2	3	0	2	2	2	2	0	7	5	0	7	5	7	0
0	2	2	2	2	4	4	4	0	0	0	2	7	7	7	7	7	7	7	7	7
1		0	0	0	0	0	3	3	3	3	3	3	0	0	0	0	0	0	0	£
12			3	3	3	2	2	2	2	2	2	2	2	2	5	5	5	5	5	S
PF	Y	Y	Y	-	Y	Y	Y	Y	-	-	Y	-	Y	-	Y	-	-	-	-	-

Fmmc Size » 4

Wo Page Replacement

N“mhcrc>fHits= I andM ·M = g

f F me	2_	O	3	0	4	2	3	0	3	2	7	2	2	2	5	£	7	5	7	0
rp	2	2	2	2	2	2	2	2	2	2	7	2	7	7	7	7	2	7	7	J
[j		0	0	0	0	0	0	0	0	0	0	2	2	2	2	2	2	2	2	J
2			3	3	3	3	3	3	3	£	3	3	0	2	0	2	£	2	£	£
3					4	4	4	4	4	4	4	4	4	4	5	5	5	5	5	5
PF	Y	Y	Y	-	Y	-	-	-	-	-	Y	Y	Y	-	Y	-	-	-	-	-

Frame Size = 4

Optimal Page Replacement

Number of Hits = 14 and Miss = 6

I Frame	2	0	3	0	4	2	3	0	3	2	7	2	0	7	5	0	7	5	7	0
0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2			3	3	3	3	3	3	3	3	7	7	7	7	7	7	7	7	7	7
3					4	4	4	4	4	4	4	4	4	4	5	5	5	5	5	5
PF	Y	Y	Y	-	Y	-	-	-	-	-	Y	-	-	-	Y	-	-	-	-	-

Frame Size = 4, LRU Page Replacement

Number of Hits = 14 and Miss = 6

Frame	2	0	3	0	4	2	3	0	3	2	7	2	0	7	5	0	7	5	7	Q
0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2			3	3	3	3	3	3	3	3	3	3	3	3	5	5	5	5	5	5
3					4	4	4	4	4	4	7	7	7	7	7	7	7	7	7	7
PF	Y	Y	Y	-	Y	-	-	-	-	-	Y	-	-	-	Y	-	-	-	-	-

When *frame size = 4 then number of hits* are mote all algorithms for given page string.

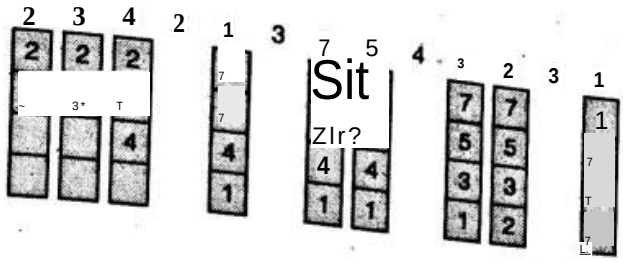
Example 4.10.6 MU - June 20T5. 10 Marks

What is paging. Explain LRU t FIFO, and OPT page replacement policy for the given page sequences. Page frame sizeis* *2 f 3, 4, 2, 1, 3, 7, 5, 4, 3, 2, 3, 1*

Calculate page hit and miss.

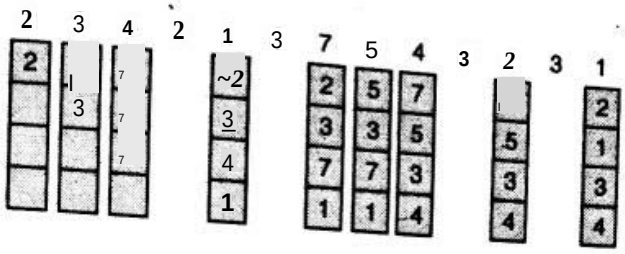
Solution :

(i) FIFO



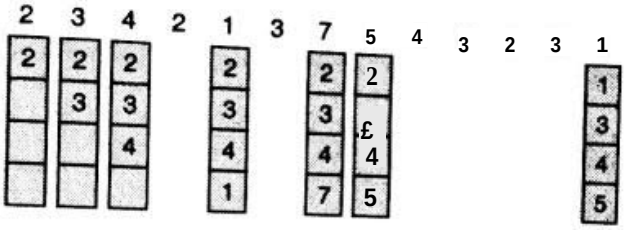
For FIFO Hit = 4 and Miss = 9

(ii) LRU



For LRU Hit = 4 and Miss = 9

(jj) Optimal



For Optimal Hit = 6 and Miss = 7

Example 10.7 MU - Dec- 2016. 10 Marks

Calculate hit and miss for the following string using page replacement policies-FIFO, LRU and Optimal. Compare it for the same size 3 & 4.

1 2 3 2 1 5 2 1 6 2 5 6 3 1 3 6 1 2 4 3

Solution:

For Frame Size = 3

Wo

Frame	1	2	3	2	1	5	2	1	6	2	5	6	3	1	3	6	1	2	4	3
0	1	1	1	1	1	5	5	5	5	2	2	2	2	1	1	1	1	1	4	4
1		2	2	2	2	2	2	1	1	1	5	5	5	5	5	6	6	6	6	3
2			3	3	3	3	3	3	6	6	6	6	3	3	3	3	3	2	2	2
PF	Y	Y	Y	-	-	Y	-	Y	Y	Y	Y	-	Y	Y	-	Y	-	Y	Y	Y

(ii) LRU

Frame	1	2	3	2	1	5	2	1	6	2	5	6	3	1	3	6	1	2	4	3
0	1	1	1	1	1	1	1	1	1	1	5	5	5	1	1	1	1	1	1	3
1		2	2	2	2	2	2	2	2	2	2	2	3	3	3	3	3	2	2	2
2			3	3	3	5	5	5	6	6	6	6	6	6	6	6	6	6	4	4
PF	Y	Y	Y	-	-	Y	-	-	Y	-	Y	-	Y	Y	-	-	-	Y	Y	Y

(Ii«j upumai

Frame	1	2	3	2	1	5	2	r	6	2	5	6	3	1	3	6	1	2	4	3
0	I	1	1	1	1	1	1	1	6	6	6	6	6	6	6	6	6	2	4	4
1		2	Z	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1
2			3	3	3	5	5	5	5	5	5	5	3	3	3	3	3	3	3	3
PF	Y	Y	Y	-	-	Y	-	-	Y	-	-	-	Y	Y	-	-	-	Y	Y	-

Frame Size = 4

(i) FIFO

Frame	1	2	3	2	1	5	2	1	6	2	5	6	_3_J	1	3	6	i 1	£T	4	3
0	1	1	1	1	1	1	1	1	6	6	6	6	6	6	6	6	I 6	6 T	6	6
1		2	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1 1	1 1	I	1
2			3	3	3	3	3	3	3	3	3	3	3	3	3	3	2	J		2
3						5	5	5	5	5	5	5	5	5	5	15	15	i 5	14	4
PF	Y	Y	Y	•	•	Y	-	- .	Y	-	-	-	-	Y	-	I -	-	i Y	Y	Y

(it) LRU

Frame	1	2	3	2	1	5	2	1	6	2	5	6	3	1	3	6	1	2		
0	1	1	1	I	1	1	1	1	1	1	J	1	3	3	3	3	3			
1		2	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1			
2			3	3	3	3	3	3	6	6	6	6	16	6	6	6	16			
3						5	5	5	5	5	5	5	5	5	5	5	5	2	2	2
PF	Y	Y	Y	-	-	Y	-	•	Y	-	1 _	•	Y	Y	-	-	-			Y

(iii) Optimal

Frame	1	2	3	2	1	5	2	1	6	2	5	6	3	1	3	6	1	2	4	3
0	1	I	1	1	1	1	1	1	6	6	6	6	6	6	6	6	6	6	4	4
1		2	2	2	2	2	2	2	2	2	2	1	2	2	2	2	2	2	2	2
2			3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
3						5	5	5	5	5	5	5	5	1	1	1	1	1	1	1
PF	Y	Y	Y	-	-	Y	-	-	Y	-	-	-	-	Y	-	-	-	-	Y	-

Frame Size = 3		Hit	Miss
	FIFO	6	14
	LRU	9	11
	OPTIMAL	11	9

Frame Size = 4		Hit	Miss
	FIFO	11	9
	LRU	10	10
	OPTIMAL	1 1 3	1 7

Example 4.10.8 MU - Dec 2015 10 Marks

Consider the following page reference string: 1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6 How many page faults would occur for the followios replacement algorithms, assuming three ff L ' ' 6 ' 3 ' *■ 1 • *• 3 - 6 res ' fiv9 ,r «™»forLRU, FIFO and optimal pagerepiacemanty SOW** 1 1

Frame size = 3

LRU 15 page faults ii) FIFO, 16 page faults iii) Optimal. n faute

The ••Y" in bottom rows denote the page fault

(i) LRU

Frame	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
0	1	1	1	4	4	4	5	5	5	1	1	1	7	7	7	2	2	2	2	2
1		2	2	2	2	2	2	6	6	6	6	3	3	3	3	3	3	3	3	3
2			3	3	3	1	1	1	2	2	2	2	2	6	6	6	1	1	1	6
PF	Y	Y	Y	Y	-	Y	Y	Y	Y	Y	-	Y	Y	Y	-	Y	Y	-	-	Y

fu)H?O

Frame	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
0	1	1	1	4	4	4	4	6	6	6	6	3	3	3	3	2	2	2	2	6
1		2	2	2	2	1	1	1	2	2	2	2	7	7	7	7	1	1	1	1
2			3	3	3	3	5	5	5	1	1	1	1	6	6	6	6	6	3	3
PF	Y	Y	Y	Y	-	Y	Y	Y	Y	Y	-	Y	Y	Y	-	Y	Y	-	Y	Y

ffl Optimal

Frame	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
0	1	1	1	1	1	1	1	1	1	1	1	3	3	3	3	3	3	3	3	3
1		2	2	2	2	2	2	2	2	2	2	2	7	7	7	2	2	2	2	2
2			3	4	4	4	5	6	6	6	6	6	6	6	6	6	1	1	1	1
PF	Y	Y	Y	Y	-	-	Y	Y	-	-	-	Y	Y	-	-	Y	Y	-	-	Y

If frame size = 5 then

LRll gives 8 page faults ii) FIFO, 10 page faults iii) Optimal, 7 page faults

Boldpage number shows the fault.

® LRU • i, 2, 3, 4, 2, L 5, 6, 2, U 3 * 3 * 2, 1 2 3 6

®> FIFO : j, 2, 3. 4, 2, 1, 5, 6, ***2,*** 1. 2, 3. " * * * " 3> 6

®> Optima]. 1, 2, 3, 4, 2, 1, 5. 6, 2, 1, 2, 3. • '

Example 4.10.9 MU - Dec 2016. 10 Marks

Calculate number of page faults and page Hits fw the page -oiacement potoes FIFO. Optimal and LRU tor given reference string «■0, 5, 2, o. 3, 0, 4, 2, 3. 0. 3, 2. 5. 2. 0, 6, «■ 6 .1 assumjng three frame srza).

Pullon:

Frame Sitta 3

FIFO Page Replacement

Number of Hits = 5 and Miss = 15

Frame	6	0	5	2	0	3	0	4	2	3	0	3	2	5	2	0	5	6	0	5
0	6	6	6	2	2	2	2	4	4	4	0	0	0	0	0	0	0	6	6	6
1		0	0	0	0	3	3	3	2	2	2	2	2	5	5	5	5	5	0	0
2			5	5	5	5	0	0	0	3	3	3	3	3	2	2	2	2	2	5
PF	Y	Y	Y	Y	-	Y	Y	Y	Y	Y	Y	-	-	Y	Y	-	-	Y	Y	Y

Optimal Page Replacement

Number of Hits = 12 and Miss = 8

Frame	6	0	5	2	0	3	0	4	2	3	0	3	2	5	2	0	5	6	0	5
0	6	6	6	2	2	2	2	2	2	2	2	2	2	2	2	2	2	6	6	6
1		0	0	0	0	0	0	4	4	4	0	0	0	0	0	0	0	0	0	0
2			5	5	5	3	3	3	3	3	3	3	3	5	5	5	5	5	5	5
PF	Y	Y	Y	Y	-	-	-	Y	-	-	Y	-	-	Y	-	-	-	Y	-	-

LRU Page Replacement

Number of Hits = 8 and Miss = 12

Frame	6	0	5	2	0	3	0	4	2	3	0	3	2	5	2	0	5	6	0	5
0	6	6	6	2	2	2	2	4	4	4	0	0	0	5	5	5	5	5	5	5
1		0	0	0	0	0	0	0	0	3	3	3	3	3	2	0	0	0	0	0
2			5	5	5	3	3	3	2	2	2	2	2	2	2	2	2	6	6	6
PF	Y	Y	Y	Y	-	Y	•	Y	Y	Y	Y	-	-	Y	-	Y	-	Y	-	-

Syllabus Topic : Allocation of Frames

4-11 Allocation of Frames

- Opentiag system maintains free frame list. Let us take example of single-user system with 512 KB of memory consisting of pages 4 KB in size
- This system has 1_{28} frames each of 4 KB in size The operating sys i era may occunv ts f„ 93 frames are for the user process. S „ d rcmaini »g
- If we consider the pure demand paging all <n would at the start be put on the free-ftatt fet - frames
- Once user process begins execution, it would produce a sequence of page faults.
- The initial 93 page faults W0Uld all get fKe frames from the free-frame list.
- When the free-frame list was worn out, a $P^{0}®^{6}$ replacement algorithm would be used to select onecf the 93 in-memory pages to be replaced with the 94, and so on.
- If the more number of frames allocated to process, the P^{a} gc~fault rate decreases.
- If less number of frames allocated to each process» the page-fault rate increases, slowing process execution.
- If P^{a} gc fault occurs prior to an executing instnKtio[11]is Uniplete, the instruction must be started again- As a result, We must sufficient frames to hold aU the diffeiem pages that any single instruction can Tr„ f ercnce.

Skate ? ie> for allocation of frames

Q. Explai[erent strategies for allocation of Irames.

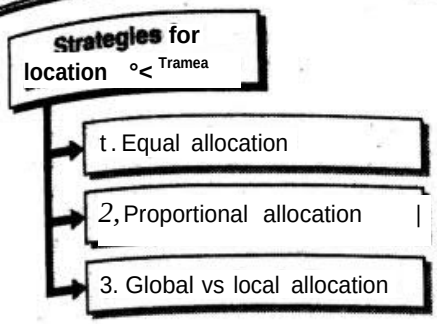


Fig. C43 : Strategies for a oca on ®f frames

b gqualallocHti°n

Consider p number of processes and q number of frames. In this case every process will get p/q frames, to cs ample suppose 5 processes and 93 frames present. Then 93/5, that is each process will get 18 frames and 3 will remain at free-frame list.

* t **Proportionalallocation**

h this allocation, frames are allocated to the processes as per their needs. Suppose 60 free frames available. Process Pl has size 10 KB and P2 has size 128 KB. And if page size is I KB. Then (10/(10+128)) ♦60-4.34, say nearly 4 frames allocated to Pl. P2 win get (128/(10+128)) * 60 = 55.65, that is 55 frames.

4 X Global vs Local allocation

* Pace replacement algorithm is categorized as global replacement if it can choose frame for replacement fom all frames although currently it is allocated to other process also.

h means processes can take frames from other Processes. Local replacement needs to replace frame by Excess only from its own set of allocated frames.

The difficulty with a global replacement algorithm is that a process cannot control its own page- fault rate,

The set of pages in memory for a process depends n ot 'W on the paging actions of that process but also on the paging actions of other processes.

Conseqoe-ndy, the same process may perform fahty diff trendy.

Syllabus Topic : Thrashing

4.12 Thrashing

→ (June 15)

Q. What is thrashing ? MU - Junc 2015. 5 Marks

- A process should have some minimum number of frames to support active pages which are in memory- It helps to reduce the number of page faults.
- If these numbers of frames are not available to the process then it will quickly page fault.
- To handle this page fault, it is necessary to replace the existing page from memory.
- Since all the pages of the process are active, it will also need in future. So any replaced page will cause page fault again and again.
- Since in paging, pages are transferred between main memory and disk, this has an enormous overhead. Because of this frequent movement of pages between main memory and disk, system throughput reduces.
- This frequent paging activity causing the reduction in system throughput called as thrashing.
- Although many processes are in memory, due to thrashing CPU utilization goes low.
- When operating system monitors this CPU utilization, it introduces new process in memory to increase the degree of multiprogramming.
- Now it is needed that the existing pages should be replaced for this new process.
- If global page replacement algorithm is used, it replaces the pages of other process and allocates frames to this newly introduced process. So other processes whose pages are replaced are also causes page faults.
- AU these faulting processes go in wait state and waits for paging device. In this case again CPU utilization goes low.
- When CPU scheduler observes the low CPU utilization, it introduces new processes again in system to improve degree of multiprogramming. This also results in page faults of running processes.
- Again the waiting queue of paging device becomes long causing further drop in CPU utilization.
- In this way page fault rate further increases and CPU utilization decreases. There is no actual work getting done and processes spend time only in pagtng.
- This thrashing can be limited by using local page „ placement algorithm instead of global page replacement algorithm.
- [illegible] page replacement algorithm replaces the pages of IL process instead of other process. Therefore f s of the other processes will not be taken and these processes wffl not thrash.

- Thrashing can be prevented by providing sufficient frames to the process as pen«rnted ts need.
- Locality model of proces s execution solve this problem by keeping track on number of frames process B is the set of active pages used currently using. I- ™
- During execution proce« moves tom locality to locality. Working Set Model is based on this assumption of locality. It examines most recent page references. tOgether
- All the pages in most recent page references are th e working set. Suppose working set window uze defm d is g. At time tI if the page references are (2, 3, 2, 3, 4, 3, 3, 4) then working set is {2,3,4} and 3 frames are required.
- At this time if demand of the process is more that working set, thrashing will occur. Operating system will monitor working set of the process and allocate needed frames.
- If sufficient frames are not available it will suspend the process.In this way working set model helps to prevent the thrashing

Syllabus Topic : Memory-Mapped Files

4.14 Memory-Mapped Files

Q. Explain memory-mapped files.

- With compare to accessing ordinary memory, accessing files is burdensome.
- To overcome the problem, files can be mapped to the address space of the executing process.
- Two system calls, map and unmap, can be conceptually used for this purpose.
- System call causes specified file map into a address space at a virtual address. the <« E. systcm to ntap
- Consider the file of length 12g KB which is manned mto the virtual address starting at address 1024 K.
- Then any machine Instruction that reads th*- ♦ the byteat 1024 K gets byte 0 of the fife, and " **
- Similarly, a write to address 1024 K + inno byte 1000 of the file. Wta«, PTO, modifies terminates after finishing the execution, the modified file is left on the disk just as though it had been chan ged by , combination of seek and write system calls.
- What actually happens is that the system's internal tables are changed to make the file become the backing store for the memory region 1024 K to 1152 K.
- Thus a read from **1024 K** causes a page fault bringing in page Oof the file.
- to the system that supports segmentatron. file mapping works better. In such a system, each file can be mapped onto its own segment so that byte **n** in the fife 8 also byte n in the segment.
- Suppose that this process copies file **fllei** to fife le2. First it maps the source file, say **fllel onto** a segn ent. Then it creates an empty segment and maps it ontoife destination file.
- At this point the process can copy the source segment into the destination segment using an ordinary copy loop. No read or write system calls are needed.
- When it is all done, it can execute the unmap system call to remove the files from the address space and then exit. The output file, **file2** will now exist, as though it had been created in the conservative way.

☞ **Advantage**

I/O is not needed programming makes easier.

☞ **Disadvantages**

- It is difficult for the system to know the exact length of the output file.
- Ltcan happen that, if file is mapped in by one process and opened for usual reading by another and if the fiN process updaie a page, that change will not be imitate in the file on disk until the page is expelled.
- The system has to take great care to make sure thet*° processes do not see inconsistent versions of the file-
- If file is bigger than a segment, or even bigger than he entire virtual address space then alternative is arrant the map system call to be able to map a portion of a file, instead of the entire file.
- Although this works, it is clearly less acceptable than mapping the entire file.

4.15 jj nofy Mapped I/O

Q. Write note on rnern °<y-mapped I/O,

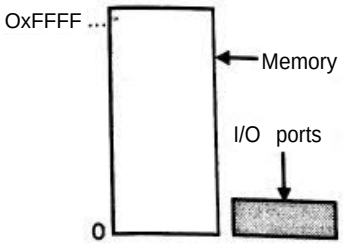
- Each controller communicates with the CPU through having few registers with them.

The operating system may instruct the devi- ,o send data, accept data, switch itself on or off, or otherwise carry out some action by writing these registers.

The operating system can know the device's reading U* ref avaihble with device control This state can be: whether device is ready to accept new command, and so on.

Besides the control registers, many devices are with data buff" 0131 the operating system can read and write-

Two address

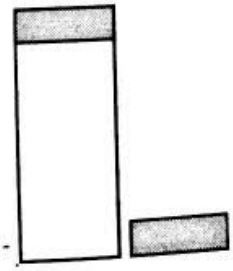


(a)

One address space

(b)

Two address spaces



(c)

Fig. <15.1 : (a) Separate I/O and memory space (b) Memory-mapped I/O (c) Hybrid

As an example, computers display pixels on the screen which is data this is possible because of video raMt buffer that offer facility to lbc prOgfa ms or the Operating system for writing in it-

- The c o nCem *S n o W to CPU interacts with the Twn o pttons are available to achieve the same. and also with the device data buffers,
- In the . lrSt toethod, I/O port number is allotted to each ion ITOl re «is,M *hich is 8- or 16-bit integer.
- P" 1 space is formed by collection of all VO ports which is protected so that other than OS ordinary user programs cannot access it.
- The second approach, introduced with the PDP-11, is to map to control registers into the memory space, as snown in Fig. 4.15.1(b).
- Each control register is assigned a unique memory address to which no memory is assigned.
- This system is called **memory-mapped I/O.** In most systems, the assigned addresses are at or near the top of the address space.
- A hybrid scheme, with memory-mapped VO data buffers and separate I/O ports for the control registers, is shown in Fig, 4.15.1(c).

Syllabus Topic : Allocating Kernel Memory

4.16 Allocating Kernel Memory

Following are the two approaches for managing free memory that is assigned to kernel processes : the "buddy system." and slab allocation.

4.16.1 Buddy System

F Q. Explain Buddy system.

- The kernel produces and wipe outs small tables and buffets repetitively during the course of execution, each of which requires dynamic memory allocation.
- In SVR4, there is dynamic allocation of several objects bv kernel as per requirement; for example these objects X proc structures, v-nodes, and file descriptor blocks.
- Several of these blocks are having considerably small siK with compare to usual machine page size. As a Xit. for dynamic kernel meraory aUocauon , the paging system seems non efficient.

7 qVR4 a revised buddy system is used. Kernel rXroory management requires the fast precess of allocation and free operations.

•nre disadvantage of the buddy system ts the tune ded lo fragment and combine blocks.

- Barkley and Lee at AT&T proposed a variation known as a lazy buddy system [BARK89], and this is the technique accepted for SVR4.
- The authors observed that the amount in time.
- blocks of a particular size changes slow y
- of size 2 is released and is
- with its buddy into a block of
- size 2^{i-1}, the kernel may next request a block of size splitting the larger block again
- To avoid this unnecessary combining and splitting, the laay buddy system postpones combining unul rt seems Zly that it is needed, and then combines as many blocks aspossible.
- The lazy buddy system uses the following parameters :
  - Ni - Current number of blocks of size 2^i
  - **Ai** - Cunent number of blocks of size 2 that are allocated (occupied).
  - **Gi -** These are number of blocks that are currently globally free having size 2 ; these blocks are qualified for combining; if the companion of such a block turn out to be globally free, then the two blocks will be combined into a globally free block of size 2 ' . If standard buddy system contains free blocks (holes) then all these could be though asglobally free.
  - **U -** These are number of blocks that are currently locally free having size 2 ; these blocks are not qualified for combining. Although the companion of such a block becomes free, the two blocks are not combined. Instead, the locally free blocks are kept awaiting future demand for a block of that size.
- The following relationship holds :

$$Ni = Ai + Gi + Li$$

- The lazy buddy system attempts to keep a group of locally free blocks and only calls coalescing if tfe number of iocrdly free blocks goes above a tLshold value. If more number of locally free blocks i available, then it may happen that, at next \ suffemnt free block will not be available to fulfill the demand.
- Usually, coalescing does not occur after block, which leads to minimum accounting and operational costs. Locally and globally free are differentiated so

acco in '3 rf locally fee blocks of particular si* does not exceed the number of allocated blocks of the same size (Li<=Ai)*

- This is a practical rule for limiting the expand locally fo* blocks - ex P crim S in ®ARK89i shows that this method results in remarkable savings,

$$Di = Ai-Li = Ni-2Li-G$$

4.16.1(A) Operation of Buddy Algorithm

ain working operation of budtfr algoritta l.

- The buddy algorithm manages chunk of memory follows. Initially memory consists of a sia contiguous piece, 32 pages as shown in Fig. 4.164.
- Any request for memory is first rounded up to a po of two, say 8 pages. The full memory chunk is divided in half.
- Since each of these pieces is still too large, the lower piece is divided in half again. Now suppose that a second request comes in for 4 pages. The smallest available chunk is split and half of it is claimed.

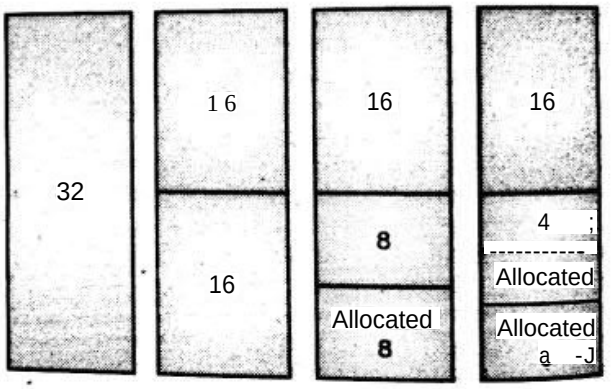


Fig. **4.16.1: Operation of the buddy algorithm**

4.16,2 Slab Allocation

Q. What is slab allocation? Explain:

- Usually main memory page frames are allocated to user-space processes, dynamically allocated W* data, static kernel code, and the page cache.
- The Linux kernel memory manages physical memory page frames. Its main function is to deal ,ocate frames for particular uses.
- manae Cat on mct hod used for user virtual tpe® XT ba « mel memory allocation

in the virtual memory scheme, a buddy algorithm is memory for the kernel can be allocated and jjsd in units of one or more pages,

The minimum amount of memory that can be allocated way is one page.

The kc mei needs small short-term memory chunks in si# 5 501 P a £ e a 0cat0r alone would be jpefficient In order to hold such small chunks, Linux uses a scheme known as *slab allocation* within an allocated page. -

Op a p c ntium/x86 machine, the page size is 4 Kbytes, and chunks within a page may be allocated of sizes 32, 64, 128.252,508, 2,040, and 4,080 bytes. '

jjnux also maintains a set of linked lists, one for each size o f chunk. Chunks may be split and combined in a yay like the buddy algorithm and moved between lists accordingly.

- The memory is allocated to kerne) data structures using dab, A slab comprises one or more physically contiguous pages. A cache contains one or more slabs.
- A cache is assigned to each unique kernel data structure. For example, cache for process descriptors, file objects, and a cache for semaphores, and so on.
- The objects that are instance of the kernel data structure are occupied in the cache which represents this data structure.
- This means, the cache representing process descriptors stores instances of process descriptor objects.
- The relationship among slabs, caches, and objects is shown in Fig. 4.16.2.

h the following Fig. 4. 16.2 two kernel objects having 3 KB size and three objects having 7 KB size is shown. These objects are stored in the respective caches for KB and 7 KB objects.

h slab-allocation algorithm caches are used to store kernel objects. A number of objects are assigned to the cache after the cache is created.

7116 of the associated slab decides the number o f o Kcts in the cache.

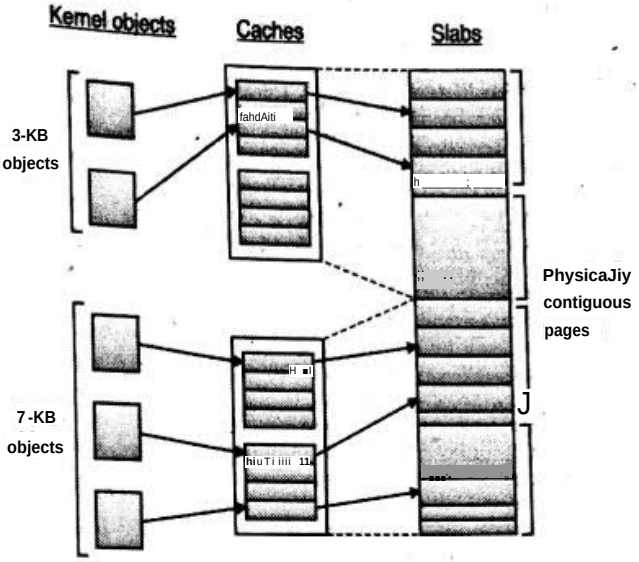


Fig. 4,16.2 : Slab allocator in Linux

- As an example, a 16-KB slab (made up of four contiguous 4 KB pages) could store eight 2-KB objects. At the start, all the objects in the cache are marked as free.
- At the time of requirement of new object for a kernel data structure, the allocator can allot any free object from the cache to fulfill the request Then this assigned object from the cache is marked as used.

In Linux at any given point of time, a slab may be in one of three possible states :

1. **Full** : All objects in the slab are marked as used.
2. **Empty** : All objects in the slab are marked as free.
3. **Partial** : The slab consists of both used and free objects.

Syllabus Topic : Other Considerations

4j 7 other Considerations

4.17.1 Prepaging

Demand paging leads to many number of page faults whTprooess starts execution. This is due to attempt to obtain the initial locality into memory.

This can occur at other times also. For example, insider restarting of swapped-out process having ah its pages on the disk, and requires each page to be f ched by its own page fault.

p Ug is nothing but trying to stop this high level of initial paging-

- The policy is for bringing all needed pages into memory at one time. Some operating system', example, Solaris prepage the page frames for small files.
- Working set model permits to keep each process a list of the pages in its working set. If it is required to suspend a process, because of an 1/0 wait or a lack of free frames, the working set for that process is known.
- For the resumption of such process due to I/O has finishM or sufficient free frames have become available, it is possible to bring back into memory its whole working set prior to its restarting.
- Prepaging may offer an benefit in some cases. The question is simply whether the cost of using prepaging is less than the cost of servicing the corresponding page faults.

4.17.2 Page Size

- Page size is important factor to decide the performance. If page size is small then any program will be divided in more number of pages leading to have a large page table.
- On some machine, it is required to load the page table in hardware register every time when context occurs.
- Hence more time will be required to load the larger size page table due to small size of the pages. More number of transfers to and from disk will be required due to more number of page faults.
- Hence more time wilt be elapsed in seek and rotational delay.
- If page size is large then last page will remain somewhat empty leading to internal fragmentation.
- Due to large size of pages, most of the unused part of program can remain in memory.

4.17.3 TLB Reach

- The hit ratio depends on the number of entries in the TLB, and the hit ratio can be increased by having more number of entries in the TLB
- This, on the other hand, is not cmt ■ associative memory is both . Ktlve , " 'he consuming. eXpens,ve "d power
- Associative memory is used to build the TLB. Hit ratio is a metric which is similar to the Ti r refers to the total memory accessible from the" h which TLB and is just the number of entries multiplied by th ' "«= Page size.
- TLB stores the working set of process. Otfte rwise the process will require more time resolving mem- references in the page table rather than the TLB.
- If number of entries in the TLB is increased the, - reach also increases.
- Other scheme to increase the TLB reach is cit c -r increase the size of the page or offer multiple page sizes.
- It may lead to fragmentation as many applica tions do not need such large page size. Operating system should manage TLB for multiple page sizes instead of hardware manage it.

4.17.4 Inverted Page Tables

- The inverted page table is already discussed. The reason behind this type of page management is to lessen the amount of physical memory required touacfc virtual -to-physical address translations.
- This saving is achieved by creating a table that hasone entry per page of physical memory, indexed by thepair <process-id, page-number>.
- As inverted page table maintains the information regarding which virtual memory page is stored in each physical frame, they lessen the amount of physical memory required to store this information.
- On the other hand, the inverted page table doesnot includes whole information regarding the logical address space of a process, and that information is necessary if a referenced page is not presently in memory.
- In order to process page faults, demand paging need5 this information. An external page table should l* maintained per process for availability of information
- This each page table is similar to the conventional p* P ess page table and includes information on whri* each Virtual page is located.
- As an external page tables are referenced only 0(1 av ■ f nCe available quickly. P a ®e fault, they do not require to t*
- As of themselves paged in and out
- me n mo TniX arc
- But, due to page fault the has
- to P e another page fault as it" 7n ex ernal

nage table it needs to search the virtual Da „ „ ' ule firing store. In kernel this situation should be handled carefully.

4.17.5 program Structure

[f user or compiler is aware of underlying demand paging then it is possible to improve systenj performance.

If data structures and programming structures carefully selected, then it can increase locality and hence decrease the page-fault rate and the number of pages jn the working set.

418 Exam Pack (University and Review Questions)

r Syllabus Topic : Memory Management Strategies Background

Q. What are different requirements for memory management? Explain. *(Refer section 4.1)*

0. Write short note on Relocation. *(Refer section 4.1.5)*

Q, With the help of example, clearly differentiate between logical and physical address space. *(Refer section 4.1.6)*

Syllabus Topic : Contiguous Memory Allocation

Q. Explain memory management with Fixed and Variable Partitions. *(Refer section 4.2.1)*

0, What is internal fragmentation and external fragmentation? Explain with example. *(Refer section 4.2. 1)*

Q. Compare internal and external fragmentation *(Refer section 4.2.1)*

Q. Whatis compaction? Explain with example *(Refer section 4.2.3)*

Q. Discusspartition selection algorithm in brief. *(Refer section 4.2.4)* **(4 Maries)**

Q. Ex Ptain different memory allocation stra (Ref er section 4.2.4) **(May2°16)**

Example 1 **(5 Marks)**

Syllabus Topic : Paging

What i» **paging?** *(Refer section 4.3)* (5 Marks) **(June 2015)**

Q. What is Page Table? Explain the conversion of Vidual Address to Physical Address in Paging with example. *(Refe rer sectJOn 4 3 .1)*

Q. Explain effect of page size on performance. *(Refer section 4.3.4)* **(5 Marks)** **(Jun® 2015, Nov. 2015, Dec. 2016)**

Q. Explain the hardware support for paging. *(Refer section 4.3.5)* (10 Marks) **(Dec. 2016)**

Syllabus Topic : Structure of the Page Table

Q. Discuss various techniques for structuring page tables. *(Refer section 4.4)* (10 Marks) **(Dec, 2014)**

Syllabus Topic : Segmentation

Q. What is segmentation? Explain it with example. *(Refer section 4.5)*

Q. Explain the difference between paging and Segmentation. *(Refer section 4.5.1) {5* Marks) **(Dec. 2016)**

Q. Explain segmentation with paging. *(Refer section 4.6)*

Syllabus Topic : Virtual Memory Management

Q. Write short note on Virtual Memory. *(Refer section 47.1)* (5 Marks) **(May 2016)**

Syllabus Topic : Demand Paging

Q. Write short note on Demand Paging. *(Refer section 4.7.2)* (5 Marks) **(Dec. 2014, Dec. 2016)**

a. What is demand paging? *(Refer section 4.7.2)*

a. List advantages of demand paging. *(Refer section 4.7.2)*

Q. List disadvantage of demand paging. *(Refer section 4. 7.2)*

Syllabus Topic : Page Replacement

Q. write short note on various page replacement policies. *(Refer section 4.9)* (5 Marks) **(Dec. 2014)**

a. E ain the various page replacement strategies. *(Pefer section 4.9)*

Q. Write short note on Belady-s anomaly. *(Peter secSon 4.9)* (5 Marks) **(May 2016)**

Q. Compare Optimal, LRU and FIFO page replacement algorithms with Illustration, *(Refer section 4.9.1)*

Q. Compare FIFO and LRU page replacement algorithm. *(Refer section 4.9.1)*

Q. Compare Optimal, LRU and FIFO page replacement algorithms with Illustration. *(Refer section 4.9.2)*

Q. Compare Optimal, LRU and FIFO page replacement algorithms with Illustration. (Refer *section 4.9.3)*

a. Compare FIFO and LRU page replacement algorithm. *(Refer section 4.9.3)*

Q. Explain LRU approximation approach, *(Refer section 4.9.4)*

Example 4.10.4 (10 Marks) **(Nov. 2014)**

Example 4.10,5 (10 Marks) **(May 2016)**

Example 4.10.6 (10 Marks) **(June 2015)**

Example <10.7' (10 Marks) **(Dec. 2016)**

·5 £!l<W .8 (10 Marks) **(Dec. 2015)**

Example 4.10.9 (10 Marks) **(Dec 2016)**

Syllabus Topic : Allocation of Frames

Q. Explain different strategies for allocation of frames. *(Refer section 4.11)*

Syllabus Topic : Thrashing

Q. What is thrashing ? *(Refer section 4.13)* (5 Marks)

Syllabus Topic : Memory-Mapped Fii 1* 2015)

Q. Explain memory-mapped files. *(Refer section 4.15)*

Q. Write note on memory- mapped I/O *(Refer section 4.16)*

Syllabus Topic : Allocating Kernel Memory

Q. Explain Buddy system. *(Refer section 4.17)*

Q. Explain working operation of buddy algorithm. *(Refer section 4.17.1(A))*

Q. " ha,iss,ab "location? Explain. *(Refer section 4.17.2)*

chapter 5

Module V

Storage Management

Syllabus Topics

File System : File Concept, Access Methods, Directory and Disk Structure, File-System Mounting, File XCS S,=,.„ Fhs „, em implementation. Allocation Methods, Free- Spac • s= J ss - Implementation, Directory NFS; y and Performance, Recovery, Secondary Storage Structure : Overview of Mn e* Disk Scheduling, Disk Management, RAID Struck orage Structure, Disk tnjcture , Disk Attachment, Structure, Swap-Space Management ure, St3 ble-Storage ieipiementation. Tertiary-Storage I/O Systems : Overview I/O Hardware, Application I/O Interface, Kernel I/O Subsystem, Transforming I/O Requests to Hardware Operations, STREAMS, Performance.

Syllabus Topic : File System - File Concept

5.1 File System

5.1.1 File Concept

- Every application needs to access and store the information. During execution, process can keeps the information in its address space. This address space to store information is sufficient for some type of applications but not for others, which requires huge size of information.
- After finishing the execution, process terminates and information also vanishes. For many applications, it is required that information should be retained forever.
- Even though computer crashes or process is killed, still information should remain retained. Most of the time, the information is sharable among many processes.
- If information kept in the address space of one process, other processes will not be able to access it. It is wsessary to keep information independent of the Processes.

lowing the necessities for long tenn information storage.

- 0 Jt must be possible to store a very larg c amount of mfortnafion.
- o The information should not loss after termination of the process using it.
- o Several processes must be able to access the information simultaneously.
- o In order to fulfill the above requirements, it is necessary to store information on disks and other secondary storage in units called **files.**
- o A file is a named collection of related information that is recorded on secondary storage. The data cannot be written directly to secondary storage unless they are within a file. Processes can then read information from the file.
- o It also can write new information into the file if needed. After process termination, information in file should remain retained and should not vanish. A file should only vanish when its holder clearly removes it. Operating system manages the files.
- o **Fite'system** describes bow files are structured, named, accessed, . used, protected and implemented.

5.1.2 Fite Attributes

- File is identified by its name which is string of the characters. Every file has a name and its data. All operating systems associate other information with

each file, for example, the date and time of file creation and the size of the file.

- Such extra information associated by operating system is called as attributes. The lists of attributes are not same for all the systems and vary from one operating system to another.
- No single existing system supports al! of these attributes, but each one is present in some system.

Some of the attributes and its meaning of file

Q. Explain various file attributes in brief.

Attributes	Meaning
Protection	Access-control information determines who can do reading, writing, executing, and so on.
Size	The current size of the file
Type	Information is needed for systems that support different types of files.
Creator	ID of the person who created the file
Owner ■	Current owner of the file
Password	Password needed to access the file
Identifier	This unique number which identifies the file within the file system.
Hidden flag	0 for normal; 1 for do not display in listings
Read-only flag	0 for rcad/write; 1 for read only
ASCII/binary flag	0 for ASCH file; 1 for binary file
Lock flags	0 for unlocked; nonzero for locked
Key length	Number of bytes in the key field
Time of last access	Date and time the file was lasl accessed
Time of last change	Date and time the file has last changed
Maximum size	dumber of bytes the file may grow to
System flag	() for normal files; 1 for system file

5.1.3 File Operations

Q \ Explain various file operations in brief.

Files are used to store information which can be used later on. For storage and retrieval purpose, different types of systems offer different operations. For the operation ike create, write, read, reposition, operating system offer the system calls. The majority of common system calls re ing to files are listed in Fig. C5.L

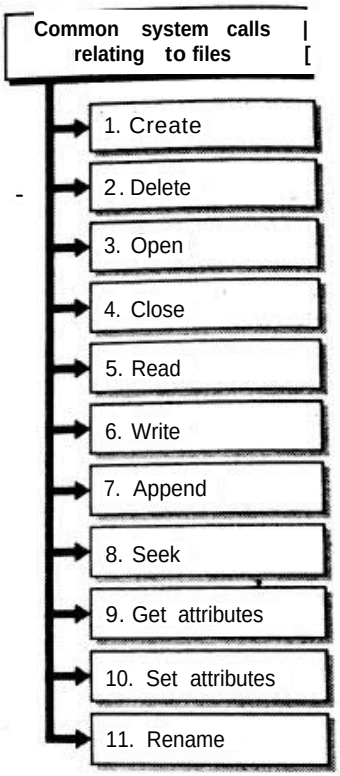


Fig. C5.1 : Common system calls relating to files

1. Create

The new file is created without data. It is needed to locate the space for this created file in file system. Tl* intention of the call is to declare that the file is created and some of the attributes needs to be set.

2. Delete

File need to be deleted to release the disk spac[e] occupied by tile when it is no longer required.

3* Open

Before making use of file, it is required to open the file first by any process. To have quick accesses in future all the attributes and disk addresses are bought ifi*° main memory.

4. Close

After use of the file completes, all accesses ets finished. The fetched attributes and disk address in memory are no longer required, so the file should be

closed to release the internal table space. The disk is vided in blocks and is written' J blocks. Jl'ng a fife[c0] "'P[el] writi "E of the file's last block.

. Head

The system call that is used for reading a file specifies tie name of the file. Generally, the bytes are fetched fin® the current posmon. The system should maintain read pointer at the address in the file from where next read is 10 begin. The caller must state the amount of jaU desired and must also offer a buffer to put them in.

x. **Write**

System '— -- ---------- - -v mi. spcciry rife name and information to be written to the file. Writing is again at the current position. The existing file size will increase if writing starts from end of the fife. In this case, the current position is at the end of file. On the contrary, if writing is done at somewhere in the middle of the file, the previously existed data will be overwritten and vanished permanently.

4 7. **Append**

With the help of append system call; data gets appended only to the end of the fife. Systems offering less number of system calls do not generally provide append,

*** I Seek**

The seek system call is used to place the pointer at a particular location in the file. It is needed for random access files to specify from where to obtain the data. After this call has completed, data can be read from, or Witten to, that position.

9. Get attributes

processes frequently need to read file attributes to [co]nipfefe theij- designated work. Get attributes system call is used for this purpose.

[101] Set attributes

User can change the some attributes of file after its <1[crca] h°O-This system call makes that possible.

[111] Rename

User needs to change the name of an existing tde. This s[Stern] call makes that possible.

5.1.4 File Naming

Pro[cess] &ves the name to the file when it creates it C* [co]«ipletio[n] of the execution, even though pw[ess] [ter]minates, file still exist and other processes can [acce]ss [Slha]'filebythe same name.

- The rules for flu[nammg] are not same for all systems, nnpftif'[S]r[Oni]s[stem] to system, but all current as [l]egal file [nan]«S. If an operating system the [ln,gU1SheS] the [of] a file, it can then operate on the file m logical [Ways]
- D[.g]s and special characters are also allowed to use w[*] giving the fife names. UNIX operating system d[ifferentiates] between upper and fewer case letters, whereas MS-DOS does not.
- Windows 95, Windows 98, Windows NT and Windows 2000 support the MS-DOS. fife system and thus inherit its properties.
- Fife name is divided into two parts - name and extension which are separated by period character. User and operating system can recognize the type of file from name only.
- MS-DOS allows the fife names containing 1 to 8 characters along with an optional extension of 1 to 3 characters. On the contrary, in UNIX user has a choice of deciding the size of extension. In UNIX, file can have two or more extensions.

Examples of file extension, Its meaning and type

- Table 5.1.1 shows the some examples of file extension, its meaning, and type.

Table 5.1.1

Extension	Meaning	Type
gif	CompuServe Graphical Interchange Format image	Image
html	World Wide Web Hypertext Markup Language document	Web page
obj, o	Complied machine language not linked	Object
exe, bin, com	Ready to execute machine language program	Executable
txt,doc	Text, document, data	Text
lib. a, dfi	Libraries of routines for programmers	Library

Extension	Meaning	Type
arc, rip, tar	Grouped files together sometime compressed for archiving or storage	Archive
tnp3,avi, mpg	Binary files containing audio or audio/vidco information	Multimedia
Pdfips	Its content cannot be changed	Portable document format and postscript

5.1.5 File Types

Q. Write short note on File Types.

Several operating systems support many types of files. Following are the different types of files.

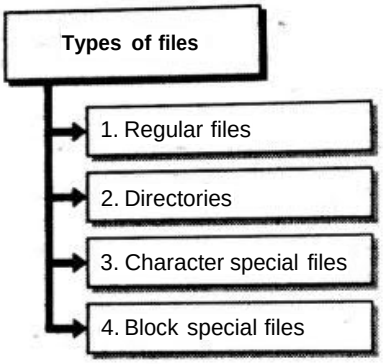


Fig. C5.2 : Different types of files

1. Regular files

Regular files contain user information. The byte sequence, record sequence and tree structured explained above are the examples of regular files.

2. Directories

These are system files for maintaining the structure of the file system.

3. Character special files

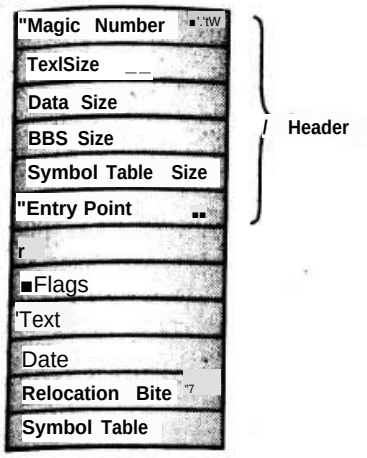
Character special files are related to input/output and used to model serial I/O devices such as terminals printers, and networks.

4. Block special files

These are used to model disks,

- All regular files are normally either A scn fit binary files. ASCII files contain lines of text Each is terminated by either carriage return character o 7 feed character- MS-DOS uses both types of files. Lines can be of different length.
- The great benefit of ASCII files is that they can displayed and printed as they are. Any text editor edit ASCII file- Furthermore, tf large nua bers programs use ASCH files for input and output easy to connect the output of one program to the of another, as in shell pipelines. Binary files are not ASCH files.
- Listing them on the printer gives listing whitk is beyond the understanding. It gives the random garbage if it listed on the printer. Usually, they have some internal structure known to programs that use them.
- UNIX binary file is an archive. It consists of, collection of library procedures compiled but w linked. The usefulness of file types can be understood from the TOPS-20 operating system.
- The recompilation of source file will be carried out automatically after its modification. This recompilation is done if user wants to execute the object fikli guarantees that the user all the time runs an up-todgt object file. It saves the waste of time in executing te 1 old object file.
- To achieve this automatic recompilation of modifd source file, the operating system must have tit capability to make a distinction between sourced and object file. Also OS should be able to check creation time and time at which file was las*h modified. In order to choose the right compiler, OS must also have a ability to determine the language of the source program.
- Fig- 5.1,1 demonstrate the UNIX version of execute binary file. Although this file is just a sequence bytes, without a proper format operating system can execute this file. The file is divided into five seg ments. These are header, text, data, relocation bi& and symbol table.
- The very first field in header is magic numbc r, It recognizes the type of file as an executable K facilitates to .prevent the unintentional exec* 00 ot file Whlch " not in this format. After magic nun** the sizes of the different sections of the 6,6 ** *2 exeenti"on ViU ous stmts, soflle action sizes, the address a1 * present flag bits are
- Program 8 the ltsel f. header "e the test and data <f mcflK and

relocated using the relocation bits. The symbol table is used for debugging.



Fig, 5.1.1 : An executable file

16 File Structure

Explain the different techniques to structure the files.

Following are the three ways in which flies can be rtructured.

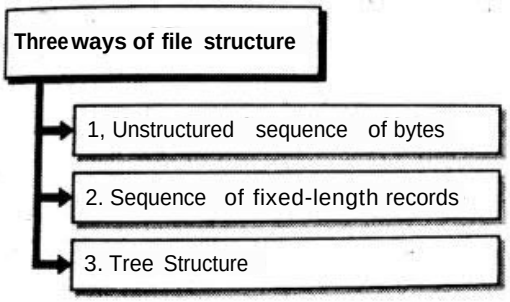


Fig. C53 : Three ways of file structure

t Unstructured sequence of bytes

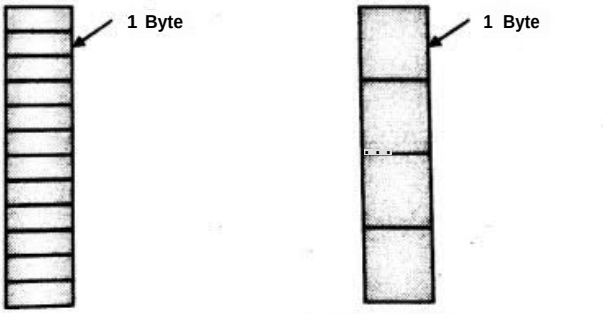
- In this structuring, operating system treats content of the file as sequence of bytes. Any meaning must be imposed by user-level programs.
- M UNIX and Windows use this approach. This structuring provides more flexibility. Any types of content can be kept in the file and as per convenience of the user, name can be given to the file.

n«nce of fixed-length records

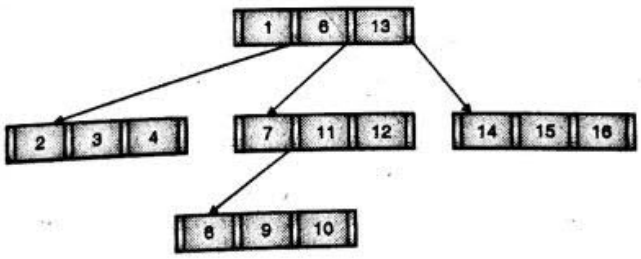
- In this approach file is treated as sequence of fixed M records. Each record has its internal structure. Any read operation on file returns one record and write operation will append or overwrite one record.
- When punched cards were in use. record size was of 80 characters and of 132 characters propose o r line Printers. Many operating systems based their file systems on files consisting of 80-character records when punched cards were in use.
- Programs read input from file in the unit of 80 characters and wrote in units of 132 characters keeping remaining 52 characters blank.

3. Tree structure

- In this organization, a file consists of the records which re organized in tree structure. Ail the records not necessarily have same length. Each record has a **key** field in a fixed position.
- rhe records in the tree are arranged in the sorted order of key field, to permit quick searching for a particular key.
- The main aim in this type of organisation is to search record on particular key. The next record also can be searched. Operating system decides where to add new record. User is not concern about this operation. This type of file is broadly used on the large mainframe computers and still used in some commercial data processing.
- Figs, 5.1.2(a), (b) and (c) shows above three kinds of files. Fig. 5.1.2 (c) shows the tree of records of student file arranged in sorted order of roll numbers.



(a) Byte Sequence **(b) Record Sequence**



(c)Tree

Fig. 5.1J

Disadvantage of supporting multiple «'• structures

- If multiple file structures are supported by the operating system Ita it would be huge in si „. For suppo "le sU uctures, the corres SlngVsS- present sys
- Ad Zonally, it is required to define every file i operating system supjre rts for one of the file type. « .SonTXs (he smJcni re of X-itcan.eadto severe problems.
- Opening systems like UNB. MS-DOS enforce and support file structures which are least in numbers. mffl treats each file as a sequence of 8-b.t bytes an operating system does not cany out the understanding of these bits. As a result of this approach, maximum flexibility but little support is offered.
- In order to interpret an input file to the suitable structure, application should have included its own code. On the other hand, all operating systems must support minimum one structure of an executable file. Due to this support, the system will be capable of loading and running the programs.

Syllabus Topic : **Access Methods**

5.1.7 Access Methods

(June 15)

Q. Explain different file access methods.

MU - June 2015. 10 Marks

Following are the different access methods for files.

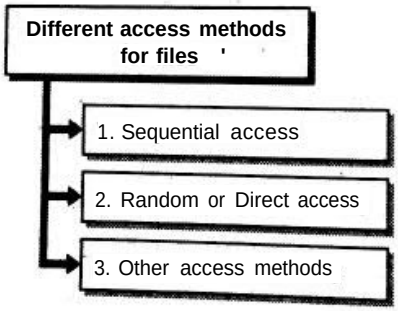


Fig. C5.4 : Different access methods for files

***4 L Sequential access**

Q. Write short note on File Access (sequential QCCGSS/e

- The simplest access method is sequential access. Early systems provided only this kind of file Opc ® this type of file access, process reads ajj the records m a We in order one record after other, starting at the beginning-
- while accessing, skipping of any record or readmg them oul of order is not possible. This access method convenient for storage medium such as mag wpe to a certain extent than disks.
- The reading of the next portion of the file is carried oat . read operation and automatically file pointet is moved forward. The appending of new information to L end of the file is carried out by write operation « me pointer advances to the end of the newly write portion.

2. Random or Direct access

Q. Write short note on File Access (Random acc 7|

- When use of disk started for storing files, it became possible to read the bytes or records of a file out of order. Il is because; disks allow random access to any file block. *
- It also became possible to access records by its key instead of by position. Files whose bytes or records can be read in any order are called **random access files.** They are required by many applications such as database systems.
- If railway customer calls up and wants to reserve a seal on a particular train, the reservation program must be able to access the record for that train directly instead of reading hundreds of records of other trains first.
- For the random access method, the file operations must be modified to include the block number as a parameter. Thus, we have read n and write n, w erc n is the relative block number. Actual absolute disk address of the block is different.
- The beginning of the relative block number is fr0 ® address 0. Then next block number is 1 and so 00 although the absolute disk address of the first blo i 14045 and second block is 319L
- The relative block numbers permits the operating system to take the decision about location of placing the file and facilitate the user to stop from accessing file system portions which are other than b* file portion.

Every read operation gives the position in the file to start reading at. In the second one, a special operation, seek, is provided to set the current position.

After a seek, the file can be read sequentially from the now-current position.

3. **Other access methods**

These access methods can be built on top of a random-access method. These methods generally involve the construction of an index for the file.

The index has pointers to the various blocks. The record in the file is searched by searching the index first and then uses the pointer to access the file directly to find the desired record.

Several factors are important in choice of file organization. These are

- Minimum access time
- Ease of update
- Economy of storage
- Simple maintenance
- Reliability

Syllabus Topic : Directory Structures

5.1.8 Directory Structures

Q. Explain different methods for defining the logical Structure of a directory.

- Directories keep track of files. Directories are itself files in many system. Systems store huge number of files on large capacity disk. In order to manage all these data, we need to organize them.
- This organization involves the use of directories.
- Following are the most common methods for defining the logical structure of a directory.

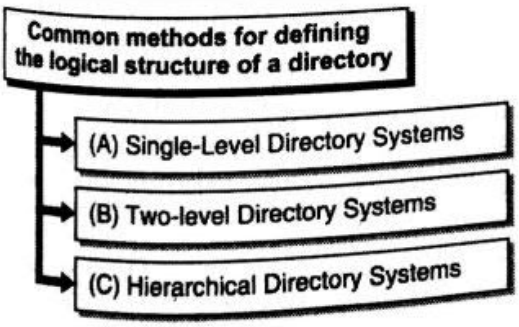


Fig. C5.5 : Common methods for defining the logical structure of a directory

5.1.8(A) Single-Level Directory Systems

Single-level directory is the simplest directory structure. In this directory structure, one directory contains all the files. This single directory is also called as a root directory. Since on early personal computers, only one user was working, this system was more general. Fig. 5.1.3 shows single-level directory system containing five files, owned by three different users P, Q, and R. User P has two files, User Q has two files and R has one file in the directory.

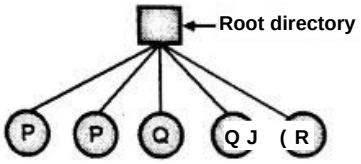


Fig. 5.13 : Single-level directory system

Advantages

- It is simple to implement.
- Locating files become faster as there is only one place to look.

Limitations

- If single user has huge number of files kept in single directory, it becomes difficult to remember the name of each file.
- If more than one user keeps the files in a single directory, then different users may give the same names to their files, violating the rule of uniqueness of names.

5.1.8(B) Two-level Directory Systems

- Two-level Directory System overcomes the limitations of Single-level directory. In this directory system, a private directory is given to each user. When a user refers to a particular file, only his own directory is searched.
- As different users directories are different, the same name given to the files do not interfere each other. There is no problem in giving the same name to the files in different directories.
- Single user directory have a compulsion of having all files unique name. While creating the file for particular user's directory the operating system makes confirmation about whether another file of that name exists in the same directory or not. To know this existence of file

or not, the OS searches the directory of the user for be created. To delete a file, the has to search only directory.

- Hence operating system cannot fortuitously delete another user's file that has the same name. Fig. 5.1.4 shows two level directory systems with separate users directories and its files.

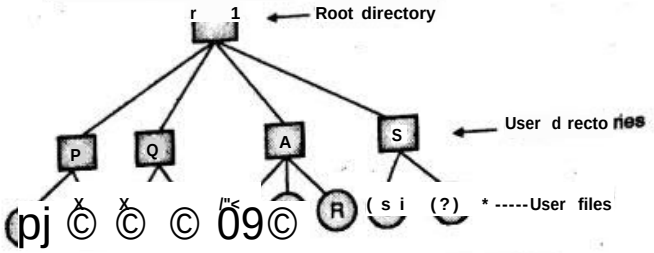


Fig. 5.1.4 : Two-level Directory Systems

- Above design multiuser computer or on a simple network of personal computers that shared a common file server over a local area network. In this system some sort of login procedure is needed.
- When a user attempts to open a file, the system knows which user it is in order to know which directory to search. Files are system programs such as loaders, assemblers, compilers, utility routines, libraries. Loader reads these files and executes when the right commands are given to the operating system.
- These commands are treated by command interpreters as the name of a file in order to load and execute. In case of two level directory systems, this file name would be searched in the current user directory.
- If system files are kept in each user directory then problem can be resolved. This solution leads to wastage of memory as each user directory contains the copied system files. If special user directory containing all system files is defined the above problem can be solved.

Advantages

- Solve name collision problem as every user has separate directory.
- Independent user gets isolated from each other.

Limitations

- If the users are co-operative, that is, working common task, then some system does not allow accessing the other user's files. In some systcm , if permitted, one user must have the ability to name a fife in another user's directory. To name a particular file uniquely in a two-level directory, we must give both the user name and the file name.
- It is not satisfactory oselS with a number of flles . Even on a singled personal computer, it is not convenient.

5.1 8(C) Hierarchical Directory Systems

- It is quite general and advantageous for users to group their files together in logical ways. A student fa example* might have a collection of files related different subjects of her curriculum.
- First collection can be the group of flles related to one subject a second collection of files related to second subject and so on. Here, some way is required to group these files together in flexible way.

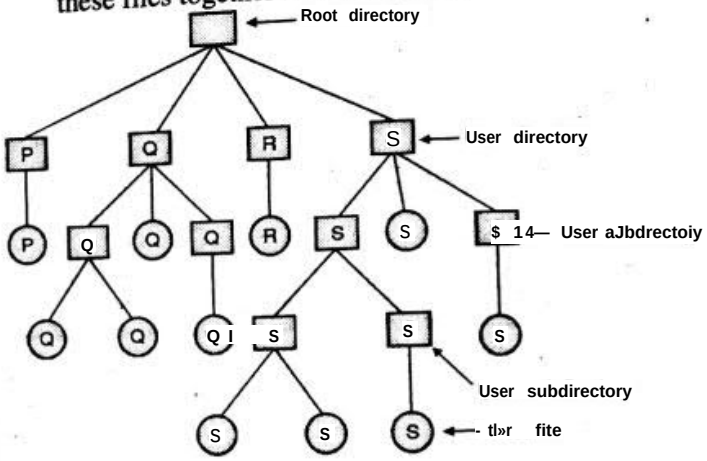


Fig. 5.1.5 : Hierarchical Directory Systems

- A tree of directories is the solution to store such group of files. A tree is the most common directory structure-
- In this approach , each user can have as naty directories as are needed so that files can be groups together in expected ways.
- The tree has a root directory, and every file 1° the system has a unique path name. The approach is shown in Fig. 5.1.5. Root directorv contains directory P, Q, R, and S which belongs to different users. Users Q $ have created the subdirectories.
- As user can create random number of subdirectories-, offers a commanding structuring tool for users to organize their work. This is the reason, nearly modem file systems are organized in this approneb-

Advantage

With a hierarchical directory system, in addih01, to their files, users can access the files of other us specifying its pathname .

* 0" fi* SyStem . o rEanize 1 " ee of dlr
sse d by specifying the path name. Paft «*»• «
* f typ esThesearesh0WDinFi 8-C5.6. can

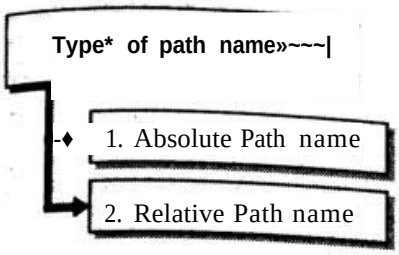


Fig. C5.6 : Types of path names

I. Absolute path name

[t consists of the path from the root directory to the file.

The meaning of the path */usr/myfolder/myfile* is that, [he root directory contains a subdirectory *user*, which in turn contains a subdirectory *myfalder*, which contains the file *my file.*

4 1 Relative path name

- ft consists of the path from the current working directory to the file.
- A user can assign one directory as the current working directory, in which case all path names not beginning at the root directory are taken relative to the working directory.
- If the current working directory is */usr/myf older*, then the file whose absolute path *usr/myfolder/myfile* can he referenced simply as *myfile.*

to UNIX operating system the elements of the path are separated by /. In Windows the separator is t The same Pat h */usr/myfolder/myfile* in UNIX, is specified in Widows as *\usr\myfolde myfile-* * thc fir chara€tcr of Path is separator then path is absolute path. If during Wor Lng, program always needs a particular ft e, it should use absolute path to access that file froin any CUntnt working directory.

5.1.10 directory Operations

Q/ 2'
VWdifferenl directory operatio

To ""wage the directories, different system calls demonstr to system with respect a*C n ore to Wm dissimilarity from syste .. calls for files.

directn * fo Ve 311 idea of w °ddng of system calls for Consider the following examples from UNIX.

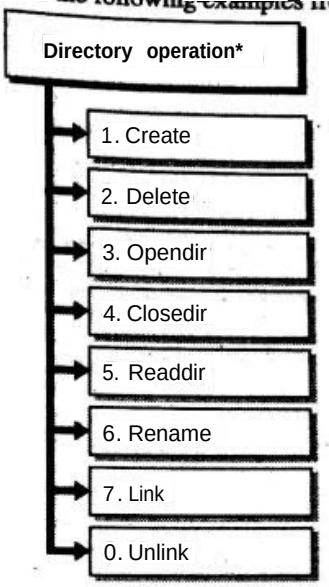


Fig. C5.7 : Directory operations

"fr 1. Create

Empty directory is created excluding dot and dotdot (. and ..), which are placed there automatically, by the operating system,

"4 2. Delete

If only single dot and double dot (. and is present in directory then it is treated as empty, single dot and double dot within directory cannot usually be deleted.

-> 3. Opendir

Directory can be read to list all the files from it Directory should be opened before reading just opening and reading the file,

-> 4. Closedir

After reading completes, a directory should be closed to free up inner table space.

5. Readdir

It returns the next entry in an open directory, **readdir** always returns one entry in a standard format irrespective of possible directory structures is being used.

6. Rename

Directories can be renamed just like files. Rename renames the directory.

<+ 7. Link

Due to linking, a file appears in more than one directory. Link system call specify an existing file and

its path name, and creates a link from the existing file the to the name specified by the path. In this fashion t same file may come into view in several directories.

8. Unlink

Unlink removes the directory entry. If the file being unlinked only exist in one directory then it ts deleted from file system.

Syllabus Topic : File Syetem Mounting

5.1.11 File System Mounting

As discussed earlier, a ms* can be divided in multiple partitions and each partition can have its own file system or there can be multiple disks connected to system. So t ere can be multiple file systems. It is possible to copy a file from one file system another. Consider the disk-1 and disk-2 with file systems as shown in ri g. 5.1.6 circles shows files and rectangles shows directories.

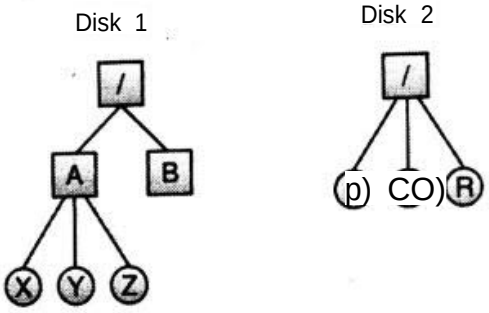


Fig. 5.1.6 : Two file systems on two disks

Suppose we have to copy file R from disk-2 to a directory A under disk-1. After mounting directory structure becomes as shown in Fig. 5.1.7.

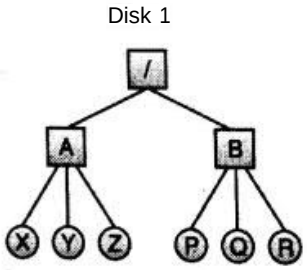


Fig. 5.1.7 : Directory structure after mounting

Now the command *cp/B/R/A/R* copies the resultant file R. In UNIX a facility is given to mount or unmount a file at prespecified time. The mount and unmount system calls are used for this purpose. A system administrator does the mounting at the time of booting of UNIX. A large number of file system are supported by UNIX for mounting.

, implementation of mounVunmourrt

File systems can be on same or different devices. With the mount . rhrsc file system can be viewed as single Fil &ie system Files can be moved between different director of different file systems thereby appearing as a singfe * system. The mounting can be realized in UNIX through mount table. Along with other information, the mount table contains following information.

- I-node of root node of the file system to be mounted.
- I-node of the directory in which mounting is to be carried out. This directory is called as mount point.

The mount table allows the kernel to traverse from one file system to other. The user remains unaware about it under the feeling of single file system.

implementation of link/unlink

If we need to access file P12 from P1 and also from R1 the Link needs to be created from R1 to P12. To this shared file now two absolute pathnames exist to this shared file as shown in Fig. 5.1.8. These are /P/P1/P12 and /R/RUP12 In directory R1 the same file can be called by different name say R1. In this case, pathnames /P/P1/P12 and /R/R1/R1 would be the same.

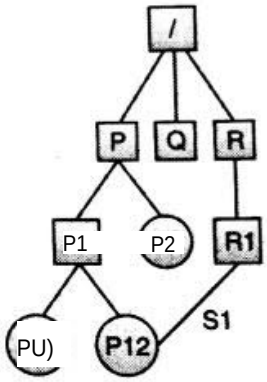


Fig. 5.1.8 : Linking

Consider the file to be linked and therefore shared is S1 in directory RL The same file is P12 in P1 as show"* Fig. 5.1.8, The linking is done by kernel with link system call as follows.

- Except super user, other users will be not permitted to execute link system call. Error messages w® be outputted if user is not super user.
- The pa* name /P/P1/P12 is parsed to obtain the i-node of the file to be linked. Store it for future use.
- i-node record is accessed and number of lb* incremented.

in order to have disk consistency, the i-node is written back to disk.

The path name /R/Rl is parsed to obtain the i-node for the directory. The content of the directory is then read and free entry in directoty is located and it makes sure that file with name SI not present in directory. If present error message is outputted and exit operation is performed.

The entry of the Jinked file is done in /R/R(with S1 as fife name and i-node number stored in step 2.

a jj directories are then written back.

5.11.2 Working of Fifes

While working on a common task, many users need to share their files. If these shared file appears simultaneously in different directories belonging to different users, then it will be convenient to all users.

Fig. 5.1.9 shows that one file of user S appears in the directory of user Q. Because of the sharing, file system is now Directed Acyclic Graph (DAG) instead of tree.

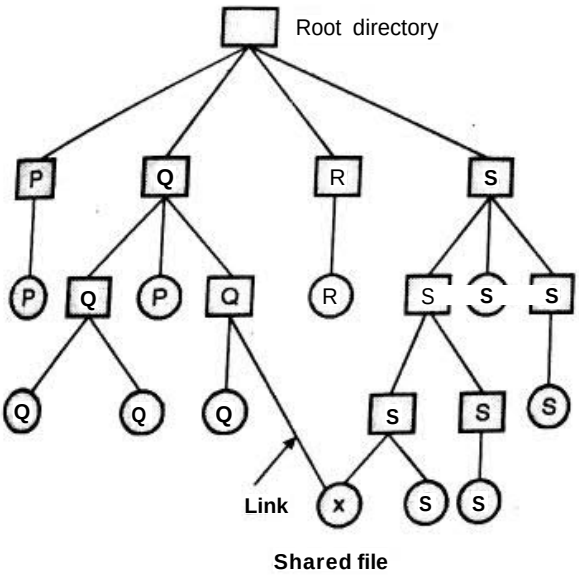


Fig. 5.1.9 : File system combining a shared file

Problem In sharing the files

Suppose directories actually do contain disk addresses then, in above example, copy disk addresses of S s directory should be made available in Q's directory When the file is linked.

The user who actually perform append operation, the appended block appears only in his directory* In above example, If either Q or S perform append operation to the file, then the new blocks will be listed only in the directory of the user, who is actually doing the append.

The changes will not be visible to the ocher user, thus overwhelming the intention of sharing. Following are two methods to handle this problem.

- Instead of listing the disk blocks in directories, it is listed in data structure related to the file, just like i-node in UNIX. The directories would then point just to this data structure.
- When Q link to S's file, system create a new file of type LINK and enters this file in Q's directory. The new file contains the path name of the file to which linked. In our example, it is S. When Q read from linked file, the operating system note that it is type LINK file and search the name of the file and read it This is called symbolic link approach.

The disadvantage of first method is that, when Q links to the shared file of user S, i-node records owner of file as S and link count becomes 2 which was initially 1. Now system knows 2 directory entries pointing to the file. If S removes the file and clears the i-node, then Q's directory entry will point to ihvalid i-node. If later on i-node is assigned to other file then Q will point to the wrong file.

Due to link count in i-node, operating system will come to know that file is in use but unable to find all the directory entries of that file. The solution to this problem is to remove S'j directory entry, but do not clear the i-node and keep it with count set to 1.

Now Q is the only user having a directory entry for a file owned by *S*. Here system will assume that file belongs to S until Q decides to remove it. When file is deleted, count goes to 0 and no longer belongs to any user.

The disadvantage of symbolic link is that, extra operating cost required. The LINK file having the path must be read, and then parsing of path is done and followed, component by component, until the i-node is searched.

All of this activity may require a considerable number of additional disk accesses. Every symbolic link requires additional i-node, as is an additional disk block to store the path.

Although path name is short, the system could store it in the i-node itself for optimization purpose.

Sullabun Topic: Sharing

5.1.13 File Sharing

q (Write nota on 'access rights'

It is necessary to share a file among multiple of users in multiuser system. It needs to deal with two issues: that are access rights and the management of simultaneous access.

5.1.13(A) Multiple Users

Q. Explain issues related to file sharing.
Q. Explain different access rights to the file.

The file system shou. J ffer a flexible facility for permitting widespread file s * among multiple users . It offcr a many is also a responsibility " ' he way of file accessing. In alternatives in order to co (Kss permissions general, it important to grant partly access a particular file. A broad OT n£W S l0 T L P Xess rights has been used. The following are the some of the examples of access rights that can be granted to a particular user for a particular file.

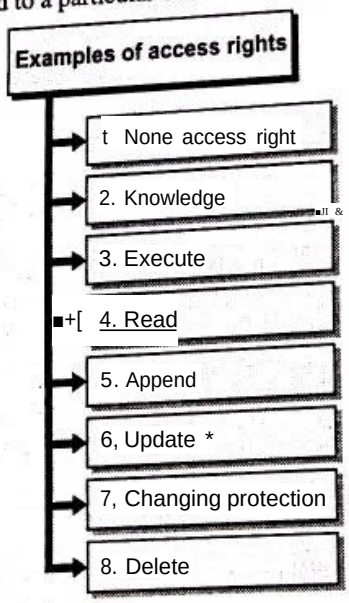


Fig. C5.8 : Examples of access rights

1. None access right

- The user may not still know about presence of the file. To put into effect this access limit, the user is restricted from reading the user directory that contains this file.

X Knowledge

With knowledge access right, the user can find out existence of file and owner of it. The user can appeal the owner for added access rights.

3. **Execute**

A loading and execution of p rogram is possible to user but copying it is not allowed.

4. **Re4id**

a rend access right allows the use to teed the flle for C.y work, together with copying and execute

5. **Appe nd**

The user can only add data to the file at the end.

6. **Update**

The upd access right allows the user to modify, delete, and add the data to t... file's data.

7 **Changing P rolec„on**

the access rights assigned to ote, Th " Normally this right is associated only with the X of the file. The owner is allowed to extend this Xht to other users in some systems

g. **Delete**

The user is allowed to delete the file from the fik | s"stem as per need.

These rights assigned to usei y to user form a hierarchy, wih each right involving those that precede it. Thus, if any user is assigned the read right for a particular file, then that user also gets following rights : knowledge, execute. The person ible to create file is by default the owner of that file.

The owner is granted all of the access rights and may put rights to others.

The different categories of users to which accessc an be given are :

- Specific user : Individual users having its own us
- User groups : A group of users who belong 6 the particular user group. The system keeps tract membership of user groups.
- AU: Authenticated users of this system/

<r Simultaneous Access

- It is necessary to protect the share * simultaneous updating from more than one u When more than one user granted access to appen or update a file, the operating system or file managed t system must implement some way to restrict it.
- A brute-force scheme permits a user to lock the complete file when it is to ,be updated, It is also possible to lock individual records during upd ate.

5.1.1 3(B) Remote File System

Explain in tietail operafo

a mote file system.

Sharing of data in the form of fife communication over the netwwkX netted implemented is transfer using ftp

fhe second method is distributed fii e PFS, it rectories are abj e local machine. In World Wide Web (WWW is required to get access to the remote essence a wrapper for ftp supports both anonymous and authenticated access.

jn anonymous access, users do not have remote machine, they can transfer file? makes use of anonymous file exclusively.

A tighter integration is required in DFS bet ween the machine accessing the remote fifes the machin providing the files.

* The Client-Server Model

- Remote file system permits mounting of one or more file system from one or more remote computers. Machine storing the file is server and machine from which file is accessed is client.
- A server can offer the service to multiple clients, and a client can request to multiple servers. This depends on tow client-server facility is implemented.
- The server generally states the available files on a volume or directory level.
- Client is specified by network name or IP address. As tose can be spoofed, unauthorized user can access the server.
- Secured authentication is required using encrypted fe y>- During exchange of these keys, unsecured access 10 *• also possible.
- ,n UNIX and its network file system (NFS), the use " IDs on the client and server must match ill not be "“hentication. In mismatch case, the server w able to decide access rights to files.

Chuted Information Sy»»n«

Dis bibuted naming service is also used ieffl01e “ftifted access to information to suPP° rt """Putation.

Storage Management

For internet (WWW) domain name Systcm (DNS) name to (p address , Befo « wide use of exchange files between a many types methods Yellow pages are known as network industries adopted its r names, host and others. But arc not secured.

prefers the use of Light weight- Protocol (LDAP) as a secure d naming method.

Active directory is based °n light weight-directory comprise LDAP Edition to system-wide retrieval of information, for example availability of printers.

Organizations can use single distributed LDAP to store all user and resource information for all their computers. This allows secure single sign on for all the users.

Failure Modes

- As there can be failures in local file system due to various reasons, there are more failure modes for remote file system. This is due tq the interference in communication over network and complexity of network systems.
- There can be interruption in network due to hardware failure, poor hardware configuration, or networking implementation issues.
- A single failure can interrupt the flow of DFS commands.
- Due to crash or failure in server remote file system will be unable to reach. In this case, system can either terminate all operations to the crashed server or delay the operations until the server is again reachable.
- Implementation of the failure semantics is a part of the remote-fte-system protocol.
- Loss of data can be resulted due to termination of operation. Hence delay is preferred.

Recovery from failure requires maintaining the state information at both client and server. If server must know that it has remotely mounted exported file

approach.

systems and opened files. NFS use stress DFS
But this stateless DFS is not secure.

5.1.13(C) Consistency Semantics

. are evaluated

File systems supporting file shan g semantics. If one important criteria which is consistency should be user makes modification in file, then other user able to seeit.

UNIX Semantics

IX

Following consistency semantics are used by the UN file system.

- Other users who have file open, can immediately vi ew writes to this open file by a user.
- One mode of sharing permits users sharing of the pointer of current location into the file. Hence, If one user advances the pointer then it affects all sharing users. Here, a file has a single image that interleaves al! accesses, in spite of of their origin,

Session Semantics

Following consistency semantics are used the Andrew file system (AFS).

- Other users who have file open, can not immediately view writes to this open file by a user.
- Once a file is closed, the modifications done in it are visible only in sessions starting later on. Already open instances of the file do not reflect these modifications.

Immutable-Shared-Files Semantics

For fmmutable-Shared-Files, if creator of file deciares it as shared then changes cannot be made in it. The name of tan immutable file may not be reused and its contents may not be changed. Therefore, the name of an immutable file indicates that its contents are fixed.

Syllabus Topic : Protection

5.1.14 Protection

Q- Exptafofite system protection in detail, ~j

- Protection mechanisms refer to the particular operating system mechanisms which are used m n, . ~ information, fifes and resources in the computer protect
- Policy means whose data should be protected f rom

which and mechanism is how system put into effect these policies.

- In some systems, a program called reference monitor is used to impose a protection. Any attempt to access the resource is verified by reference monitor whether it is legal or not. The reference makes a decision - the basis of policy table.
- Modern prote mechanism has develop improve the reliability of any complex system i D to use of shared resources is involved. There are many reasons to ofl er the prote Jtion. which
- [t is t needed to provide the protection to avoid mischievous, deliberate violation of an access restriction by a user. It is necessary to make SUre that, each program component running in the system uses system resources as per defined policies to ensute the reliability of the system.
- Policies to make use of Che resources of CORlp ter system are put into effect by mechanism which is offered by protection. Some of the policies are included in the design of the system. Other policies are decided by the management of a system. Many policies are defined by the individual user of the system to protect their own files and programs.
- rt puntvnv j'stem should be flexible enough to oner different and put into effect the different types of policies. Different types of applications have different types of resource use.
- The policies designed should allow offering this resource use need. The resource use of the applications can change over the period of time. Hence, instead of relying totally on operating system, application programmer should use protection mechanism to protect the resources created against misuse.

5-1.14(A) Type of Accesses

Protection mechanisms give controlled access by restricting the types of file access that can be done. Access is permitted or denied because of a number of factors. Asan example of this is the request type. Followings arc the type of operations that may be controlled :

- Read : Indic ates reading from the file.
- Write : Indicates writing or rewriting the file.
- Execute and : Ind,cates file must be loaded into menKfl then carry out execution of it.
- APPend . Write new content at the end of the file-
- **Delete : D**eletion of tile and reuse the freed space-
- **List : List the name** and attributes of the file.

Sfations, example rena J 5-tS

Up- ...s, copying, and editing the file, may also be controlled. In many systems operations are implemented as system ...

Z > * only the low « tevet. For ex > * 01cclion a'fposwfle 'o implemen ' I" [8] > by a s 2 P ,e - copying a V J Hence. a USCr haVing re «d access can a, so cause i*[1] |S l0 be copied, printed, and so on.

5.1.14(B) Protection Domains

The system contains many hardware objects CPUs, memory segments, disk drives, prinJ„ 5Uch 45 and many software objects such as nJ?' mag,l «* *t s es. or semaphores. Each object can be ref uses, files, erenced by * 2 Je o n every o bject 4 ite set of S performed. For exampie. WAFT and SIG J semaphores and READ and WRITE on files.

A system should enforce a mechanism, to restrict the processes from accessing the needed objects for which they are not unauthorized. The mechanism should also ensure to restrict processes to a subset of the legal operations when that is needed.

For example, process P has a permission to read file F but not of performing write operation on it. A set of object and rights pair is called domain. Each pair denotes an object and some subset of the operations that can be performed on it. One domain corresponds to one user and specify the permissions to user for certain activities. Consider the Fig. 5.1.10 three domains.

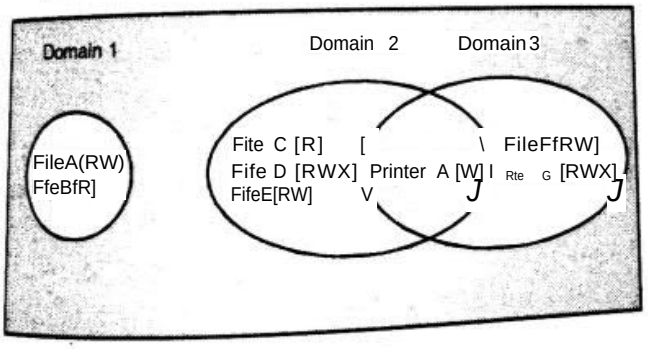


Fig. 5.140 : Domains

It is possible for the same objec ... r tn be in multiple domains, [Read, Write, eXecute] ngnis are ava jlable on each object. At a particular time o execution, each process executes in some protection domain. f objec B it can I" that domain there is some set o set rf rights access, and for each object it has some shown in square brackets.

During execution, processes can go from on e domain to other domain. The rules for domain pitching are from sysw m to Vc ry much depends on and v astern.

.Gti$;' d ®<'). TwrZL by arer-id M d gro up- s>J " mh, "«rio n W,,h llw simil " <U1D - p roce tuf objects. VC t0 P ' y thc that ls< Fr o m oncXm* fTOm 0SCT area 10 ken,tl arca , ac CtSs to a different se f 0 o t cr ' 'temc ' part has ... objects from the user part.

5.1.14(C) Access Control

The kernel ... from the us rT access 10 a dif f eR ot set of objects System keens track on whlch domain hv ... object belongs to which FiR 5 J i? 3 g access matrix Thc matrix for above «; d l0 lSs hOWninFig 5 1 U ... Darticnio... C4n tell if an access lo a given object in manner from a specified domain is permissible.

For this purpose, system makes use of this matrix and the current domain number.

Object

	Fite A	File B	File C	File D	File E	File F	File G	Printer A
Domain 1	Read write	Read						
Domain 2			Read	Read write Execute	Read Write			Write
Domain 3						Read Write	Read write Execute	Write

Fig. 5.1.11 : Access Matrix for Fig. 5.1.10

It is possible to switch domain in the matrix model by considering that a domain is itself an object, with the action enters.

... in domainl arc allowed to go in domain 2, tat once it is in domain 2. they cannot go back to domai ...

This s."mtaon models executing a SETUID program in UNIX. In this example, other domain switches are not permitted.

rpomain|	Fite A [Fite &) File C I		File D	File E	File F	File G	Printer A	Domain 1	Domain 2	Domain 3
1	Read Write	Read							Enter	
2			Read	Read write Execute	Read write			Writ*		
3					Read Write	Read Wn te Execute	Write			

Fig, 5.1.12 : Matrix nMd *| w ttli<lo"" ;n s switching

v Disadvantages of access matrix

- In fact storing the matrix is not preferred practically due to its large size and it is also sparse.
- Most domains have no access at all to most objects, so storing a very large, mostly empty, matrix is a wa ste of disk space.

The two practical methods to store the matrix are, first storing the matrix by rows or by columns, and then Storing only nonempty elements. In the first method, an ordered list consists of all the domains that may access the object, and in *what way*. This list is associated with each object and called as Access Control List (ACL). In the Fig. 5.1.13 shown below, three processes, each belongs to different domains X, Y, and Z and three files F1, F2, and F3 are shown. Each domain corresponds to exactly one user, X, Y, and Z.

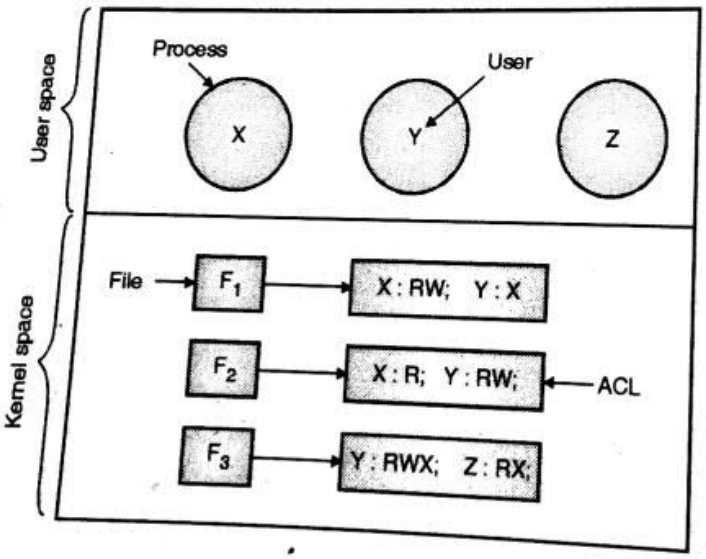


Fig. 5.1.13 : Access control list

- File entries in ACL are separated by semicolon. Each file *tai* an ACL coupled with it. File *F1* has two entries in to ACL. The firat entry indicates that any process begged by user X may read and w the file The second entry indicates that any process Xngedtouserrmayreadthefiie.
- Other than these accesses by these users are prohibitd. Apart from this, all accesses by other users are also prohibited. Again, process does not approve the rights, these are approved by user. Any numbers of processes belonging to user *X* can read and write file F1. It is the owner who plays the role for protection not the process ID.
- ACL contains three entries for file F2 : X, Y, and Z can alt read the file, and also Y has a right to write it. A pan from these accesses other accesses are not permitted File F3 seems to be an executable program, since Y and Z can both read and execute it. Y can also write it.
- Above example demonstrates the fundamental form of protection with ACL's. More sophisticated systems are frequently used in practice. Other than read, write and execute, many other rights are given in practice. Apart from, having ACL entries for single user, many system supports group of users. Groups have names and can be included in ACLs.

Sy *biiS-Topic File System Implementation

— — —*e System Implementation

Secondary storage such as disk is designed to hoMs huge of data permanently. The file system resides Perma? 0 ? 1 file svste* a ° n Secondar y storage. Implementation of Ble ideals with the following.

In what way fi,es Bd directories are ... management of the & tOre d. managed. that K h s H(?vv to ensure that everything Woi *. unfailingly Work efficiently pjsks off the mass of secondary st ... file system is maintained. The data ...orage On which a main menrory >sperformed in unit of b £ between di... *Lii> one or more sectors. These seeing ' * hich may iB w 4,096 bytes depending on disk ... differ fron > 32 of sector is 512 bytes. Because of ... drives used. Usual J is » suitable mediunl f« storing various fil ... llowing reasons

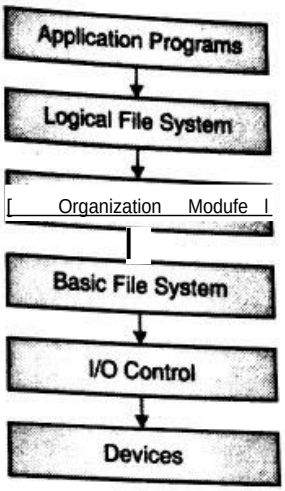
- A Hock can be read ffo m the ... es , modification can be written it back in ... Sk and after ... the same place.
- Any g ven block o f information can ... a« essed b y disk which makes it simple to access fi e either sequentially or directly, and switchingfr ... sorter requires only moving the read-wriT ... e heads to *waiting for* the disk to rotate ... and

5.2J Hie System Structure

[£" Explain file system structure? ----------1

- The file system contains several various levels. The layered design of file system is shown in Fig. 5.2.1.
- In this design, each level in the design uses the features of lower levels to create new features for use by higher levels.
- The lowest (bottom) layer consists of device drivers and interrupt handler to exchange information between main memoiy and disk.
- The file-organization module keeps information about files, their logical blocks and physical blocks.
- The file organization module can translate logical block addresses to physical block addresses for the basic file system to transfer. This is achieved by identifying the type of file allocation used and the location of the file.
- Logical block numbers for each file's block is numbered from 0 or 1 to N.
- The data is stored in physical blocks. These physical Woch do not match the logical numbers. Hence, a translation is required to find each block.
- Free-space manager is also a of file-organization module. It tracks unallocated blocks andW ... es these blocks to the file-organization modulo when ...

p' e anagem nt ... ~Ut logical flip metadata information is carried Metadata c ... d « s — ... structure. It The management of dire c ory structure is carried out by l°gica| fi| e system in Order to off « the file or S ization m ... paired, given □ * the informa hoQ the latter Return via fit. 1, mbolic Rie name. It maintains file FCS h X ... M0CkS (FCS) inf °m>atio n ato U| O ril J n « d « UNIX « « « ns P cr tnissions, etc. 38 Ownershi P' location, The ... secu ri<y X i s m 5tem alSO « * proteclion « d



Fifr 5.2*1: Layered File system

Advantage of layered structure

- Advantage of layered structure for file-system implementation is that it minimizes duplication of code. Multiple file systems can use I/O control and sometimes the basic file-system code. Each file system can then have its own logical file-system and file organization modules.

Limitations

- The limitations of layering is that it can bring in more operating system overhead, which may lead to decreased performance.

Syllabus Topic : Implementing File System

5.2.2 Implementing File System

Q. Explain the implementation of file system in detail.

5.2.2(A) File System Layout

- Disk surface can be divided in many partitions. Sector 0 of the disk is called MBR (Master Boot Record) used to boot the system.
- Each partition can hold the independent file system. Beginning and ending address of the each partition is given by partition table which resides at the end of MBR.
- When system is booted, BIOS reads in and executes the MBR. The MBR program locates the active partition, read in its first block, called the boot block, and executes it. Fig. 5.2.2 shows the possible file system layout.

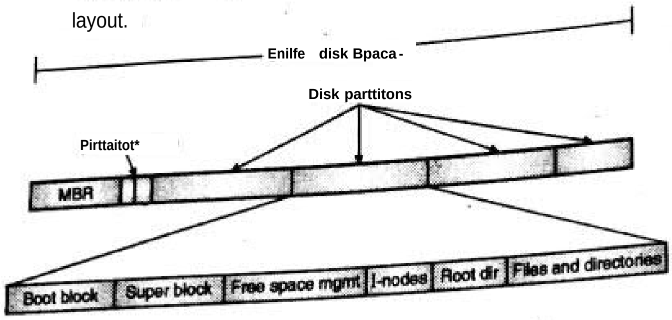


Fig. 5.2.2 : Possible file system layout

Bootblock

- For homogony, every partition begins with a boot block, even if it does not contain a bootable operating system.
- In future operating system might be loaded in that block.

Superblock

- When system boots, all key parameters regarding file system is read in memory from this Superblock.
- Usual information in the superblock includes a magic number to identify the file system type, the number of blocks in the file system, and other key administrative information.

Free space management

To keep track of free disk space, the system maintains a free-space list which records all free blocks.

I-node

The information regarding each file in file system is kept in data structure called I-node. For each file there is one i-node.

Root directory

It is the top of file system.

Files and directories

Files and directories in disk.

5.2.2(B) Virtual File System

Q- Write notes on virtual file system.

All modern operating systems must support multiple types of file systems simultaneously. A straightforward method of implementing multiple types of file systems is to write directory and file routines for each type. Instead, most operating systems, including UNIX, use object-oriented techniques to simplify, organize, and modularize the implementation. The file system implementation includes three key layers. The first layer is the file system interface, based on the open(), read(), write(), and close() calls and on file descriptors. The second layer is called the virtual file system (VFS) layer having following two functions:

1. It separates file-system-generic operations from their implementation by defining a clean VFS interface.
2. The VFS offers a means for uniquely representing a file throughout a network. The VFS is based on vnode, which is a file-representation structure. The vnode includes a numerical designator for a network-wide unique file. This network-wide uniqueness is needed for support of network file systems. The kernel keeps one vnode structure for each active node (file or directory).

Thus, the VFS differentiates local files from remote ones. The local files are again distinguished as per their file-system types. The activation of file-system-specific operations is carried out by VFS to handle local requests consistent with their file-system types and even calls the NFS protocol procedures for remote requests. File handles are bulid from the relevant vnodes and are passed as arguments to these procedures. The third layer of the architecture is the layer implementing the file system type or the remote-file-system protocol.

- The object-oriented principles are used to design VFS. It has two modules: a set of definitions that states what file-system objects are permissible to seems to be and software layer for these objects manipulation.
- Following four major object types are defined by VFS:
  1. An individual file is represented by i-node object
  2. An open file is represented by file object

3. At, e01ire file system ' is rapt esented by sUP erblock object.

b- A" iadiVidual direct entry 's nted by eDt ry object.

- A set of operations are defined for each of these types of objects. Each object of one of these types Points to a function table. The record of addresses of the actual functions is kept in function table. These functions implement set of operations for that object.

- The VPS software layer need not know about what kind of object it is dealing with. It can carry out an operation on one of the file system's objects by invoking the right function from the object's function table.

- The VFS remains unaware about whether an i-node presents a networked file, a disk file, a network socket, or a directory file. The function table contains suitable function to read the file.

- The VFS software layer will invoke that function without worrying about the way of reading the file.

- The file can be accesses with the help of i-node and file objects. An i-node object is a data structure that holds pointers to the disk blocks. The actual file data is present on disk block.

- The file object denotes a point of access to the data in an open file. In order to access the i-node's contents, the process first has to access a file object pointing to the i-node.

- The i-node objects do not belong to single process. Whereas file objects belongs to single process. The i-node object is cached by the VFS to get better performance in near future access of the file. Therefore, although process is not currently using the file, its i-node is cached by VFS.

- All cached file data are linked onto a list in the file's i-node object. The i-node also keeps standard information about each file, for example the owner, size, and time most recently modified.

- Directory files are treated in a different way from other files. The programming interface of UNIX define several operations on directories, for example creation, deletion, and renaming of a file in a directory.

- system calls for these directory operations do not have need of the user open the files concerned, u...

the case for reading or Writing data. Therefore these directory operations are defined by VFS in the i-node object, instead of in the file object.

- The superblock object represents files of entire file system. The superblock object ... in kernel keeps a single file system ... for each disk device mounted as a file system and ... network connect ... networked file system at ...

- The main ... of the superblock object is to offer access to i-nodes. The VFS recognizes each i-node by a system ... number ... to find the i-node corresponding to a particular i-node number by requesting ... the superblock object to return the i-node with that number.

- A entry object represents a directory entry that may include the name of a directory in the path name of a file (such as /usr) or the actual file.

Syllabus Topic : Directory Implementation

5.2.3 Directory Implementation

> **Q.** Explain approaches for implementation of directory.

Efficiency, performance, and reliability of the file system depends on which directory-allocation and directory-management algorithms are used. Following are the algorithms that are used for directory implementation.

5.2.3(A) Linear List

- Every file must be opened before it read. User provides path name to access the file. When a file is opened, the operating system uses the same path name to find the directory entry. The directory entry contains the information needed to find the disk blocks of the file.

- Every file also has its attributes. Different file attributes we have already discussed. Directory entry stores these attributes of a file.

- In the simple design, a directory consists of a list of fixed-size entries, one per file, containing a file name of fixed length, a structure of the file attributes, and one or more disk addresses(up to some maximum) denoting where the disk blocks are. Following Fig. 5.2.3 shows this design.

Flit	Attribute*
A	attributes
B	attributes
C	attributes
D	attributes

Fig. 5.2-3

- Fig. 5.2.3 shows simple directory containing fixed-size entries with the disk addresses and attributes in the directory entry.
- Some systems use ,,no deli , In this directory entries are just name of the file and i-node number. All the attri— ui a ttle 15 stored ·n ·-node.
- F lg- 5 2 4 shows this design.

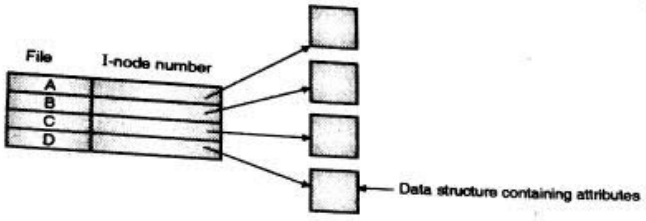


Fig. 5.2.4 : A directory in which each entry just refers to an i-node

- Above two approaches are corresponds to MS-DOS which have short file names of 1-8 characters and optional extension of 1-3 characters. In UNIX Version 71 file names were 1-14 characters, including any extensions.

All modem operating system support longer and variable-length file names. Following are the two approaches to implement it.

- The simplest solution is to keep file length of maximum 255 characters. The above two designs then can be used in which maximum 255 characters is reserved for file name. Since it is not common to have all files such longer names, directory space would be wasted.
- In this approach, fixed size of all directory entries is not considered. Each directory entry contains a fixed section, which begins with length of the entry, and then followed by data with a fixed format which are generally file attributes. It is shown in Fig. 5.2.5.

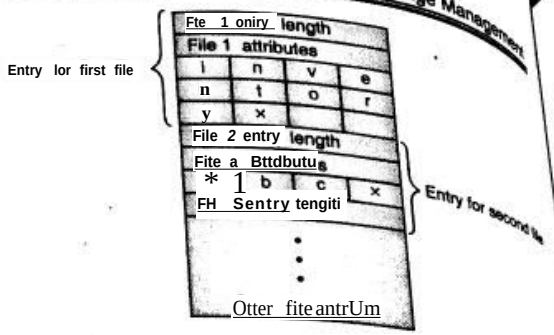


Fig. 5.2J: In-line handling of i-node names

- In this example two file inventory and abc is shown. Termination of each fi|p ; u * symbol.
- -- —• ____ —apu uiv'unvFU inql after removing file, a variable-sized free space is ' initiated into the directory. It may happen that, ne xt file t0 not be accommodated in this free s piiCe o Co!S*- may
- Solution on this issue is compaction of directory. Another problem is that a if sing le directory entry spreads in several pages, page f aull occurs during reading of a file.

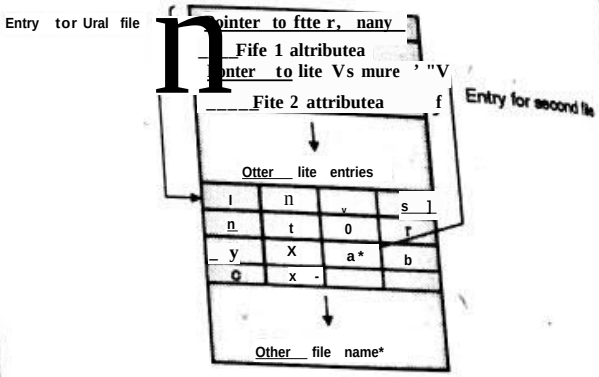


Fig. 5.2.6 : Long name handling usingheap

- The second approach to deal with variabte-kntih names is to allow directory entries of fixed length and maintain the file names jointly in a heap at the end of the directory as shown in Fig. 5.2.6.
- The drawback discussed above is defeated here-When I an entry is removed, the next file entered will *a!! fit in freed space. The management of heap wilted burden here and page faults cannot be avoided as 1C previous method.
- AU the designs discussed, suppons searching directory sequentially from beginning t0 et

tiding up the. fi le name . Seo... " Ual search Wn . deot if directories arefo uld H jamming the linear list m er k.

more time " "" U,e - To create od is simple hut ta> of directory to Ca out to ' new file, searching file bus *the* same name. A f[e , tha ? ' that no end of the directory fa add(Jd '• » new eotrv """J „ «d file « "««hed in di rector y tion of ' the f0 this file JS released. space «>e

Following many options can be th...

- The file entry can be marke . * 'o teus e the file
- *Assign* a special name to fit, unused blank name. CXan) Ple an an.
- A used -unused bit in each cntl * can be maintained.
- Attach it to a Portree directory entries.

A Adoption is to copy the last entrv into the released location and to red n the diiecfo D the directory. A linked list can also be ice the ength of the time required to delete a file. used t0 decre ase

. Solution to this problem is use of hash table m each directory as hash search i s e ffici ,CIU COrn pare to sequential search.

5.2.3(B) Hash Table

~ *In this* method, a linear list is used t0 stQre entriesJn addition to this, a hash Jata structure is Z used. The hash table uses a value computed from the file name and returns a pointer to the file name in the linear list As a result, it can significantly decrease the directory search time. Insertion and deletion are also quite simple. It is require to make some provision for collisions-situations if two file names hash to the same location.

The main problems with a hash table are its usually faed size. The hash function is dependent on size of hash table. Consider a linear-probing hash table l,n j[holds 64 entries. Using hash function, file names gets converted into integers from 0 to 63, For creating 3 file, we must increase the size of directory hash ta e * for example to 128 entries. Consequently, we fei)u, rc a new hash function which will convert file JajI]es to the integers ranging front 0 to 127. to this Case, I' is necessary to reorganize the present directory entries to reflect their new hash-function values.

Alternate solution is to use a chajned ,0 „ e, Tr* eac b table. In this table, a linked list is maintained ... llision can entry instead of an individual valwj?£ ----------

be handled by adding th... Although ... ui >« o Ching £ ,t to <he Unkcd "■'O-Xo ...mplete » is out " " link , d Ust is director ter d> "" s««h

Syllabus Topic : Allocation Methods

5.2.4 Allocation Methods

Q. What are th ---------- **(Dec. 14. May 16)**
r_t9r ce 10 FiteT tX 8 „0 ,10 „ me ®d8 with
MU - Dec. 2014. May 2016. 10 Marks

Disks supports for direct aCCeSS ' 10 which we can have flexibility in ... of files. There should * Op, »»"'n e * **-* allocati onofdto ... space. the Optln >um and eff ... « should always allow for the access >o the f lles w u r™"" o f SP"' so that quick USed majorly in different UpPO,Ud blowing methods are ---------- grating systems

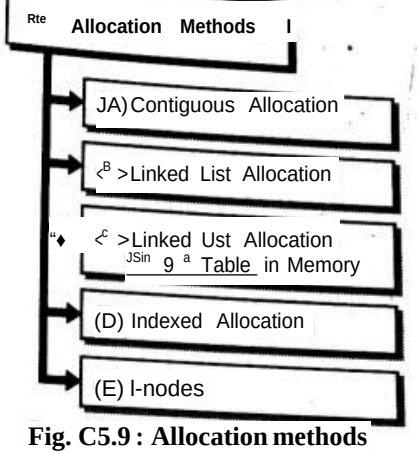


Fig. C5.9 : Allocation methods

5.2.4(A) Contiguous Allocation

Q. Explain contiguous file allocation method with its advantages and disadvantages.

- In contiguous allocation each file takes up a set of contiguous blocks on the disk. Disk addresses define a linear ordering on the disk. If each block size on disk is 2 KB, then 30 KB file would be allocated 15 consecutive blocks.
- Contiguous allocation of a file is defined by the disk address of the first block and length (total blocks occupied). If the file is *m* blocks long and starts at location *i,* then it takes up blocks i, i + 1, i + 2, ..., $i+n-1$.
- The directory entry for each file specifies the address of the starting block and the total number of blocks allocated for this file.

– *Directory entry* in Fig. 5.2.7 shows that file A starts at block 0 and it is 3 block king occupying block 0, block J and block 2. Similarly, file B starts at block 6 and it is 5 block long and so on.

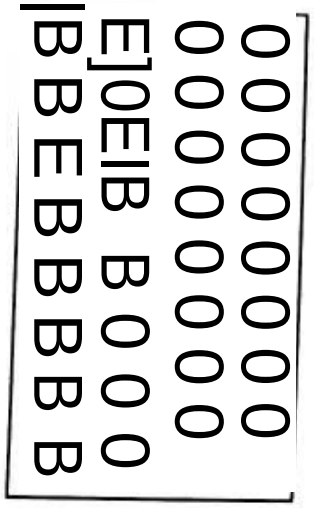
Advantages

– It is simple to implement because, only information needed to keep track on files block is starting block and length (total number of blocks occupied).

– For sequential access, the file system memorizes the disk address of the last block referenced and. when required, reads the next block.

– To access block k of a file directly which starts at block i, we can immediately access block i + k. Thus, both sequential and direct access can be supported by contiguous allocation.

– Performance is high as whole file is read from disk in single disk operation. To reach to the first block only one seek is needed.



Disk space

Dimdnry

Fite	Star, Woe*	Length
A	0	3
8	6	S
C	14	B

Fig. 5.2.7 : Contiguous Allocation of disk space

Disadvantages

– When allocated file is deleted, continuously allocated blocks to the file become free. For new file to allocate, these blocks might not be sufficient. If new file size is small than previously allocated size, some holes remains which are not enough to allocate any new file. As a result, the disk ultimately consists of files and holes causing external fragmentation.

– In some cases, it is simple it is simple to find out space is needed for a file to be allocated. For output file this size estimation becomes difficult. For CD-ROM, contiguous allocation is reasonable as all file sizes are known in advance.

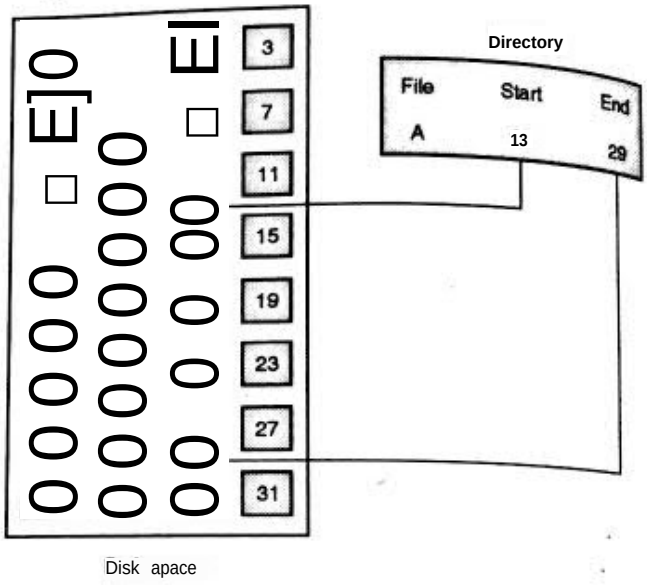
5.2.4(B) Linked List Allocation

Q. Explain linked list file allocation method with its advantaged and disadvantages.

– Linked list allocation overcomes the limitati... contiguous allocation. In linked allocation, eacj.? of a linked list of disk blocks. The scattered disk ...it disk can be allocated to the file. The directory c[? ...ains a pointer to the first and last blocks of the fife

– Creation of a new file is simple. For this, it is required io create a new entry in the directory. The directory entry holds a pointer to the first disk block of the file. The pointer value is nil for empty file and file si... this file is zero. * for

– The pointers need to be followed from block to ki...ock in order in read a file.



Disk apace

Fig. 5.2.8: Linked List Allocation of disk space

– Fig. 5.2.8 shows shows linked list allocation for file A The file A of five blocks start at block 13 and continue at block 16. then block 21, then block 24, and finally block 29. Each block holds a pointer to the next block. These pointers are unavailable to the user.

- User can see the block size excluding the size required to store the address. Accordingly, if each block size is 512 bytes, 4 bytes are required to store address, then the user sees blocks of 508 bytes.

Advantages

- In linked list allocation, every disk block can be used which is not possible in contiguous allocation. Hen«> there is no external fragmentation except for internal fragmentation in the last block.

– Reading a file sequentially is simple.

– It is not necessary to compact disk space.

ZJ, dva nta 9 es

Z o Z inef6 cient to su Pport a rand- m acce ss c *-pability.

To retrieve to block i t the onemf system has to start at the beginning and read the i – 1 blocks previous to it, one at a time.

In each block pointer takes some space, so each file require slightly more disk space rather than it® actual

jj e. if pointer is lost or damaged f tJe WOuld become

z jj essibJe ', A tfroog pointer can be chosen due

operating evstem Software breaka° bUg « the hardware. As a result of ww ch. J r, of disfc result in unking into the ftee- space list . could tun, file. nto another

5.2.4(C) Linked List Allocation using a Table in Memory

In linked list allocation, each block information, therefore entire bkvt *o slore used to store file content. This linX.J! n o« fully eliminated by keeping pointer inform ation i n Ub,e # bich always remains m memory,

If file A occupies block 5, 9, 1, 12 and 7 in the same odcr and file B occupies block 3, 8, 4, 14 and 6 in the same order. The File Allocation Table (FAT) is show n io Fig- 5«2.9.

pbydcal Neri block	
0	
1	12
2	
3	8
4	14
5	9
6	– 1
7	– 1
8	4
9	1
10	
11	
12	7
13	
14	6
15	

← File B begins here

← File A begins here

Fig. 5-2.9: Linked List Aitoa** 100

Advantages

- The entire block is available for data. So all disk space is utilized for allocation of file.
- Random access is muclofTT*' The Chain need to U Jllow to gel in memory 5 1 flle > but the chain Direct any d ,Sk references. * f o Howed without number) ofX'fik d St in8 address (ime *cr block e n though fl le size d XT to locate all blocks

Dlsa vantage

- work. Ubie TMUSt ** *n "*TM»y all the time to make it

5.2.4(D 2) j exed location

Explain indexed file allocation With advante 9es nd disadvantages.

list alloclf A1IOCatiClT1 Table AT) in memory, linked table m °il SUPPOrt raild0Tn access , but the ent ire allocation. all the pointers are kept in one location called as index block.

- The - a jiiuca djock assigned to each file and this mdex block holds the disk block addresses of that particular file. There is a pointer from i^th entry in the index block to the 1^th block of the file, it means n* entry in index block holds the address of the 0^th disk block. The address of the index block of the file is maintained in directory.
- In order to locate and read the i^tb block, pointer in the i index-block entry is used.

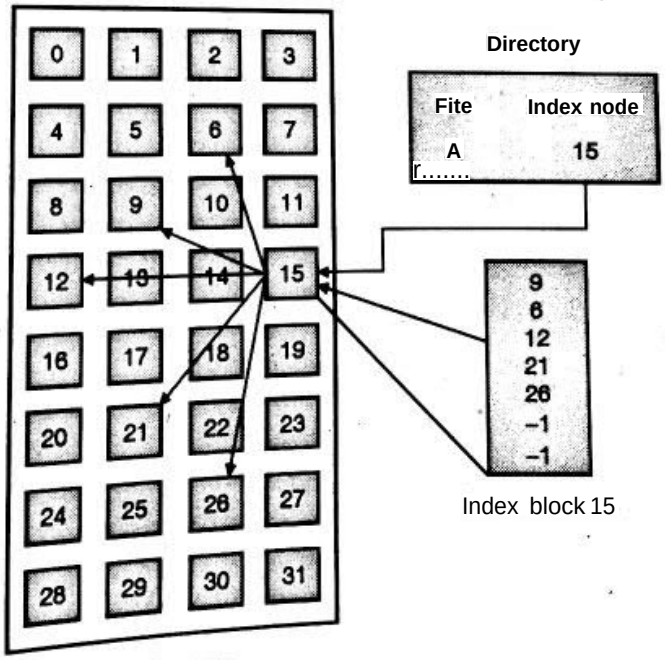


Fig.5.1W: Indexed Allocation of disk space

Initially when the file is created, all pointers in the index block are initialized io nil. When the i“ block is first wntten. a block is obtained from the free-space manager. and its address is put in the i^{th} index-block entry.

Advantages

- Indexed allocation supports random access.
- There is no external fragmentation, because any free block on the disk can be allocated as per request for more space.

Disadvantages

- The pointer overhead of index block is more with compare to the pointer overhead of linked allocation.
- Indexed-allocation schemes suffer from some of the same performance problems as does linked allocation.

UNIX Example

The structure of i- node

Q- Explain the structure of i- node in detail.

The information regarding each file in file system is kept in data structure called i-node. For each file there is one i-node. An active i-node is associated with exactly one file, although many file names may be associated with a single i-node. Every file is prohibited by precisely one i-node. Operating system manages all types of file by means of i-nodes. The exact i-node structure differs from one implementation of UNIX to another. The file attributes as well as its access rights (permissions) and other control information are stored in the i-node. The FreeBSD t-node structure is shown in following Fig. 5.2.11. It contains the following data elements:

- The mode of access and type of the fife.
- Identifiers needed for group access and h$_o$ i der of the file-
- The time at which the i-node is just now revised by the system and the creation fime of the file.
- The time at which file is just now read and written.
- The file size which is in bytes.
- The series of block pointers.
- Total number of actual physical blocks Occtaiea file. It also comprises the blocks holdin oy the 8 indirect pointers and attributes.
- Total number of directory entries which refers tllA file.
- The flags which user or kernel Can which demonstrate the uniqueness of the fife.
- An arbitrarily chosen number allocated to node every time that the latter is assigned to a new fit as generation number of the file. The same numfe* used to identify references to deleted files.
- The data blocks size pointed by the i-node. Usually it is the same as, but sometimes larger than, the fifesy tem block size. FreeBSD has a minimum block rf 4,0% bytes (4 Kbytes).
- Extended attribute information size.
- Zero or more extended attribute entries.

I-node table, or i-node list present on disk, contains ife i-nodes of all the files in the file system. When a fik b opened, its i-node is carried into main memory and stored in a memory-resident i-node table.

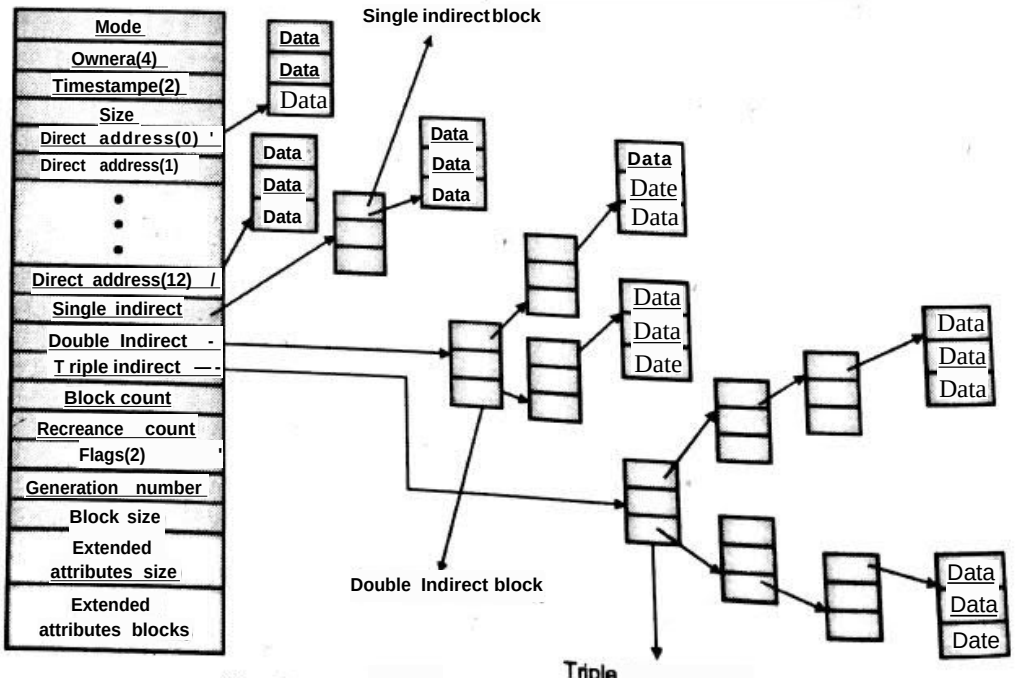


Fig. 5.2.11 FreeBSD i-node and file structure

5.2.4(E) I-nodes

Q. How i-node is used to allocate the file?

- I-node (index-node) is a data structure which lists the attributes and disk addresses of the file's blocks. I-node is associated with each file. It tells which block belongs to which file.
- If particular file is open, only its i-node is in memory. This is beneficial in comparison with linked list allocation which requires the file allocation table in memory.

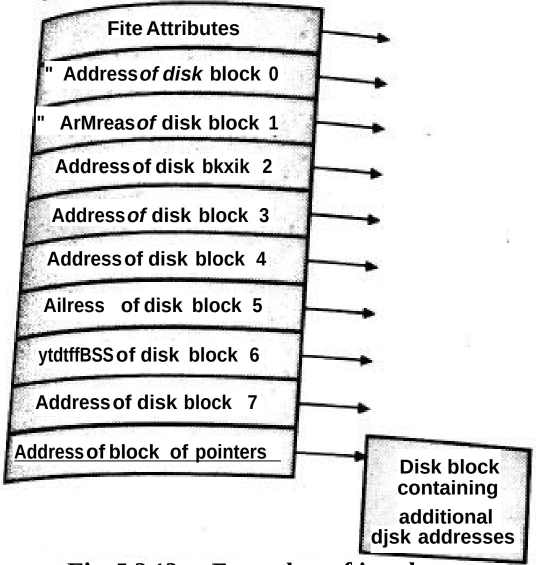


Fig. 5.2.12 : Example of i-node

- The size of file allocation table is proportional to the number of blocks that disk contain. If disk has k blocks file allocation table will have k entries.
- In contrast, the i-node scheme requires an array in memory whose size is proportional to the maximum number of files that may be open at once.
- Fig. 5.2.12 shows example of i-node.
- The last disk address is reserved for the address of a block containing more disk block addresses.
- The reason is, each i-node has space for fixed number of block addresses. If file grows beyond this limit, this last disk address point to disk block containing additional disk addresses.

Syllabus Topic : Free Space Managements

5.2.5 Free Space Management

When file is deleted, the disk space becomes free. This space should be reused to allocate to new files. System keeps track of free blocks. The list which maintains all free disk blocks which are not allocated to file or directory is called free space list. To create a new file, free space list is searched for required space and allocated. The space is then removed from free space list. Again, free space is added to free space list after deletion of file.

Q. List methods of free space managemant.

- To implement free space list following are used two methods

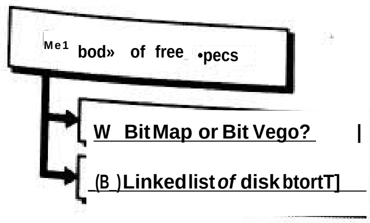


Fig. 5.2.10 : Methods of free space

5.2.5(A) Bit Map or Bit Vector

Q. Explain bit map method in detail.

- In this method, every block on disk signified by a 1 bit. Free block is represented by bit 1 and allocated block is represented by bit 0.
- Suppose block numbers 3, 5, 7, 9, 10, 15, 18, 20, 23, 25, and 28 are allocated and rest of the blocks are free. The free space map would be as follows. Allocated blocks are shown in bold space.

11101010100111101101011010110............

- With a bitmap, it is also possible to keep just one block in memory, going to disk for another one only when it becomes full or empty. Since the bitmap is a fixed-size data structure, if the kernel is (partially) paged, the bitmap can be put in virtual memory and have pages of it paged in as needed.

Advantages

- It is relatively simple and requires less space, since it uses 1 bit per block.
- It is efficient in searching the first free block or k successive free blocks on the disk.

Disadvantages

- The whole vector is needed to keep J wRtten to disk Otherwise it would be inefficient- occasionally for recovery needs.
- Keeping bit map in main memo Je for larger smaller disks. It is not necessarily P disk size.

5.2.5(B) Linked List of Disk o

Q. Describe linked list of disk block rnetlhod

- In this method of free space management all the free disk blocks are linked together
- The pointer to first free block is kept in partl L s location on disk and cached it in main memory first block contains a pointer to the next rec block, and so on.
- Zn our bit map example, we would keep a pointer to fi free block which is block 0. Block 0 would contain the pointer to block 1, which would point to next ree block, which is block 2 and so on. In Fig- 5.2.1 shown, block 0, 1, 2, 4, 6, 8, 11, 12, 13, M, lb. 17*1 » 21, 22, 24, 26, and 27 are free blocks.
- This method is not efficient because, we must read each block in order to traverse the list, which requires extensive I/O time. But advantage is that, traversing the free list is not regular operation.
- Usually, the operating system simply needs a free block so that it can allocate that block to a file, so the first block in the free list is used.
- The FAT method includes free-block accounting into the allocation data structure. No separate method is needed.

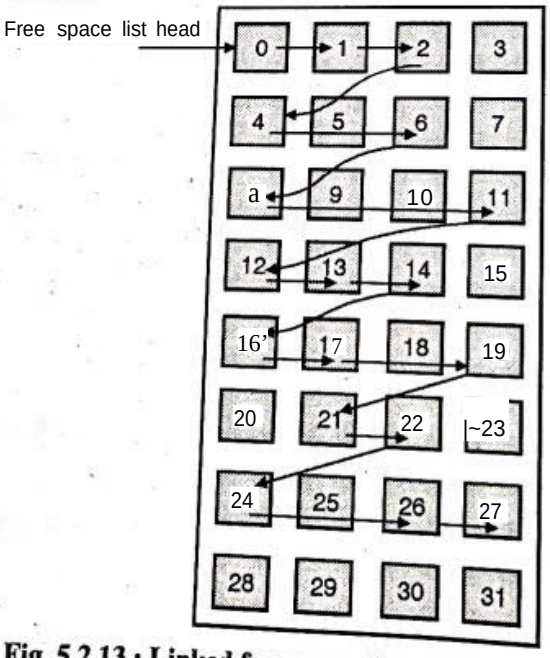


Fig. 5.2.13 : Linked free-space list on disk

5.2.5(C) Grouping

This meth #(|(Jrcsses of „ f, ee blocks are In mis meth* . * Actually . the first n-1 of thcse-biZ' 4 * ■he flrstf, 7« t block stores the addresses of % ftee. The ® As compare d wtth standard link -list blockS ' r.he addresses of a large number of free bloc« can -Si-""

5.2.5(D) Counting

- allocation algorithm is used for space L f l“aUon°* e „ many Co nl,gU , US blocks may be allocated or freed simultaneously. Hence, instead of keeping a list of n free disk addresses, it advantageous to keep the address of the first ftee » and the number (n) of ftee conttgueus block, 2 follow the first block. The entry ln free-space list denotes a disk address and a count.
- In this approach, more space is needed for each en|q 1 with compare to a simple disk address. But adva,age is that, the overall list is shorter, as long as the count is generally greater than 1.
- Instead of linked list, a B-tree can be used to store these entries so that, efficient lookup, insertion, and deletion can be carried out.

Syllabus Topic : Efficiency and Performance

5.2.6 Efficiency and Performance

Q. Explain various techniques to improve efficiency and performance of secondary storage.

As disk is slowest memory component, it may become bottleneck in system performance. There are various, techniques to improve efficiency and performance of I secondary storage.

5.2.6(A) Efficiency

- The disk allocation and directory algorithms used mainly decides the efficient use of disk space. Tte preallocation of UNIX inodes is done on a voW[1] This results some space occupied by inodes altho[111] disk is empty. Preallocating the inodes and. scatterin un em across the volume improves file syst - P ormance. The UNIX allocation and free-space algonthms maintains a file's data blocks near that ile's inode block to lessen the seek time.

It is ssary to consider the types or data that is kept fn a file's directory (or inode) entry. Normally, the recorded write date" gives the informa- user to decide whether backin- done or not. Some sys(em "P of file " <o t date." Which is useful t0 U5et the - las| u * b e #as last read. *>>»» the file

As this information is maintain writing th* fin'd in the directory > Astern. j(h- file * Block " essential l0 *henev£* then if section Changes, and if the h- be back oul to disk, as o ** 3110 requi- place only in block chunks. cX d · n S On disks ta£ situation, if any time a file is o "« pract ical its directory entry must be read and wri £ " "8 «*n

s necessity cannot be efficient for r " a,S o· jessed frequently. Therefore, We mu . Which are benefit against its performance cost wh about its file system. Tn general, all fj]e en designing a to be considered for its effect related data items needs performance. 00 efflc,e &cy and

The size of pointers used by systems is differe„ L , f system uses 16 bit then length of fi le restricts to " KB. For 32 bi! pointers, file length supported restricted to 4GB. Some systems use large size penmen, which occupies more disk space.

5.2.6(B) Performance

Accessing the memory is faster with compare to accessing the disk. For accessing single word, the memory access is on the order of a million times as fast as disk access. Due to this differentiation in access time, many file systems have been designed with various following optimizations lo improve performance.

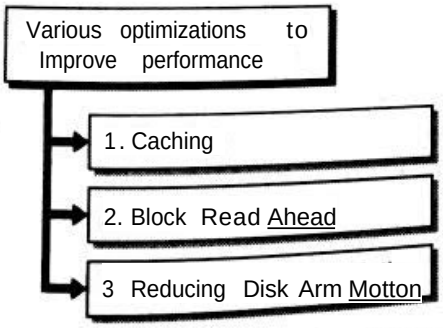


Fig. C5.11 : Various optimizations to improve performance

1 QtW „ 8 u • the most 8 ral Block cache or buffer cache is relation to technique used to minimize disk accesses. In

this, a Cache- 101ht d«k w;* 8rou Porbi,» clcs that logically belong * of Perth "e '*'>« Cpt in mory for the and utilized to handle the c *C|ic, VCra) *8orithms exist The most 8w , t41 algorithm is to make certain about all c*he?* SIs to notice if the needed block is in the fulfilled not. If it is in cache, the read request If lbc b| * k "Ut a disk access. « * cache, "d then coXd t U « f TM »«> the >. sqcoes sive r Xs f T" « « needed Aft « h,lr,U 'd f ro >n thc X'e blOdt c « be ln cache there ,KSKs ar y l o h a v? n be lar8e „ umber o f bl «ks. It is 1 bl «k. Hash , hl technique to locate blocks >1 cache ' o r8" i"«® is used to organize value are linked together form,n g a linked list so the Elision n chain can be followed,

To bring a new block needed 1010 1 fUU Cache « some block « been m r- rem o VCd iU,d tran8fer to to dlsk , lf U has circumst SltKC being brou 8ht in - TMS normal page replacement algorithms such as FIFO. LR U> Wopnate to replace block from cache.

Since cache references are relatively occasional, it is practicable to keep all the blocks in exact LRU order with linked lists. Fig. 5.2.14 shows buffer cache data structures.

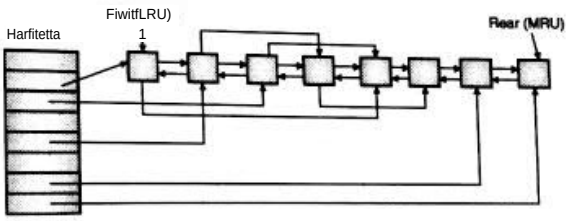


Fig. 5.2.14 : The buffer cache data structures

The Fig. 5.2,14 shows a bidirectional list running through all the blocks in the order of usage, with the least recently used block on the front of this list and the most recently used block at the end of this list. When a block is referenced, it can be removed from its position on the bidirectional list and put at the end. In this way, t L.RU order can be maintained. It seems that LRU **is Arable, treasons are;**

i-node like critical block is read into the cache Stl ified but not rewritten to the disk, a crash will leave the nlesystem in an inconsistent state. H the i-

node block put at the end of the LRU chain, it may be quite a while before it reaches the front and is rewritten to the disk.

- Some blocks, such ak i-node blocks, are not often referenced two times within a small interval.]n order to consider the above two factors regarding crashes and file system consistency, modified LRU scheme is possible. The blocks can be categorized as i-node blocks, indirect blocks, directory blocks, full data blocks, and partially full data blocks. Blocks that will most likely not be required again quickly go on the front, instead of the rear of the LRU list, so their buffers will be reused quickly.
- Blocks that might be required again soon, such as a partly full block that is being written, go on the end of the list, so they will stay around for a long time. If the block is necessary to the file system consistency and it has been updated, it should be written to disk straight away, in spite of which end of the LRU list it is put on.

2. Block Read Ahead

- The performance of file system can be improved by bringing block in cache before it is required. Whenever file system create n* block, it checks if n+l* is available in cache or not. If not available, it brings it in cache taking into consideration that, it would be required in future.
- This read ahead policy does not work for random access and only works for files that are being read sequentially. The read block in cache would force to replace the useful block and it might not be required. The file system can keep track of the access patterns to each open file. If next time file read is suppose sequential and not random, the read ahead strategy is useful otherwise not.

3. Reducing Disk Ann Motion

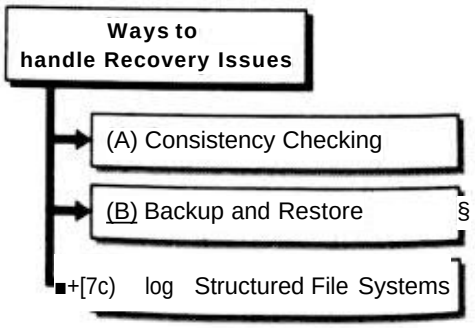
- Apart from the caching and block read ahead technique to improve performance, other technique is to minimize the total disk arm motion by keeping blocks that are likely to be accessed in succession close to each other, if possible in the same cylinder. In the following ways performance can be improved
- If bitmap technique is used to record free blocks, and the entire bitmap is in main memory, then it is simple to select a free block as close as possible to the previous block
- Grouping of the blocks can be done with free list. Now. system will keep track of disk storage in groups of consecutive blocks instead of only block- sedor holds 512 bytes, lhe sX««n coul? / oti, blocks occupying *1* sectors but assign disk st—age 1-KB units of 2 blocks occupying 4 sectors. j,
- |D this case, cache would still use 1-KB b ,Ock- disk transfers would still be 1 KB but readi„ ' ** sequentially would minimize the number of J? r"e factor of two, considerably improving petfood by variation on the same theme is 10 take Kc J' A rotational positioning. When allocating bloc, of system tries to place consecutive blocks in a flle ' "k same cylinder. die
- Those systems which use i-node like data structq,.. that two disk accesses are required to access ev ' n short file : one for i-node and other fo, J" 1 Generally i-nodes reside at the start of disk having average distance from its blocks about half th number of cylinders.
- Because of this reason long seek is required. Performance can be improved by placing i-nodes the middle of disk which would minimize the average seek between the i-node and the first block by a factor of two.
- Other approach to improve the performance is to the disk into cylinder groups, each with its own / nodes, blocks, and free list. When creating a new file any i-node can be selected, but a try is made to search a block in the same cylinder group as the i-node U nothing is available, then a block in a nearby cylinder group is used.

Syllabus Topic : Recovery

5,2.7 Recovery

Q. Explain file system recovery in detail.

Directories and Files reside on disk or remains in main memory. System crash may lead in data loss or in data inconsistency. Following are the ways to handle these issues.



f Ifr C5.I2 : Ways to handle recovery issue*

5.2.7(A) Consistency Checki...

In order to get the pe rf ... 06 ""9 information, it is cach ... in acce...

updated copy of dir ectory in... *=Pt in T ... k di, ecto- memory. Whereas co rrespon... memory. g. <«* is old copy as cached ... information resides in main cssen tially written to disk On ... -y info ... n o„ the

there can be loss of co ... w "Pdate is not also of TO operations curn ... J. Cacl * an d h * ma cbinc crash. As a res ult ... n, y Wor ... anq drones of opened f 1les ... y " Uion, £ ... af ... a Ieadi the file system i n M - ** lost Th ... in the state the actual state of son* ... consistent s ... Won compare to present i n t. "** 's diff er ... n "lis Regularly, during reboot specie ... rec, °ry „ *hh sU,e and correct disk 'neons, slen, £" ... gram verify tfe * such consistency checker p r o g * ... The ... chkdsk in MS-DOS. 8 dn,s "ft/sck in ... »f ... X or

The directory structure data and ... are com P ared b * the » prt>g rams , ? 1(o Cks Of > disk KCU rs then tries to correct any incor ... any ... match The disk space allocation and f ree ... stenck « if exist algorithms direct the checker proe/'"'*" 7 '-"'! ... tement type of problems and how ... to dis C0'er the correcting them. SUCC -M it will be in

- in case of linked allocation, a link is block to its next block. ThereforT 6 -*"' My reconstruct the file from the data blwfc ... easy to directory structure is created again ... After this ,
- In case of an indexed allocation, loss n f , a- entry is difficult to handle. This i s because thTdaZ blocks are unknown about the other data blocks Therefore, For read accesses, UNIX caches directory entries. On the contrary, for write leading to space allocation, or other metadata updating, is carried out synchronously, prior to the corresponding data blocks are written.

*5.27(8) Backup and Restore

- If failure of magnetic disks occurs then loss of data should not be forever. Back up of data from disk to ° t storage devices can be carried out by system programs. Recovery of any data is now just the restoring the data from backup.
- The amount of copying can be minimized by making use of information from each file's directory entry. I k** up program is aware of the time of the last bac. P of a file, then file's last write date in the ... directory denotes that the file is unchanged since that date, then there " no need to copy the file again-

.Slcrm... Mana gemsnt

Full backup means all files are copied from disk to backup devices. Incremental ba... files changed ... ICe - Thii c da y 1 arc don P COpy of a11 ch ... ged The „ •Mneyek? «*« l y ?2 - day 3 „.. X bKI ,„p storage a «"" far files Eve.y ... nc * "*"•'•" 7. w unai d ' y -'• W , * toC * "* S> « Of7 k 7 Wri ,,,,, o W tb , ,,,,, ia Un g re r W ,b,e ,o restore' „ ack „ P mwli , to "• ... ghL-k wil h the full . . com P'«' disk by ... the incremental backups ... and continuing The period of this cycle is a findii ... amounts of backup i22* mi<Wte 8round «*'X2 e of days back from which a restore can be done. If a permanent backup is maintained far distance away from the regular backups then protection against hazards like fire can be achiev... more safely.

5. ... le Systems

- "■:r— applied effce... recovery algorithms can be 'becking. The im D|eni , , "* problem of consistency ... The disk file (nLd syste <™T' resulted in l o KbaMd joumalIng) fi| e systems. dit «'«y structures f re ,7? . SmKturcs for «">pl K pointers can leads' £ T* ... pointers, and f« FCB failure of system. "'consistent state due to Changes ... place whenTise" 2?* 10 ab o Ve 5tn,ctures ■» incorporated in n~. « 8 ,,based techniques was not results in 0PeCab „ S systems ■ The creation of file X on ra Z S X Ur te « Wi,hin,heBle modificauonof dhretory struck, XoXrTs th ... ui ... 1 DIOCKS and flwrefore free counts for all of these blocks are decreased. If crash occur then these changes can be interrupted and structures can leads to inconsistent state. These structures can be fixed after recovery. But in such scheme following difficulties may arise .
- The inconsistency may be beyond the repair. The consistency check made may be incapable to recover the structures, which can lead in loss of files and yet whole directories. In order to get rid of the conflicts, a human involvement is required. Availability of system will not remain to the user until the human directs it about how to proceed. For the consistency checking of huge size data can require hours of clock time to check.
- If log-based recovery techniques are applied to f le-systetn, metadata updates then the above problems 1 be solved. NTFS and the Veritas file system use

the log-based recovery techniques All changes done in *metadata* are written sequentially to a log-

- Thcsc written changes to lhe log considered to be committed. After this the control of execution o system call can return to the user process which f it carry on execution. In the meantime, replaying o log entries are carried out across the actual file system structures. Because the changes are done, a pointer is updated to show which actions have finished an which are still unfinished.
- Once transaction finishes the execution and committed entirely, it is detached from the log file. This log file is nothing bm a circular buffer. In circular buffer writing takes place from start to end and when reaches at other end it then continues at the beginning, overwriting older values as it goes.
- After the crash of the system, there can be no or any number of transactions in log file. If the transactions present currently in log file were not completed to the file system, although operating system has committed it, they must now be completed. The transactions can begin execution from the pointer until the work is finished so that the file-system structures stays in consistent state.
- If transaction does not committed prior to system crash and aborted then Any changes chat were performed to the file system must be undone. So the file system will remain in original state and consistency will be maintained. The logging on disk metadata updates are performed faster than with compare to applied straight to the on-disk data structures.

Syllabus Topic : NFS

5.2,8 NFS

NFS is developed by Sun Microsystems and it is used on Linux to join file systems of different computers into one logical whole. Version 3 of NFS was introduced in 1994 NSFV4 war introduced in 2000 and offers a number of improvements over the previous NFS architecture.

5.2.8(A) NFS Architecture

Q. Explain operation of network system in detail.

- NFS allows a number of clients and servers to share a common file system. These clients and servers can b! on the same LAN or in wide area network if served 'ocated geographically at long distance every machine to be both a el permi simultaneously.
- Remote clients accesses the NFS taw. directories. Every NFS serv servers exported er exports one or more of its directories. These directories then accessed by remow clients. A particular directories along with 'U subdirectories, so in fact entire directory usually exported as a un:t
- The directories list that server exports is kept in a file, often */etc/exports*. As a result of this, these directories automatically whenever the
- ed. Clients can access exported them. When a client mounts a remote NFS its directory hierarchy and client can access mounted directory.

5.2.8(B) NFS Protocols

- As in the network, client and server may have diff operating systems on different hardware, envi is heterogeneous. It becomes necessary thai interface between the clients and servers be defined.
- Then and then it is possible for everyone to write a neiu client implementation and guarantee it to work correctly with existing servers, and vice versa. Up* achieve this objective by defining two client-server protocols. The mounting is handled by **first** NFS protocol.
- A client can send a path name to a server and request permission to mount that directory somewhere in i directory hierarchy. The location at which it is to bt mounted is not specified in message from client to server.
- The reason is. server do not care for this oi mounting. If the path name is legal and the director specified has been exported, the server returns a fife **handle** to the client. The file handle includes fields exclusively identifying the file-system type the disk the i-node number of the directory, and secum information.
- Succeeding calls to read and write vnes m the mounted di ectory or any of lls subdirectories use the file handle. i e or directory replication is not supported by NFS,so user may arrange for all the file systems to be the same.
- As a result, auto mounting is preferred for read-only file systems with system binaries and other files that hardly ever change.
- For di of nfc C, 01 an d access purpose, **second** protocol is used. Clients can send messages to servers to manipu ate dtrectories and read and write files. TM also access Z f« e atlribUteS , such as file mode, size, UNIX I aS , m<x '*lcat *on - Protection mechanism of un 'a also used in NFS.

5.2.8(C) NFS Implementation

a- Explain the layered structure of NFS.

Fig- 5 2 15 shows NFS layer is the system-caij layer ... in which «k» °pen - rcad ' " d close. Th ? is ''Yer fa, File System (V « "ten ittX d C"? calls and inspect' the parameter. aft 'r Pars|„ Vi '"m The VFS layer maintains a m., * "* Ca 'l open elong with a W ' kee P one , enBy for open file. V-node5 1- , '"N fa, local or remote. In case of le H whettl .. V"»de eeded is different and is supp,'!'" fil «, inCf 115 « them. For local files . »'"<• to be Ration ltEO rded since modem Linu, 7 stem and i-nJ ccss (Dultiple file systems. Syste ms Ca| e are StJ Pport I

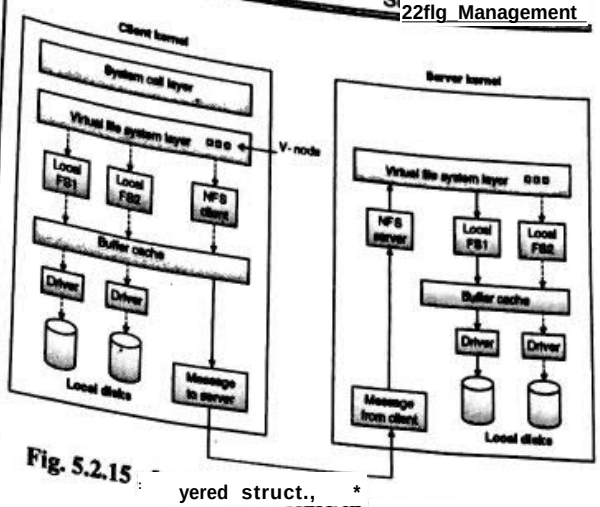


Fig. 5.2.15 : yered struct., * (NFS) ""ww* hie system

V-node plays the important rot ■ sider system cafts io a » ">d as an exam e read. The *mount* program "tount. open and *administrator* for /elc /rc) is called by the sy stem directmy, the local director on whic " " " to be mounted, and other information.

, The *mount* program parses th » » directory to be mounted and find ou[. "* re mote NFS server on which the remote directo "'"'e of the then contacts that machine, asking f '* located - It the remote directory. or a file handle for , ff the directory exists and is available for rem , mounting, the server returns a file h. ...e ...ndle for the directory. Finally, it makes a mount system call, passing the handle to the kernel.

- Ute kernel then constructs a v-node for the remole f directoiy and asks the NFS client code to create an r-node (remote i-node) in its internal tables to hold the file handle. The v-node points to the mode.
- Each v-node in the VFS layer *will* ultimately contain cither a pointer to an r-node in (he NFS client code, or a pointer to an i-node in one of the local file systems Ttos. from the v-node it is possible to see if a file or tetory is local or remote.
- If it is local, (he correct file system and i-node can be loc ted. If h is remote, the remote host and file handle can te located.

Syllabus Topic : Secondary Storage Structure

5.3 Secondary SlQ-ngeStructure

The second- ■"Westlevelof " d ,en ,ar > slora «' structures are the ...ystem.

Topic : Overview of Mass-Storage ..... structure

5 ,3 ,1 on 6rviewof **Mass-Storage Structure**

This section ... "d tertiary storag'devkes PhyS,CIU structure o f sc « ndar >'

5-3.1(A) Magnetlc Disks

L-51 Ex P*ain physkai uaure of magnetic disk.

The magnetic disk is used as main storage device in your computer system, ft is magnetic type of storage device. Within one magnetic disk unit, many physical disks are present. Each disk is known as a platter. Several metal platters are present within the magnetic disk and these are coated with a special magnetic material. The platters rotate many thousands of times within a second for accessing the data. The diameters of platters are usually between 1.8 to 5.25 inches. Magnetic read and write head are present which floats just above the surface of the platter.

Material used for making the platters is Aluminum, glass, or ceramic and two read/write heads are present, one for upper and lower surface. Platters are arranged in stack because of which the position of the read/write heads often is referred to by its **cylinder.** Cylinder is

the location of a single ... X ... disk cache mechan ... write to ... magnetic disk. Mag ... each partition ... capacity of ... The spinning ... disk is n* use. ... to 200 notations for ... data flow between dK ... as The transfer rate ... the time to move the disk arm to the ... (seek time) , and the time for the ... rotate to the disk head (rotational latency).

disk can have ... ittencies of several milhsecon ... everythin cushion of am ... Sometimes it is also possible th ... magnetic surface *call* ed ... crash, the entire disk must tie ... cannot be repaired.

- A removable disk permits different ... mounted if needed. Removable ... contain one platter, covered in a plastic cas ... damage while not in the disk drive. Floppy ... inexpensive removable magnetic disks that ave a soft plastic case containing a flexible platter. The head of a floppy-disk drive generally sits directly on the dis surface, so the drive is designed to rotate more slowly than a hard-disk drive

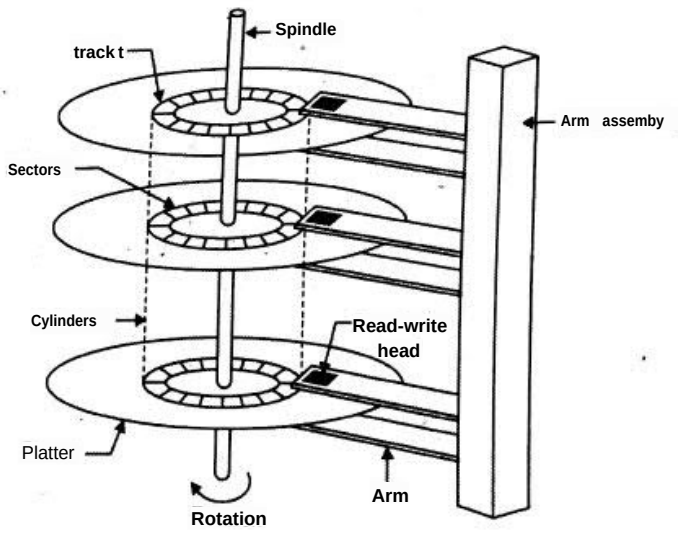


Fig. 53.1: Moving-head disk mechanism

- The storage capacity of floppy disk is only 1.44MB. It is a small plastic plate coated with magnetic material. Data is recorded in floppy disk in terms of in magnetic spots. We can transfer the data from one computer to

... disk. Floppy disk drive is used to ... Other using n ... floppy disk. Its s... read and ... floppy disk. The di... which is ... is 3W inch. ... this ... used ro cover it. A metal ... Hardplaa ... over the ... Read /Write window. ... used to c... ... gets ... me disk is ... a ... floppy ... drive. A notch is used to write ... disk. ... /Opacity Of 1.2 mb. It is a 5¼ inch ... fastic plate coated with magnetic ...

5.3.1(B) Magnetic Tapes

Q. ... physical structure ot maanetic tape,.

- Magnetic tape is the example of an early secondary-... medium. ... storage ... and have large ... capacity, ... is slow than that of RAM and magnetic disk. Magnetic tape is thousand times slower for random access compared to random access to disk. So disk is more preferred secondary storage with compare to tape.
- Primarily, the main use of tape is for taking backup and for storing the infrequently used information. It is also used to transfer information from one system to another. A tape is kept in a spool and is wound or rewound past a read-write head.
- For moving to the desired position, tape may take time in minutes. But once located, the write operation is carried out with speed equal to speed of disk drive. Tape capacities are decided by particular type of tape drive. Usually, tape storage capacity ranges between 20GB to 200GB. The storage capacity is double for some tapes which are having built-in compression.

Syllabus Topic : Disk Structure

5.3.2 Disk Structure

- Modem disk drives are viewed as large ooe-dimensional arrays of logical blocks. This logical block is the smallest unit of transfer. Logical block size is usually 5 12 bytes. The **low-level formatted disks** can have a different logical block size, for example 1,024 bytes.
- The mapping of one-dimensional array of logical blocks is carried out onto the sectors of the disk in sequential manner. On the outermost cylinder, in first

track, the very first sector is sector 0. The mapping is carried out in order through this track, then through the remaining tracks in that cylinder, and then through the remaining the cylinders from ...

fW conversion of logical ... address ... preorcdcally mto disk ... ppiriber, track nurnbor ... sector ... 'hat ... easy IO carry out thia u- ... practically ...

Most of the disks contain ... me mapping hides this ... Mty scotors from elsewhere on ... but spare

The ntmber of sectors per ... is not a const ant on some dnves.

Syllabus Topic : ...

5.3.3 Disk Attachment

On small systems, disk storage ... accessed by ...ers via VO ports called as host-attached storage. In a ... file system this access is via ... remote host called ... network attached storage.

5.3.3(A) Host-Attached Storage

Q. Explain host-attached storage.

- Local VO ports are used to access host-attached storage. Different technology is used by ports. Desktop PC makes use of an VQ bus architecture referred as IDE or ATA. Maximum of two drives per 1/0 bus is supported by this architecture. In SATA protocol the cabling is simplified, SCSI and fiber channel (FC) is used by high ended servers and workstations.
- SCSI bus architecture has its physical medium as a ribbon cable that contains 50 or 68 conductors. There can be at the most 16 devices supported by SCSI on the bus. In general, the devices contain one controller card in the host called as SCSI initiator. Devices also includes up to 15 storage devices called as the SCSI targets.
- The common SCSI target is SCSI disk. The protocol offers the capacity to address up to 8 logical units in each SCSI target. The logical unit addressing is directs ... RAID arcay or ... ite commands to components or a components of a removable media library
- FC ... serial architecture ... optical fiber ... has two ... large switched fabric ... PPCr cable. It ... expected to dominate in the ... address space. This alternative is storage-are... networking ... th®1>uis ot

5.3.3(B) Network-Attached Storage (NAS)

Q. Explain network-attached storage.

- A network-attached storage ... data network. ... interface ... UDP ... networks Performing RPC over ...
- The network-attached ... implemented as a RAID ... unit ... generally ... protocol that ... NAS is a storage-access ... NAS offers ... TCP/IP.
- NAS ... to share a ... machines ... access similar ... storage with simple naming and ... provided ... having ... on the other hand, it is less efficient and storage option ... than ... direct-attached
- ISCSI is the most recent network-attached storage protocol ... uses the ... network protocol ...

5.3.3(C) Storage-Area Network (SAN)

- One limitation of NAS systems is that the storage I/O operations require more bandwidth on the data network. Therefore, latency of network communication increases. This problem can be sensitive in case of large client-server installations where the communication between servers and clients needs more bandwidth along with bandwidth needed for communication among servers and storage devices. Hence, there is competition for bandwidth during communication which can lead to decrease in performance.
- Private network is the example of SAN. It uses storage protoco) to connect server and storage unit instead of networking protocol. SAN is very much flexible. Several systems and storage arrays can be attached to SAN and dynamic allocation of storage can also be done to these machines.
- A SAN switch permits or restricts access between the ... and the storage. Consider the example, If host ... tly have low disk space then SAN can be configured ... host ...

SAN',. i' * possible for clusters of servers to sh tbe same storage and for storage arrays to comprise many direct host connections. SANs usually have several porw. and less cosdy pon's, than storage arrays.

SyftobU* Topic : Disk Scheduling

5 3 4 Disk Schedul ng → (Nov. 1®)

0 . Explain techno of dtaksebe duling. v. 2015. 10 Marks

Firstly, we have to consider, how long it takes to read of write a disk block. The ume required is determined by considering three factors:

1. **Seek time :** It is - .ime to move to - lo -be proper
2. ZX-I detay " roc for the proper sec,or tO rotate under the head.
3. Actual data transfer time.

- For most of lhe disks, the seek time is greater than rotational delay and actual data transfer time.
- Hence by minimizing mean seek time; it is possible to enhance system performance significantly.
- If the processing of requests by disk driver is in First-Come, First-Served (FCFS) basis then it is not possible to optimize seek time.
- Yet, another approach is possible in case of heavily loaded disk. When the arm is seeking for one request, other disk requests may be produced by other processes.
- Several disk drivers keep a table which is indexed by cylinder number. In this table, all the pending requests for each cylinder chained together in a linked list headed by the table entries.

5.3.4(A) FCFS Scheduling Algorithm

→ (Dec. 16)

Q. Compare the following Disk scheduling algorithms using appropriate example-SSTF, FCFS, SCAN, C-SCAN, LOOK. MU - Dec. 2016. 10 Marks

- The fust-come, first-served (FCFS) algorithm is the simplest scheduling algorithm.
- This algorithm is basically fair Tm. . . algorithm general doesnot offer the fastest service, "

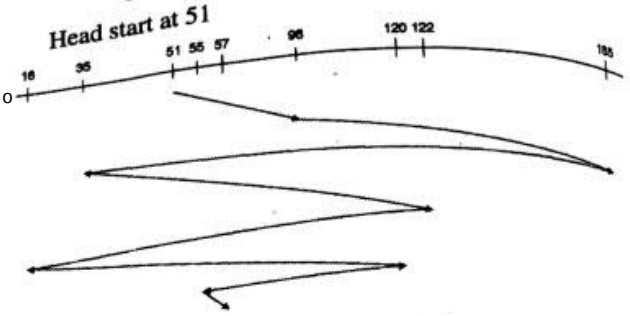
R queue with requests for I/O to blocks

- Consider

o yim 1 22, 16, 120, 55, 57

96, is'- 35,1 = 96, 185, 35, 122, 16, 120, 55, 57

nue« e



Fig, 53.2 : FCFS disk Scheduling

- Initially, the disk head is at cylinder 51. It will then first move fro[m] 5Ho9Mhentol 35,m,U.no. s5, and a' last to 57. for a total head movement of 648 cylinoeia. This schedule is diagrammed in above Fig. 5.3.2.
- The larger swing from 122 to 16 and then back to i20 demonstrates the problem with this schedule. If the requests for cylinders 35 and 16 could be serviced together, before or after the requests at 122 and 120, the total head movement could be decreased considerably, and performance could be thereby enhanced.

5.3.4(B) Shortest-Seek-Time-First (SSTF) Scheduling Algorithm

→ (Dec. 16)

Q. Compare the following Disk scheduling algorithms using appropriate example-SSTF, FCFS, SCAN, C-SCAN, LOOK. MU - Dec 2016. 10 Marks

- This algorithm services all the requests near to the cunent head position before moving the head far away to service other requests. The SSTF algorithm chooses the request with the minimum seek time from the current head position. As seek time increases with the number of cylinders pass through by the head, SSTF chooses the awaiting request closest to the current head position.
- In our example, closest request to the original bead position (51) is at cylinder 55. Once we are at cylinder 55, the next closest request is at cylinder 57. F[fOtn] there, the request at cylinder 35 is nearer than the ok" 96, 50 35 is served next. Continuing, we service U* request at cylinder 16, then 96,120, 122, and finally 185,

This scheduling method results in head movement of only 3,6 ■»**-. . '7 ;nis algorithm g-s a consider fa, * - * performance.

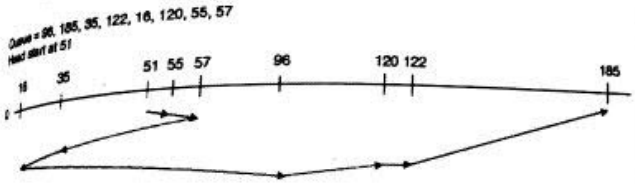
Just ,ite a shortest-job-f™ (SJF) Kh edulin- may ,«d » Ration of wme reques(s Reme £ , a. requests may «»7 « time • Suppose lh „ * e tw° ueStS " «« qUeue - for linden, j6 and and white servicing .he requesl from 16, a new request near >6 arrives.

.-st-'-:™_-.

serviced, another request close io 16 could arrive, jn theory a continual stream of requests near one another could arrive. causing the request for cylinder)8S lo wait indefinitely-

Que„ee96. 185,35. 122, 16. 120,55,57

Head start at 51



Fig, 533 : SSTF disk Scheduling

5.3.4(C) SCAN Scheduling Algorithm

-> (Dec. 16)

7 Compare the following Disk scheduling algorithms using appropriate example-SSTF, FCFS, SCAN, C-SCAN, LOOK (MU - Dec. 2016, 10 Marks)

- The movement of the disk arm in this algorithm is from one end of the disk and goes in the direction of lhe other end. While doing so, it service requests as it arrive at each cylinder, until it arrive at the other end of (he disk.

- After reaching at the other end, direction of head move: Wh< 8cl5 reversed, and servicing continues. The head re Petitiyely scans backward and forward across the disk,

- T** arm acts just like an elevator in the building. Inilia Jly j, servjggj all the requests while going up and w'en reverse the direction towards down, it service the requests on this way. Therefore the SCAN algorithm is a lso called as the elevator algorithm.

- l 1 s consider the same example discussed above. Before proceeding with SCAN to schedule the requests on cylinders 96. 185, 35, 122, 16, 120, 55, and 57, the direction of head movement should be known along with the head's present position (51).

- If movement of the disk arm is towards direction of 0, the head first will service 35 and then 16. After reaching at cylinder 0, the arm will reverse and will go toward the other end of the disk, servicing the requests at 55, 57, 96, 120, 122, and 185,

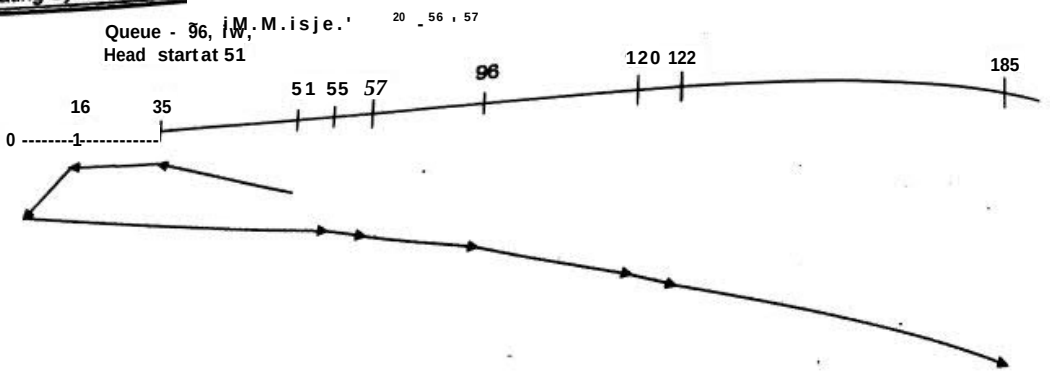
- If the incoming request in queue is just in front of the head, then it will be serviced almost without delay; a incoming request just at the back the head need to wait until the arm goes to the end of the disk, reverses direction, and comes back.

- Considering an even allotment of requests for cylinders, consider the density of requests when the head arrives at one end and" reverses direction. At this moment, comparatively a small number of requests are right away in front of the head, since these cylinders have just been serviced.

The density of requests is very heavy is at the other end of the disk. These requests have also waited for the longer period of time, so why not go there first? That is the consideration of the next algorithm.

fee =96, J85, 35, 122. 16, 120. 55. 57

Head stan at 51



FJg. 53*4: SCAN disk Scheduling

5.3.4(D) C-SCAN Scheduling Algorithm

Q. Compare the following Disk scheduling algorithms using appropriate example-SSTF, FCFS, SCAN, C-SCAN, LOOK. ■* (fee. 16) MU - Dec 2016.7b" arks

- C-SCAN is called as Circular SCAN scheduling and it is a variation of SCAN. It offers a more uniform wait time Just like the SCAN, C-SCAN moves the head from one end of the disk to the other, servicing requests along the way.
- When the head arrive at the other end, however, it straight away returns to the beginning of the disk, without service any requests on the return trip.
- The C-SCAN scheduling algorithm essentially treats the cylinders as a circular list that wraps around fro k* final cylinder to the first one.

Queue= 96, 185, 35, 122, 16, 120, 55, 57

Head start at 51

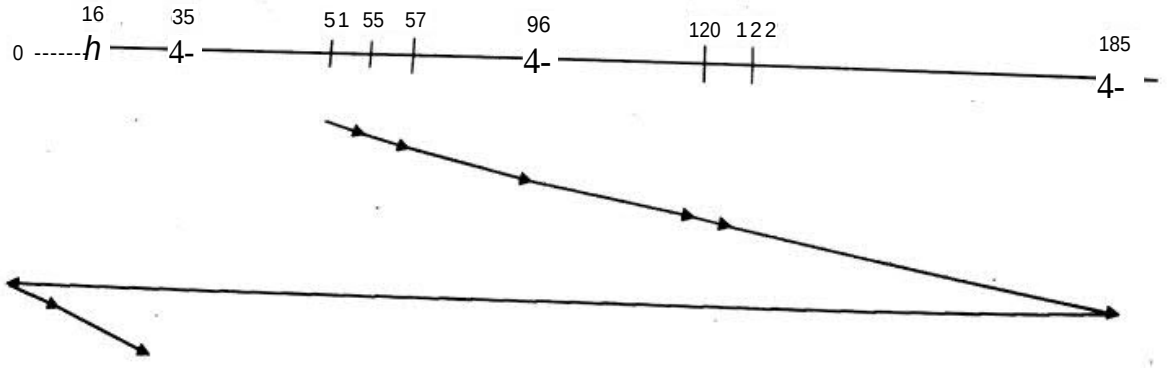


Fig. 5.3.5 : C-SCAN disk Scheduling

5.3.4(E) LOOK Scheduling Algorithm

Q. Compare the f0ll0win g Disf(scheduling algorithms → (Dec. 16)

US ,n9 apprapria,e example-SSTF, FCFS. SCAN, C-SCAN

~ Abo aJgoritbmsscAN and v AN, move the disk TM across ,he *, w ,dth „f d ik . MU - Dec. 2016. 10 Marks

Normally, the arm goes only as far as the final request in each direction. Then, it reverses direction straight away, without going all the way to the end of the disk.

Versions of SCAN and C-SCAN that follow this above pattern are called **LOOK** and **C-LOOK scheduling,** because they *look* for a request before continuing to move in a given direction.

Queue = 96, 185, 35, 122, 16, 120, 55, 57

Head start at 51

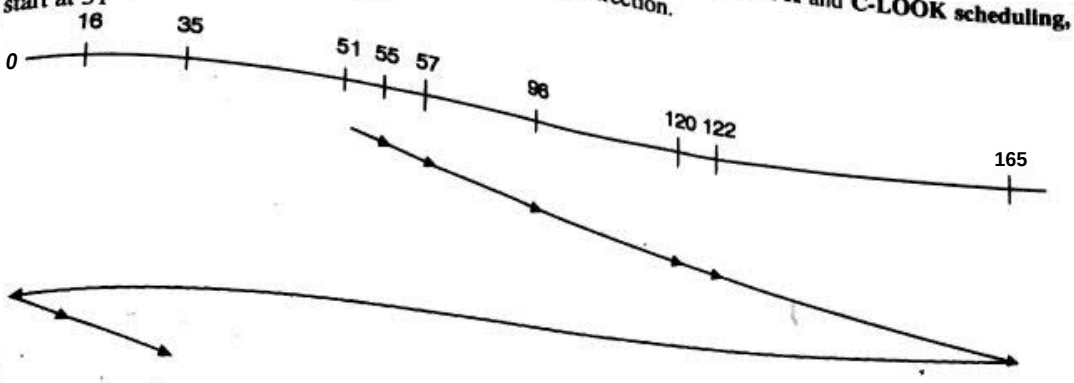


Fig. 5.3.6 : LOOI$_{Cdis}$ **ksched**$_{U}$**|**$_{Ing}$

5.,C Examples on Disk Schedni! edull "9 *Algorithms*

Example 5.3.1

Suppose that a disk drive has 5000 cylinders, numbered 0 to The dri a request at cylinder J and the Previou. reject was at cylinder 125. The queue of pending requests, in FIFO order, is 86 , 147Q $_{f7}$74,S48.1509, 1022, 1750, 130 Starting f$_{rom}$ the . head position, what is the total distance ((in cylinders) that jetfisKarm moves Io satisfy all the pending requests, for each of the following disk scheduling Algorithms.

I, FCFS 2. SCAN

*%rtk)n:

TbeFCES schedule is 143, 86, 1470, 913, 1774, 948, (M1022,1750, 130. The total seek distance is 7081.

He SCAN schedule is 143, 913, 948, 1022, 1470, 1503,1750,1774, 4999, 130, 86. The total seek distance is 9769

---

i»n(* 5.3.2

• requests come in to the disk for cylinders 10, 22, 20, 2, * Sand 38. A seek takes 6 msec per cylinder move. How !<tt Mek time is for Closest cylinder next algorithm? "tyam is at cylinder *20.*

lolution :

fll ***Come, First-Served algorithm** accepts requests t ®**atime** and caries them out in that order,

al time : «20-iO) + (22-10) + (22-20) + (2)

+ (40 - 2) + *(40 -* **+ (38 – 6))**

Cylinder Seek «77' * = 876 ms «

Algorithms:

SSF Shortest Seek Fi$_{rBt}$ **B,BOrit**

closest reque-t h^{m} **always handles the**

°° **Zea S * " "* time, independently**

Tow «lme , « $_{20}$ – 20) + (22 – W)

+ (10 - 6) + (6 - 2) + (38 - *2)* + (40 - 38))

' 6 msec = $_{M}$ **• 6 msec = 360 msec**

nearest requests in $^{A)}$ **8° nlh** ^{TM} **increasiog ■ Elevator** (decreasing) **algorithm handles order until no «« requests on the way. Afterwards the direction is reversed.**

Total time: ((20 - 20) + (22 - 20) + (38 -*22)*

+ (40 - 38) + (40 - 10) + (10 - 6) + (6 - 2))

*** 6 msec = 5 8 * 6 msec = 348 msec**

Example 5.3.3 MU - Dec. 2014. 10 Marks

Suppose that a disk drive has 5000 cylinders, numbered 0 to 4999. The drive is currently serving a request at cylinder 143, and the previous request was at cylinder 125. The queue of pending requests, in FIFO order, is 86, 1470, 913, 1774, 948, 150971022, 1750, 130. Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests, tor each of the following disk-scheduling algorithms?

a. FCFS & ^{SSTF} C, SCAN d L oo K

OSCAN I.C-LOOK

Solution :

a. The FCFS schedule is 143, 86, 1470, 913, 1774, 948, 1509, *1022.1750,* 130.

The total seek distance is 708 1.

b. The SSTF schedule is 143, 130, 86, 913, 948, 1022, 1470, *15O9, 1750,1774.*

The total seek distance is i !•••>■ 1022, .470,

c. The SCAN schedule is 143, 913, *948, 1509, 1750. 1774,4999,* 130. 86.

The total seek distance is 9769. 1470,

d. The LOOK schedule is 143, 913, 948, 1022, 1509, 1750. *1774,130,86.*

The total seek distance is 3319.

e. The C-SCAN schedule is 143, 913, 948, 1022, 14 , 1509, 1750, 1774,4999, 86, 130.

The total seek distance is 98 13,

f. (Bonus) The C-LOOK schedule is 143, 913, 948, 1022, 1470, 1509,1750, 1774,86, 130.

The total seek distance is *3363.*

Example 5.3.4

None of the disk-scheduling disciplines, except FCFS, is truly fair (starvation may occur).

a. Explain why this assertion is true.

b. Describe a way to modify algorithms such as SCAN to ensure fairness.

c. Explain why fairness is an important goal in a time -sharing system.

d. Give three or more examples of circumstances in which it is important that the operating system be unfair in servig I/O requests.

Solution :

a. New requests for the track over which the head currently resides can theoreticafJy arrive as quickly as these requests are being serviced.

b. All requests older than some predetermined age should be compulsorily placed the top of the queue, and an associated bit for each could be set to specify that no new request could be moved ahead of these requests. For SSTF, the rest of the queue would have to be reorganized with respect to the last of these "old" requests.

c. To avoid unusually Jong response times

d. User requests should have lower priority over paging and swapping. It may be desirable for other kernel tmnated I/O, such as the writing of fife sX metadata, to take priority over user I/O If the S supports real-time process priorities, the VO reque 7™ M u f those processes should be favoured.

Example 5.3.5

Disk requests come into the disk for cylinders 10, 22, 20 , 2 , 40, 6 and 38. A seek takes 6 millisecond per cylinder move. How much seek time is for Closest cylinder next algorithm? Initially arm is at cylinder 20.

solution ■ |rgt , **Served algorithm** acce pls r **First** - CorneT and ca ries them out in that order; equests

On " " * T Total ume. ((20 - 10) + (22 - 10) + (22 - 2) + (40-6) + W .A + (40 _ 2)

__]46*6msec = 876 msec. msec

Cylinder Seek Algorithms: SSF Shortest Seek Firs handles the closest request

" l8< 'mX«k ,i,nc ' independently on the request, *>

Total stae : ((20 + (6 - 20)+ (38 - 20) + (22 - 38)) . 6 ' 6) msec

= 60 * 6 msec = 360 msec.

Cylinder Seek Algorithms : Elevator algorithm h™, nearest requests in increasing (decreasing) ot(3n until no more requests on the way. Afterwards the direction is reversed.

Total time : ((20 —20) + (22 - 20) + (38 —22) + (4q_38) + (40 - 10) + (10 —6) + (6-2))* 6 a isec

= 58 * 6 msec = 348 msec.

Example 5.3.6

Suppose that a disk drive has 5000 cylinders, numbered 0 to 4999. The drive is currently serving a request at cylinder 155 and the previous request was at cylinder 170. The queue of pending requests is 86, 1350, 948, 130, 1500, 50. Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests, for each of the following disk scheduling Algorithms.

1. FCFS 2. $STF 3. SCAN 4. C-SC AN
5. LOOK 6. C-LOOK

Solution :

1. FCFS : The FCFS schedule is 155,86,l350M 130,1500,50

It gives (155 - 86) = 69, (1350 - 86)

= 1264, (1350 - 948) = 402,

(948- 130) = 818, (1500- 130)= 1370,(1500-50) = 1450

Total head movements are :

69 + 1264 + 4oq j oio . , 1370 + 1450

= 5373 Cylinders

2. SSTF * The SSTF schedule is 155,130, 6, 50, **948,1350,1500** '

(155 – 130) = 25, (130 – 86) = 44, (86 – 50)

(948 – 507= 898.(1350- 948) = 402, (1500

= 150 ~ 1350)

Total head movements are :

25 + 44 + 36 + 898 + 402 + 150 = 1555 Cylinders

SCAN : The SCAN schedule is 155,130,86, 50,0, 948,1350,1500

It gives (155 - 130) = 25,(130 - 86) = 44, (86 - 50)

= 36-(50 - 0) = 50

(948 - 0) = 948,(1350 - 948) = 402,

(1500-1350) = 150

Total head movements are :

25 + 44 + 36 + 50 + 948 + 402 + 150 = 1655 Cylinders

C - SCAN : The C-SCAN schedule is 155, 130, 86, 50,0,4999, 1500, 1350.948.

It gives (155 —130) = 25, (130 - 86) = 44, (86 – 50) = 36,

(50 - 0) = 50, (4999 - 0) = 4999, (4999 - 1500) = 3499,

(1500- 1350) = 150, (1350-948) = 402.

Total head movements are :

25 + 44 + 36 + 50 + 4999 + 3499 + 150 + 402 = 9205 Cylinders

C-LOOK : The C - LOOK schedule is 155,130,86, 50,1500, 1350,948

It gives (155 - 130) = 25, (130 - 86) = 44,(86 - 50)

= 36,(1500 - 50) = 1450,

(1500- 1350) = 150 (1350- 948) = 402.

Total head movements are :

25 + 44 + 36 + 1450 + 150 + 402 = 2107 Cylinders

6. **LOOK : The LOOK schedule is 155,130,86, 50 , 948, 1350,1500**

It gives (155 - 130) = 25. (130 - 86) = 44, (86 - 50) — 36,

(948 - 50) = 898. (1350 - 948) = 402.

(1500-1350) = 150.

Total head movements are :

25 + 44 + 36 + 898 + 402 + 150 = 1555 Cylinders

Ex. 5.3.7 Suppose that a disk drive has 5000 cylinders numbered 0 to 4999. The drive is currently serving a request at cylinder

143, and the previous request was at cylinder 125. The queue of pending requests in FIFO is ordered as 80, 1470, 913, 1777, 948, 1022, 1750,130. What is the total distance that the disk arm moves for following

1. FCFS 2. SSTF 3. LOOK 4. SCAN 5. C-SCAN

1. **FCFS : The FCFS schedule is**

143, 80, 1470, 913, 1777, 948, 1022, 1750, 130

It gives (143 - 80) = 63. (1470 - 80) = 1390,

(1470 - 913) = 557,

(1777 - 913) = 864. (1777 - 948) = 829,

(1022 - 948) = 74, (1750 - 1022)

= 728, (1750 - 80) = 1670

Total head movements are :

63 + 1390 + 557 + 864 + 829 + 74 + 728 + 1670

= 6125 Cylinders

2. **SSTF : The SSTF schedule is**

143,130, 80,913,948, 1022, 1470, 1750,1777 .

It gives (143 - 130) = 13, (130 - 80) = 50,

= (913 - 80) = 833,

(948-913) = 35,(1022 - 948) = 74,

(1470 - 1022) = 448,(1750 - 1470) = 280,

(1777 - 1750) = 27

Total head movements are :

13 + 50 + 833 + 35 + 74 + 448 + 280 + 27

= 1760 Cylinders

3. **LOOK:** The LOOK schedule is

143,913,948, 1022, 1470, 1750, 1777,130, 80.

It gives (913 - 143) = 770, (948 - 913)

= 35, (1022 - 948) = 74,

(1470 - 1022) = 448, (1750 - 1470)

= 280,(1777 - 1750) = 27,

(1777-130) = 1647,(130 - 80) = 47

Total head movements are

770 + 35 + 74 + 448 + 280 + 27 + 1647 + 50

= 3331 Cylinders

4. **SCAN:** The SCAN schedule is

143, 913, 948, 1022, 1470, 1750,1777, 4999, 130. 80.

It gives (913 - 143) = 770, (948 - 913) = 35.

(1022 - 948) = 74,

(1470 - 1022) = 448,

(1750 – 1470) = 280, (1777 – 1750) = 27,

(4999-1777) = 3222,(4999 – 130) = 4869

(130-80) = 50

Total head movements are :

770 + 35 + 74 + 448 + 280 + 27 + 3222 + 4869 + 50

= 9775 Cylinders

5. C-SCAN : Tbc C-SCAN scbedule is 4999, 80, 143, 913, 948. >022, 1470. 1750.1777. J30.

It gives (913-143) = 770. (948 —913) = 35,

(1022-948) = 74,

(1470 - 1022) = 448,

(1777- 1750) = 27,(4999-1777) = 3222,

(4999 - 0) = 4999,

(80-0) = 80,(130"8°) = 50

Total head movements are :

770 + 35 + 74 + 448 + 280 + 27 + 3222 + 4999 + 80 + 50 - 9985 Cylinders

Example 5.3.8

Suppose that a disk drive has 200 cylinders, number 0 to 199. The initial head position is at 100th track . The queue of pending requests in FIFO is 55, 58. 39. 18 90, 160. 150, 38, 184. Calculate average seek time tor each of the following algorithm,

1. FCFS 2. SSTF 3. SCAN
4. LOOK 5. C-SCAN 6. C-LOOK

Solution :

1. FCFS : The FCFS schedule is

100, 55, 58, 39, 18, 90, 160, 150, 38, 184.

It gives (100-55) = 45, (58 - 55) = 3,

(58-39) = 19,(39-18) = 21,

(90-18) = 72,

(160-150) = 10,(150-38)= 112,

(184-38) = 146

Total head movements are :

45 + 3 + 19 + 21 + 72 + 10 + 112 + 146

= 428 Cylinders

2. SSTF : The SSTF schedule is

100, 90, 58, 55, 39, 38, 18, 150, 160, 184

It gives (100 - 90) = 10, (90 - 58) = 32,

(58-55) = 3, (55 -39) = 16,

(39-38) = 1,(38-18)= 20,

(150 ~18) = 132 (160-150)= 10,

(184 – 160) = 24

Total head movements are:

10 + 32 + 3 + 16 + 1 + 20 + 132 + 10 + 24

= 248 Cylinders

3. SCAN : The SCAN schedule is

100, 90, 58, 55, 39, 38, 18, 0, 150, 160, 184

It gives (100 - 90) = 10, (90 - 58) = 32,

(58 - 55) = 3. (55-39) =16.

(39 _38) = 1,(38-18)= 20,

(18 -0) = 18, (150-0) = 150,

(160-130) = 10,(184-160) = 24

Total head movements are:

10 + 32 + 3 + 16 + 1 + 20 + 18 + 150 + 10 + 24

= 284 Cylinders

4. LOOK- The LOOK schedule is 100. 90, 58, 55. 39. 38. 18, 150, 160, 184

It gives (100 - 90) = 10.

(90 - 58) = 32,

(58-55) = 3. (55 - 39) = 16,

(39 - 38) = 1 ,(38-18)= 20,

(150- 18) = 132, (160- 150) = 10,

(184-160) = 24

Total head movements are :

10 + 32 + 3 + 16 + 1 + 20 + 132 + 10 + 24

= 248 Cylinders

5. C • SCAN : The C - SCAN schedule is

100, 150, 160, 184, 199,0, 18, 38,39,55,58,90

It gives (150- 100) = 50, (160 -150) = 10,

(184-160) = 24,

(199-184) = 15, (199-0)= 199,

(18-0) = 18, (38 - 18) = 20,

(39-38)= 1,(55-39) = 16,

(58 -55) = 3, (90 -58) = 32

Total head movements are :

50+ 10+ 24+ 15+ 199+ 18 + 20+ 1 + 16 +3 +32

—388 Cylinders

®- C-LOOK : The C - SCAN schedule is

100, 150, 160, 184, 18, 38, 39, 55, 58, 90

Itgives (150- 100) = 50,

(160-150) = 10,

(184-160) = 24,

(184-18) = 166, (38 - 18) = 20,
(39-38) = 1, (55 - 39) = 16,
(58-55) = 3, (90 - 58) = 32

Total head movements are :
50 + 10 + ... + 1 + 16 + 3 + 32 = 322 Cylinders

Example 5.3.9

Suppose that the head of a moving-head disk with 200 tracks numbered 0 to 199 is currently serving a request at track 143 and has just finished a request at track 125. The queue of requests is kept in the FIFO order 86, 147, 91, 177, 94, 150, 102, 175, 130

What is the total number of head movements needed to satisfy these requests for the following disk scheduling algorithms ?

1. FCFS 2. SSTF 3. Elevator

Solution :

1. **FCFS : The FCFS schedule is**

143, 86, 147, 91, 177, 94, 150, 102, 175, 130

It gives (143 - 86) = 57, (147 - 86) = 61,
(147-91) = 56,
(177-91) = 86, (177 - 94) = 83,
(150-94) = 56, (150 - 102) = 48
(175-102) = 73, (175 - 130) = 45

Total head movements are :

$57 + 61 + 56 + 86 + 83 + 56 + 48 + 73 + 45$

= 565 Cylinders.

2. ***SSTF* :** The SSTF schedule is 143, 147, 150, 130, 102, 94, 91, 86, 175, 177

It gives (147 - 143) = 4, (150-147) = 3,
(150-130) = 20,
(130-102) = 28, (102 - 94) = 8,
(94-91) = 3
(91 - 86) = 5, (175 - 86) = 89, (177 - 175) = 2

Total head movements are :

4 + 3 + 20 + 28 + 8 + 3 + 5 + 89 + 2 = 162 Cylinders

3. **Elevator : The elevator schedule is**

143, 147, 150, 175, 177, 199, 130, 102, 94, 91, 86

It gives (147 - 143) = 4, (150-147) = 3,
(175-150) = 25,
(177-175) = 2, (199 - 177) = 22
(199-130) = 69.
(130-102) = 28, (102 - 94) = 8

Total head movements are :

4 + 3 + 25 + 2 + 22 + 69 + 28 + 8 + 3 + 5

= 169 Cylinders.

Example 5.3.10 MU - May 2016, 10 Marks

Assume that the disk head is initially positioned over track 100. For the disk space requests 41, 10, 64, 186, 27, 129, 110, 120, 147. Compute SSTF, C-SCAN, C-LOOK scheduling. It carried out the average seek time. Show the tracing of the requests.

Solution :

SSTF :

100, 110, 120, 129, 147, 186, 64, 41, 27, 10

(110 - 100) + (120-110) + (129-120) + (147-129)
+ (186-147) + (186-64) + (64-41) +
(41-27) + (27-10) = 10+10+9+18+39+122+23+14+17

= 262/9 = Average 29.11

C'SCAN :

100, 64, 41, 27, 10, 186, 147, 129, 120, 110

(100 - 64) + (64 - 41) + (41-27) + (27-10) + (186-10)
+ (186-147) + (147-129) + (129-120) + (120-110)
= 36+23+14+17+176+39+18+9+10 = 342/9 = Average 38

C-LOOK :

100, 110, 120, 129, 147, 186, 10, 27, 41, 64

(110 - 100) + (120-110) + (129-120) + (147-129) +
(186-147) + (186-10) + (27-10) + (41-27) + (64-41)
= 10+10+9+18+39+176+17+14+23 = 316/9

= **Average 35.11**

Example 5-3.11 MU - June 2015, 10 Marks

The requested tracks in the order received are 54, 57, 40, 20, 80, 120, 150, 45, 180. Apply the following disk scheduling algorithms. Starting track is 90.

(I) FCFS (ii) SSTF (iii) C-SCAN

Solution :

(i) **FCFS :** FCFS schedule is 90, 54, 57, 40, 20, 80, 120, 150, 45, 180

It gives (90-54) + (57-54) + (57-40) + (40-20) + (80-20) + (120-80) + (150-120) + (150-45) + (180-45)

Total head movements are
36+3+17+20+60+40+30+105+135 - 343

(ii) **SSTF :** SSTF schedule is 90, 80, 57, 54, 45, 40, 20, 120, 150, 180

It gives (90-80) + (80-57) + (57-54) + (54-45) + (45-40) + (40-20) + (120-20) + (150-120) + (180-150)

Total movements are
9+5+20+ 100+.1O+.30* 23 0
10+23+3+

(ii) C-SCAN ,.«0. 37. 54. 43. 40. 20. 0.120.
C-SCAN schedule is 9C
150, 180
It gives (90-80) + (80-57) + (57-54) + (54-45)
+ (45-40) + (40-20) + (20-0)+(120-0) + (150-120)
+ (180-150)
Total head movements are
10+23+3+9+5+20+20+120+30+30 – 27

Example 5.3.12 MU - Dec- 20lb~ioM*±s
On a disk with 1000 cylinders numbered 0 to 999 compute tn. number or *trad".* »> disk arm must move to satisfy all in. requbsls >n *th* *q* ueue. Assume the *lae , r6que3t* nd the haad ,g moving toward service at track , Z queue in F'FO order contain, requests tor the •• »«, 692. 475. 105. 376 .

Perform the computation for the following scheduling algorithms.

j) FIFO S) SSTF Wb> SCAN

Solution :

i) FIFO : The FIFO schedi lle is 345,123,874,692,475,105 and 376.

Total head movement = (345-123) + (874-J23) + (874-692) + (692-475) + (475-105) + (376-105) = 2013

if) SSTF : The SSTF schedule is 345,376,475, 692,874, 123 and 105

Total head movement = (376-345) + (475-376)

(692-475) + (874-692) + (874-123) + (123-105) =1298.

(iii) SCAN : 345,123, 105, 0, 376, 475, 692 and 874

Total head movement = (345- 123) + (123-105)
+ (105-0) + (0-376) 4-*(376-475)* + (475-692)
+(692-874)= 1219.

Syllabus Topic : Disk Management

5.3.6 Disk Management

The disk management activities of the operating system include disk initialization, booting from disk, and bad-block recovery.

5.16(A) Disk Formatting

If the magnetic disk is new then « is - a P' a «- of a magnenc recording raaterial . Directly on such surface we cannot store data. The surface needs to be divided m seem so that disk controller is able to perform reaJ rcad and write operations.

- The procedure of doing this is called low level or physical formatting. This formatting fills each secto, with data structures :
  - Header
  - A data area (usually 512 bytes in size)
  - Trailer
- The header and trailer hold the information used by disk controller - such as error correcting codes ECC and sector r number. ECC is updated with a value calculated from all the bytes in the data. This is dOne during the coinuollcr writes a sector of data at the time of normal I/O, die area-
- The ECC is recalculated at the time sector is read and compared with the stored value. If lhe stored and calculated numbers matches data is correct and corrupted. If mismatch occurs, it indicates that tu ** area of the sector has become corrupted w that the disk sector may be bad.
- The ECC contains sufficient information, if few data have been corrupted, it facilitate the control recognize which bits have changed and calculate what their correct values should be. It then reports recoverable soft error. The controller autonomously does the ECC processing whenever a sector is read or written.
- During the manufacturing process, low level formatting of hard disk is carried out. It helps to manufacturing initialize the mapping from logical block numbed defect-free sectors on the disk.
- For several hard disks, the disk manufacturer is instructed for how many bytes of data space to leave between the header and trailer of all sectors at the time when n perform low level formatting. For example, the sizes can be 256,512, and 1,024 bytes. If disk is formatted with larger sector size, fewer sectors can fit on each track.
- As a result fewer headers and trailers are written on each track and more space is obtainable for user data.
- Some operating systems can handle a sector size of 512 bytes. Operating system keeps its own data structures on disk before it use disk to store the files. It performs this with following two steps :
  - Il partitions the disk into one or more groups oi cylinders. Each partition is treated by OS an separate disk.
  - Logical formatting. That means creation of file system.
- In order to increase the efficiency, file system gr* blocks in chunks called as clusters.

fflz operaung systems give spccia| 'Ji programs the ability to uw a disk partition a, a lar ge sequential array of logical "»* s - without any file-system structfU ThJ5 's somcl 'rnes called m da >| i/o to this is caJIcd 33 ra * I/O.

< e oot Block

S-3' bootstrap program is required f w a .tTthe boot g after it is powered up or J tcr * ifl|ha it«es 4,1 coni p° nenUi Of the system CPlJ regist -s w device controller* and the com memo-y and then starts the operating o f ,nain program ,ben ,ocates 'be OS kernel The *''*£ tW t kerne! into memoiy, and ju mps to " d,sk - addre0 <> start *he <, P era,in 8- tem execution.

The Read Only Memory (ROM) does not require initialization and is at a fixed location that the processor can begin executing when powered up or Therefore bootstrap is stored in ROM

* use Of read only feature of ROM; it CMnot Lification » a computer vtnts. diffk Uifes chan 8 8 ROM hardware chips.

Lafore. most s >6,e,ns slore a 'Mil bootstrap loader in the bOOt ROM Wh,Ch inVOkcs a,ld bri "g full Ltstrap program from disk into main memory. The Ldified version of full bootstrap program can be simpJy written onto thc disk .

Tbc fixed storage location of full bootstrap program is in the "boot blocks", A disk that has a boot partition is called a boot disk or system disk , Boot ROM code gives instruction to controller for reading the boot Hocks into memory (no device drivers are loaded at this point) and then starts executing that code.

pje full bootstrap program is more sophisticated than [fie bootstrap loader in the boot ROM; it is able to load the complete operating system from a non-fixed location on disk and to start the operating system execution. Yet, (he full bootstrap code may be small.

5.3.6(C)Bad Blocks

Failureof the disk can be :

- Complete, means there is no way other than replacing thedisk. Back up of content must be taken on new disk.
- Oneor more sectors become faulty.
- Aftermanufacturing, (he bad blocks exist. Depending on the disk and controller in use, these blocks are handledin a different ways.

If the disks are with IDE controllers then these are wple disks. In these disks bad blocks are handled "snuafly.For example, the MS-DOS format comman carry out logical fonnatling md. as a part of the process

scans tbe dkk (n discover had blockv If format discovers a bad bi<Kk, '* whies a special value into thc related I A tnat blodk .. blocks go bad during normal operation, a >peel a P gram (such „ chkdsk} must be cun msnuidly to look for |C, ad thcbad blocks and to lock them away. Data that resided on bloeks generally are lost.

Tbc 5051 disks are used in high-end PCs and most stations and servers, are more intelligent about recovery of bad-block. Thc controller preserves a record of ad blocks on the disk. The list is initialized during the low-®vc l formatting at die factory and is updated over the life of die d ,sk ' Low-level formatting also sets aside spare sectors not vk 'ble to the operating system. The controller can be fold to replace each bad sector logically with one of the spare sectors. This is called as sector sparing or forwarding.

A usual dealing with bad-sector is like follows :

- The OS attempt to read Logical block 67.
- The controller computes the ECC and concludes that the sector is bad. The same result is reported to OS.
- When the system is rebooted, a special command is run to inform the SCSI controller to replace the bad sector with a spare.
- Afterward, at whatever time the system requests logical block 67, the request is translated into the replacement sector's address by the controller.

Syllabus Topic : RAID Structure

5.3,7 RAID Structure

5,3.7(A)RAID Levels

Q. Explain various RAID levels.

- The performance of CPU has been rising immensely over the past decade,, approximately it is getting increasing by 100 percent after every 1.5 years. On the contrary disk performance is not increasing with this speed. Now days average seek time of the disk is near about 10 msec, which is very less compare to average seek time of 50 to 100 msec on minicomputers in the decade of 1970
- If in any other industry, performance is enhanced by 5 to 10 percent in two decades, it would be considered as great achievement. However, in the computer industry it is an discomfiture, As a result, the difference between CPU performance and disk performance has grown to be much larger in due course of time.
- In order to get faster processor performance, parallel processing is being used to a greater extent. Over the years people thought that parallel I/O might be an excellent scheme as well. In research paper of 1988, Patterson et al. recommended six specific disk

organizations that could be «j lized tO gC better disk performance, reliability, or both- and has

- This proposal was soon agreed by m ca lled a made possible a new class of l definition of **RAID.** Patterson et al- had give " nsive **Disks,** but RAID as **Redundant Array of Inexp*** „p, as industry soon changed the definition or "Independent" instead of "Inexpensive nf a

– The fundamental thought behind a RA1 J* o is bunch of disks beside the computer, us y of disk farce server, put RAID controller m f and ther carry controller card, copy the data on RAID s hould on usual operation. The intention was a K l P» rinsa appear a single large expensive disk to t e ope roved system but have improved performance and } reliability.

– As SCSI disks are characterized by better periormanc, low cost, and the capability to include up to 7 hit a single controller/ 15 for wide SCSI), it is usu n most RAIDs contain a RAID SCSI controller l addition to a bunch of SCSI disks which operating system can thinks as a single large disk. In this fas on, no software modifications are necessary to make use o the RAID, a great selling point for a majority of system administrators.

– Besides coming out similar to a single disk to the software, every RAID have the feature of distributing the data over the drives, to permit parallel operation. Many varieties of methods for achieving this were defined by Patterson el al., and they are at present recognized as RAID level 0 through RAID level 5. There are only six distinct organizations possible.

– RAID level 0 is demonstrated in Fig. 5.3.7(a). It gives the impression of the virtual single disk simulated by the RAID as being split up into strips of *k* sectors each, by means of sectors 0 to *k* - I being strip 0, sectors *k* to 2£ -1 as strip 1, and so on. Each strip appears to be a sector for value k=l; a strip is two sectors if k = 2, etc. The RAID level 0 collection writes successive strips over the drives in round-robin manner, as shown in Fig. 5.3.7(a) for a RAID having four disk drives.

– When the data is distributed like this on more than one drives, the operation is called as striping. If data block comprises four consecutive strips beginning at strip boundary and software gives a command to read this data block, the RAID controller will split this command up into four separate commands, one for each of the four disks, and have them run in parallel. Thus parallel I/O is achieved and software remains unaware about it.

– If the large numbers of request are present then RAID level 0 works better. If number of drives times the strip size is less than the request, some of the drives win receive mor- than one requests, with the intention that when they comply the first request they initiate the

5€COnd "Teller decides to divide the request up and

– The eon ro priate commands to the appropriate P v|d ' £e right order and then collect the outcome i diskS £ right manner. Performance i memory mentation Is simple. RAlb Nk and the onorlv with operating systems that Q dS O"« sector " a Ume The "suits dem " K but there is no increase in perform £*>« £ ofparalielism.

– Another drawback of this organization is that the probability of failure free operation is potentially poor than having a single large expensive disk (SLED). If a RAID includes four disks, having mean time to failure (MTTF) of 20,000 hours for every disk, about once every 5000 hours a drive will crash and all the d vanished. A single Urge exp 'Oi be e (L E D) with a MTTF of 20.000 hours woum"* ** imes more reliable. Since tedundaocy is absent infoj, design, it is not actually a true RAID. thii

– The next alternative, RAID level L, demonstrated in J,g. 5.3.7(b). is an exact RAID. In this, duplic each disk is present, so total 8 disks are present 2 1 of these 8 disks, four are primary disks and f 0Ur b disks. The strip is written twice for each operation. A read can be earned out on any of the * assigning and distributing the load over more drives. '

– As a result, write performance with compare to sinou drive is worse; however read performance Can V improved neatly by double. Fault tolerance ' outstanding : if a drive fails, the copy is just used in place. Recovery includes simply fixing a new drive and copying the complete backup drive to it.

– The RAID levels 0 and 1 operate with strips of sectors Whereas RAID level 2 supports the working on either word basis or even on a byte basis. Consider division of each byte of the single virtual disk into a two 4-b rt nibbles, then inserting a Hamming code to each one fo build a 7-bit word having bits 1, 2 and 4 parity bits.

– Consider again Fig. 5.3.7(c) in which the seven drives are synchronized with respect to arm position and rotational position. Then it would be possible to write one bit per drive out of the 7-bit Hamming coded word.

– This idea used by the the thinking Machines' CM-1 computer. It considers 32-bit data words and append 6 parity bits to form a 38-bit Hamming word, in addition to one additional bit for word parity, and distribute each word over 39 disk drives.

– The total throughput was enormous, because it was possible to write 32 sectors worth of data in one secioi writing time. The crash of one drive did not lead w pro ems, as loss of a drive cause the loss of 1 M in each 39-bit word read> which Hamming code a»W manage easily.

– I hJ raW k o f the above this a 1 ttie erne is that, in drives needs to be rotaJ Si

...synchronized and ... of drives (regardless ... benefited ...) the overhead is ... of the controller ... checksum every bit ... level 3 ... a simpler version of RAID level 2 ... parity ... should ... because ... words are synchronized ... distributed over ...

several drives. Initially, it might come into view that a single parity bit offers only error detection, not error correction. If the errors are random ... true. But in case of failure of drive ... it offers full 1-bit error correction as the location of the bad bit is recognized. In case of a drive crash, the controller just consider that all its bits are 0s. If parity error occurs for the word, the bit from the crashed drive must have been ... thus it is corrected.

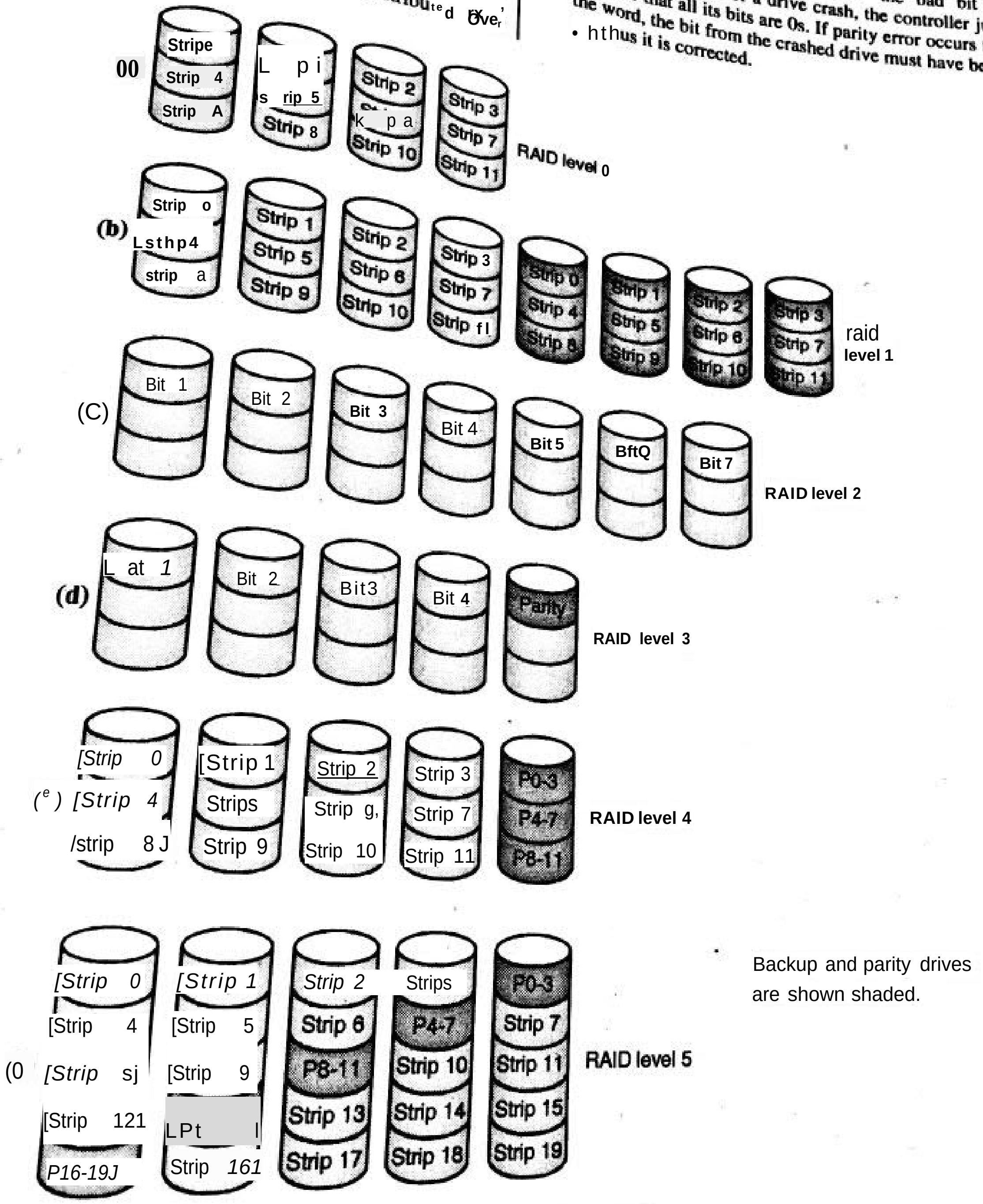


Fig. 5.3-7: RAID levels 0 through 5

... rates, the total separate I/O requests per second

- Despite ... RAID levels 2 to 3 ... they can handle is no better than for a single drive.

– In RAID levels 4 and 5 do not need synchronized drives and again operate ... RAID level 4 consider individual words ... similar to RAID level 3, as demonstrated in Fig. 5.3.7(e). In this, parity written onto an extra drive ... of each strip, and here a strip-for-strip parity is carried out in k bytes. ... the EXCLUSIVE OR of all strips together. This operation results in a parity strip of length k bytes.

– In case of failure of drive, the parity bytes can be recovered by recomputing from the parity drive. This design defends against the failure of single drive, but penned ... for small updates. If only one sector is altered, it is essential to read all drives to facilitate emulation of the parity, which must then be rewritten.

– On the other hand, it may read the old user data and the old parity data and compute again the new parity from them. Still with this optimization, a small update needs to have two reads and two writes.

– As a result of the huge load on the parity drive, it may turn into a bottleneck. This bottleneck is removed in RAID level 5 by spliting the parity bits evenly over all the drives, in round robin manner, as illustrated in Fig. 5.3.7(f). Yet, in case of a failure of drive, rebuilding the contents of the failed drive is a complex process.

Syllabus Topic : Stable-Storage Implementation

5.3.8 Stable-Storage Implementation

– For the write-ahead log, stable storage is required. The information stored in stable storage remains permanent and never lost. In order to implement this stable storage, it is necessary to replicate the information over multiple disks with independent failure modes.

– The update writing should be carried out in a manner that promises that a failure during an update will not damage all the copies. It should also guarantee that during recovering from a failure, all copies must remain consistent and with correct value, though another failure takes place during the recovery.

– One of the following can happen due to disk writes.

o **Successful completion.** The writing of data on disk correctly done.

o **Partial failure.** ... failure occurred ... hence, some of the sectors ... the sector that is written ... may have been corrupted at the time failure occurred prior to the ...

o ... write ... the previous data value, ... disk remains unchanged.

– If during write of block, failure happens then system should detect it and a recovery procedure should be called to reinstate the block to a consistent state. For this purpose system must keep two physical blocks for each logical block. The following are the execution steps.

1. Output the information onto the first physical block.
2. If step 1 completes successfully, the same information is then written onto the second physical block.
3. When the second write completes successfully, then declare the operation is completed.

– As a part of recovery procedure from failure, each pair of physical blocks is carried out. If no error is not detected and both blocks are identical, then no further action is needed. If one block has a detectable error, then its contents are replaced with the value of the other block. If detectable error does not exist in both block, but both differ in content, then it is necessary to replace the content of the first block with that of the second.

– This recovery procedure gives the guarantee that, a write to stable storage either succeeds completely or results in no change. This procedure can be extended with no trouble to permit the use of an arbitrarily large number of copies of each block of stable storage.

Syllabus Topic : Tertiary-Storage Structure

5.3.9 Tertiary-Storage Structure

5.3.9(A) Tertiary-Storage Devices

Q. Explain various Tertiary-Storage Devices.

The main characteristic of tertiary storage devices is its low cost. Following are the examples of tertiary storage devices.

Removable Disks

– Floppy disk is the example of removable magnetic disk. The storage capacity of floppy disk is only

<uB. « 's a small Plastic ...
... D' 13 is recorded in n Co a, e d v P B=Se5 41?
✓ spots. We can transfer ...ppy disk in te ic I
X r t0 other using floppy the data from ... or in
computer ... Floppy disk drive is used to read and write data on floppy disk. Floppy disk drive one
3 ½ INCH which is called as micro floppy disk. Its size is
diameter of this micro floppy disk is 3½ inch. The
Hard plastic cover is used to cover ...
used to cover the Read/Write L* A me tal ...
automatically opened after **W S1> is
, „v disk drive. A notch is used lsk « mten *
floppy disk. Other type of floppy di slt ° *rite p to
disk. storage rapacity of 1 .2 M B ... size of **
with ...ter plastic plate coated with m ' 1 " a 51/, ' C,1

il „,agn£to , opt , C diSk « anoth er exa T* " Wrii1 "
* K, this disk, data is recorded P e of 0 Vah ,

® o with magnetic material i. J "M plan
... used is reasonably different from ... ...nology used to
L. This disk is more re s , slant tQ . 01 a ma goetic
coil in drive is responsible to create a m J?"* The
the field is produced at room temper atu J «« If
, „ e and too weak to magnetize a hi. ' « too
... on the disk. To
rf* a bib the disk head flashes a laser beam To
disk surface. at thc

Opdctd disk is also a removable disk which does
make use of magnetism. It uses special "«
can be altered by laser light to have icmuvely dark n materials that
bright spots. Example of optical * is pha . * « r
disk. This disk is coated with a material that can frKac
into either a crystalline or an amorphous state. Thl
phase change drive uses laser light at three different
piners: low power to read data, medium power to
erase the disk and high power to write to the disk.
Rumples are re-recordable CD-RW and DVD-RW.

All above disks arc called read-write disks. In contrast, wriie-once, read-many-times (WORM) disks can be written only once. CD-ROM and DVD-ROM are the eumples of Read-Only disks on which prerecorded data is available. Technology used by these disks is similar to that of WORM disks and they are very durable.

Tapes

- Magnetic tape stores large amount of data compared to Wai or magnetic disk cartridge. Although disk drives and tope drives have same transfer speed, random access on tape takes more time compared to disk seek. Tape drive is more expensive than disk drive. But cost of tape cartridge is less than that ...

Storage Management

magnetic ... is less than that S*. * ot vwk capacUy
C(j " SC(I I ... keep ...
*** to?'* *** 'dentT V,), umc <laU related to
used b y la lhc daia P... research. Usually tapes
between tape installation m tHIC , apcchan „ crH arc
The overall cost dhvc s and si™...age slots movc ,bcm
Upe library. of data storage is ? a hbTafy
The F reduced by robotic
...le which is usually
then ...

8i«al3 he " 0 «don |2 « disk _* "« needed
be ... If the same file ... This will reduce the cost per
"»■"> tooved... « w « w 1" future .hen it can

* Upe us *«"'*"«■
... between ... is called
disks and "* M " n « r , ll « , , Ma8e « *
, the |QW uonance of on-line magnetic
5helV& to a storage rlZ o f O, ,, hne Up « S,ttra|! 00

☞ Future TWMWngy

holographic storage makes USe
h raphic nhett ...graphs laser light to record
can be th ght o n Wial med A hologram
Pi*el is black the ' imcris ional array of pixels. If
represents bit 1 ep eRe lt,1 bit 0, and white pixel
single bit Ooe n ? lS W , piXd represents a
pixels in a hoi... o f hRhl UailsfcTS aU the
tremendous ... « 4 result the data transfer rate „
Micro...

sto teri?*T mOAanical stems (MEMS) based
fabric... o Ogy 1S n o* aclive arca of research The
...on technologies used to produce electronic
chips is applied for manufacturing the small data
storage machines,

5.3.9(B) Operating-System Support

This section explains how operating system carry our its job when removable storage devices are used,

☞ Application Interface

- Operating systems handles removable disks in the same maimer as they do handle fixed disks. It is necessary to format blank cartridge when it is inserted or mounted into the drive. An empty file system is then generated on the disk. This generated file system is used just like a file system on a hard disk.
- The handling of tape is carried out in a different manner. Initially. OS treats tape as a raw storage medium An application does not open file on device. An application opens the whole tape drive as a raw

device. Typically, the tape drive then is reserved for the exclusive use of that application untii 0* application exits or closes the tape device.

- As tape drive is presented as raw device. OS does no offer file system services. The application most make a decision about how to use the array of blocks. * application decides its own rules for organizing a tape, a tape full of data can usually be used by only the program that created it. For example, although a backup tape holds a list of file names and Hie sizes followed by the file data in that order, it is not easy to use the tape.
- Disk drive performs operation like read( ), write( ) and seekf). Instead of seek( ), tape drive uses locatef) operation. The locate) operation of tape is more accurate than seek( ) operation as tape gets positioned to correct logical block instead of entire track. Many number of tape drives supports read-position ( ) operation that returns the logical block number where the tape head currently exists. Many tape drives also support *a space(*) operation for relative motion.

☞ ***File Naming***

- On personal computer, file name contains name of the drive followed by path name. In case of removable disk, it does not mean that *if* drive containing the cartridge at some time in the past is known means how to find the file is known. If every removable cartridge had a unique serial number, the name of a file on a removable device could be prefixed with the serial number, but to make sure that no two serial numbers are the same would require each one to be about 12 digits in length. It would then difficult to remember 12 digits file number as a file name.
- In case of writing the data on a removable cartridge on one machine and then use the cartridge in another machine, the problem will become more complex. If both machines are same and with same kind of removable drive, then only trouble is to find the contents and data layout on the cartridge. If both machines are not of same types or drives are different, many other problems can come up.

Machines with different architectures have their data representation in different formats. Usually in modem operating systems, this name-space problem k

yedandaddressedforremovaHemedTX

on applications and users to discover how to acceT" 7 d

[illegible] the data. But, only some kinds of „ “,“ d

nie dta are so w ell standardized that all n lach i n es u * them in similar way.

☞ **Hierarchical Storage** Mana9em 7

- H„ ric jukebox facilitates for changing the A m 1 cartridge in a tape or disk drive without "“"L intervention. A hierarchical storage sy5tem hU X the storage hierarchy beyond main memory and W1 ' dary storage to include tertiary storage. Tcrtlajy storage is implemented as a jukebox of tapes w removable disks. This storage hterarchy is larger, cheaper, and slower.
- The normal method to include tertiary storage is te widen the file system. Small size and frequently USM files resides on disk, whereas large and old files WWcll having no active use is stored to the jukebox. In some file-archiving systems, the directory entry for the file keeps on existing, but the contents of the file no long;er holds space in secondary storage.
- Now day, hierarchical storage management is set up in installations with large volumes of data that are used rarely, at irregular intervals, or sometimes. Existing work in HSM comprises extending it to offer full information life-cycle management (ILM). In this case, data go from disk to tape and back to disk, as required, but are deleted on a schedule or in accordance with policy.

5.3.9(C) Performance

Tertiary 'Storage performance is measured in terms of speed, reliability, and cost.

☞ **Speed**

- Bandwidth and latency arc two metrics to measure performance of tertiary-storage devices. Bandwidth is measured with bytes per second. The bandwidth which is steady is the average data rate in a huge transfer- spectfically, the number of bytes divided by the transfer time. The effective bandwidth considers the average over the complete I/O time, together with the time for [illegible] locate(d any ajukeb°r. >an cartridge switching time in
- Essentially, dat lhe suslained ba "d»idih u cUeuhted as Se band w “ d „„ e — °r d.u and ». l(f„l« [illegible] is the overall data rate given by the drive. The sustained bandwidth dm. [illegible] » usually a bandwidth of a [illegible] case of removable disks which are slowest, the bandwidth ranges from a few megabytes per second

and for fastest; it is 40 MB per second. Tapes encompass a same range of bandwidths, from few megabytes per second to over 30 MB per second.

Latency is second metrics to measure performance. ... which tok performance is better ... jukebox is in operation, then ... to ... considerably ... performance ... latency ... robotic arm ... needed ... is required to ... and then the drive should spin ... operation takes ... front-access time within one ... longer ...

pitching disks in a jukebox leads ... high performance penalty.

The robotic-arm time is same for ... etching the tape, old tape requires re... ejected ... required for ... about 4 minutes. Several seconds are required ... loading of new tape in drive to get ... after prepare for I/O. In case of performance ... ready and ... a jukebox, the bandwidth and latency ... drives seem to have a terrible bottleneck ... reasonable.

Reliability

- Reliability is important metric to measure the performance. Removable magnetic disks ... extent is less reliable than are fixed hard disks. In case of removable disks, the cartridge is probably more exposed to damaging environmental conditions; for example, dust, changes in temperature and humidity, and mechanical forces like shock and bending.
- Optical disks are very much reliable. In this disk, layer storing the bits is protected by a transparent plastic or glass layer. The reliability of magnetic tape depends on type of drive and it differs largely. Some low-cost drives wear out tapes if used for few number of uses. Whereas, some other type of drives permits millions of reuses. The magnetic-tape drive head is a weak spot compared to magnetic-disk head.

Cost

- Storage cost is also important factor that is necessary to consider. Removable media lower the overall storage cost. Though it is somewhat expensive to create a removable cartridge, the cost per gigabyte of removable storage may well be lower than the cost per gigabyte of a hard disk. The reason is, the cost of one d...

... With ... many removable ... cost ... megabyte magnetic ... more ... orders of ... improvement ... for primary memory ... primary ... storage ... fall ... price per megabyte ... more speedily for disk drives than for tape drives as well. ... the price per megabyte of magnetic tape drive ... cartridge without ... small- and medium ... result, ... cost than disk ... system with equal capacity.

5.3.10 Swap-Space Management

Q. Explain management of Swap Space in detail

Swap-space management is ... the operating system ... job performed by ... hold all ... sufficient to ... disk ... simultaneously, virtual memory uses ... accessing the main memory, using swap space considerably reduces system performance. The main aim towards the design and implementation of swap space is to offer the best throughput for the virtual memory system.

5.3.10(A) Swap-Space Use

- The use of swap space is carried out in different ways by various operating systems. The way operating system uses swap space merely depends on the type of the memory-management algorithms in use. Such as, Swap space may hold complete process image, with code and data segments.
- Paging system may store the swapped out pages from main memory. The needed size of swap space differs based on the amount of physical memory, the amount of virtual memory it is backing, and the manner in which the virtual memory is used. It can range between few megabytes of disk space to gigabytes.
- It is better to keep sufficient amount of swap space that is required. Otherwise operating system will be forced to abort the processes as system may run out of swap space. There can be chances of system crash due to its running out of swap space.

- Many upereting systems and Linux permit the use of multiple swap spaces. These swap spaces are maintained on separate disks- As a res ■- can be o„ *the VO* system by P«S'"8 spread over the system's I/O devices.

5.3.10(0) Swap-Space Location

- **A swap space can part of *the normal* Hie system, or it can be in a separate disk partition. If it is J^usl 0 / large file in the file system, normal file-system routines can be used to create it, name it, and allocate its space. This scheme is simple to implement but is not efficient. In this approach, navigation of direc toiy structure and the disk-allocation data structures consumes more time and also extra disk accesses are needed. External fragmentation can very much increase swapping times by forcing multiple seeks during reading or writing of a** process image.
- Instead, swap space can be created in a separate raw partition. In this approach , a separate swap-space storage manager allocates and reclaims the blocks from the raw partition. The algorithms used by swap-space manager are better in terms of speed. For these algorithms, speed is important instead of efficiency as (here is frequent access to the swap space. As data remains for short period in swap space, internal fragmented is accepted.
- This is not the case with file system. Adding extra swap space needs repartitioning the disk, this requires to move the other file-system, partitions or destroying them and restoring them from backup or adding another swap space in another place, Linux supports both I schemes: in raw partition and in file system.

Syllabus Topic r 1/0 *systems*

5.4 I/Q systems

5.4,1 Overview

- As we know that I/O is one of the main functions of an operating system and is used to control the entire computer's Input/Output devices.
- Basically it should have lo igsu . ' «ces, catch i Bfcmipts . J, commands to the errors. » should also *Provide* an j Bterface between the devices and the rest of the system that is simple and easy to use.
- The — m . Computers operate with ma- rOt* devices. As we know that it include * kinds m sks tapes), transmission devices (tet work device* modems), and h0 man-interface devices (screen, k ri.®' use) -

- a KA) *I/O Devices*

Q. i xplain different ! o,>/OdeMicro

n . ral |y when computer system uses *VO* devices « a externa "devices can be grouped into three categories -

1. **Huma**" «" d « bk , « * SUi,ab ,e Co mrnuni «Un g with the computer user.

   Examples : P™ ters ternunals , vid « display, keyboard, and mouse.

2. **Machine readable** : It is suitable for communicating with electronic equipment.

   Examples : Disk drives, USB keys, se nsors , controllers, and actuators.

3. **Communication** : It is suitable for communicating with remote devices.

   Examples : Digital line drivers and modems.

- Computers system operates with many kinds of devices. It includes storage devices (disks, tapes) transmission devices (network cards, modems), and human-interface devices (screen, keyboard, mouse) ,
- A device communicates with a computer system by sending signals over a cable or even through the air The device communicates with the machine via a connection point termed a port (for example, a serial port). If one or more devices use a common set of wires, the connection is called a bus.
- When device X has a cable that pl ugs into device Y and device Y has a cable that plugs into device Z, and device Z plugs into a port on the computer. such type of a ZZu DUS. ment is called a daisy chain - " >y w

5.4d(B)Differences between I/O Devices

Q. *SZ. dif,eren, Parameters* distinguish 1/O

Devices can be disti mguished with respect to following parameters.

1. Data transfer rate
2. Device application
3. Control r - plexiiy
4. Data transfer unit
5. Data representatii on
6. Errors

- **Data transfer rate :** Data transfer rate of different devices can be different and can be of several order of magnitudes.
- different devices ... tom.
- **Control complexity :** The complexity required to control the devices are also varies.
- **Transfer ...** characters or blocks.
- **Data representation :** The data ... encoding ... also varies different devices.
- ... characters of errors ... one to another. ... broadly from ... are

Topic : Overview of I/O Hardware

Device Controllers

- Input/output units consist of a mechanical component and an electronic component. A controller is a section of electronics that can operate a port, a bus, or a device.
- A serial-port controller is an example of a device controller. It is a single chip in the computer, controls the signals on the wires of a serial port.
- The implementation of SCSI bus controller is done most of the time as a circuit board which separately plugged into the computer.
- It normally includes a CPU, microcode, and some private memory to facilitate it to process the SCSI protocol messages. The SCSI bus controller is integral part of the some of the devices.
- Total four registers are present in I/O port. These are as follows :
  - o Status register
  - o Control register
  - o Data-in register
  - o Data-out register
- The host can read the hits in status register. These bits show states like :
  - o The current command is completed or not
  - o A byte exist in data-in register or not which is to be read
  - o If device error present
- The host writes the control register to ... ate a command or to alter the mode of a device. A particular bit in the control register ... full-duplex ... another ... communication.
- Some of the ... are for selection of one of the speeds provided by the serial port.
- The host gets input by reading data-in register. It writes the data out register in order to send output.
- The size of data registers ... typically 1 to 4 bytes. Some controllers have FIFO chips that can hold several bytes.
- ... This is because ... of the size of the data register.
- ... a small bit ... FIFO chip work like buffer ... receive those data until the device or host ...

5.4 Polling

Q. Write note on "Polling".

- In complete protocol for interaction between the host and a controller can be intricate, but the basic handshaking notion is simple.
- The controller indicates its slate through the busy bit in the status register. (Recall that to set a bit means to write a 1 into the bit, and to clear a bit mean to write a 0 into it.)
- When controller is busy in operations, it sets a busy bit. This busy bit is cleared when it is not working and ready to accept the next command. The host signals its desires by means of the command-ready bit in the command register.
- This command-ready bit is set by host when a command is available for the controller to execute.

For this example, the host writes output through a port, coordinating with the controller by handshaking as follows.

1. Until bit becomes clear, the host constantly reads the busy bit.
2. In the command register, the host sets the write bit and then writes a byte into the data-out register.
3. Command-ready bit is set by host,
4. When the controller notices that the command-ready bit is set, it sets the Busy.
5. After this, the command register is read by controller and sees the write command.

6. In *order* to get the byte, it reads the data-out register and perform the I/O to the device.

7. The different bits are then cleared as : The controller clears the command-ready bit, clears the error bit in t e status register to specify that the device I/O succeeds and clears the busy bit to specify that it is finished.

If the status of host is busy-waiting or polling; It reads the status register repeatedly until the busy bit turn out to be clear. This means it remains in loop until busy bit clears. If the controller and device speed is better, this method seems to be practical one. If host has to wait for longer period of time; it should perhaps go for another task.

5.4.2(C)Interrupt Handler

Q, What is intemjpt? Explain the tasks carried out by I interrupt haodter.

- The CPU hardware contains interrupt request line that the CPU senses after executing every instruction. When the CPU detects that a controller has asserted a signal on the interrupt request line, the CPU saves a small amount of state, such as the current value of the instruction pointer, and jumps to the interrupt-handler routine at a fixed address in memory.
- The interrupt handler finds out the reason of the interrupt and carries out the required processing. After this the interrupt handler executes a return from interrupt instruction to return the CPU to the execution state before the interrupt.
- The device controller first affirms a signaJ on the mtemrpt request line. It does so to raise rhe interrupt. The CPU catches this raised interrupt and ;ends out 'o the terenupiban.gerforxerviciugihe dev.ee.
- ITe imenupt handfer cJears in(cmjpt by giving the rupt handling ="X,TcT„" *asynchronous event fn.* respond to an example a device controller becoming ready for service.
- The modern operating system should following interrupt-handling features. support for
- The ities to when processi
- It should have efficient X XX g «. inten * h * B * j
- Wre Pri a te int eWhandJer Ue to send out to the 50 out poij ing ajj (he /, deWce , I(sh oM do oae *raised the* int emip(deWCeS t0 rve which
- A multilevel interrupts should be supported. It helps the operating system to differentiate between high and interrupts so that it can respond with the Iropna.e degree of necess -
- modem computer hardware, CPU and the inter controller hardware offers alt above three fea tUn .s *
- CPUs have two interrupt request lines. One is ttll — n- maskable interrupt, which is reserved for events S1)ch unrecoverable memory errors. The second in ternip , £ js maskable.
- Itcanw '-" — - off by the CPU before the execution of critical instruction sequences that must not be interrupted. The maskable interrupt is used by d eV j ce controllers to request service. This address is an offset in a table called the interrupt vector. This Vc€ contains the memory addresses of specialized i nterru * handlers.
- The purpose of a vectored interrupt mechanism i s to reduce the need for a single interrupt handler to se rch all possible sources of interrupts to determine wk- WQich one needs service.
- The interrupt mechanism also implements a ** joi£in of interrupt priority levels. This mechanism enables th CPU to defer the handling of low-priority intcrnm. without masking off all interrupts, and makeT* possible for a high-priority interrupt to preempt the execution of a low-nriority interrupt.
- The interrupt mechanism is also used to handle a wide variety of exceptions, such as d ividing by accessing a protected or nonexistent tnetnoty address attempting to execute a privileged instruction from user mode.
- A system call is a that is called by " application to invoke a eme" k 1 service - The checks th. system call data structure 8 tree' given by the appUca,ion >builds <• and then execute * arguments to kernel, spec,ai called a software ■n'errupt, o 'X inStlUC,iOn
- o be used contro| P with the ,0 manage 0,6 flow of efficiently, We n d eme ,, If "" disks are to be used Previous one completes* 1311 the next V o " Soon " the
- Consequently, the kem i read is implemented bv ! code that com P letes a disk
- The i/o statu • y apairof interrupt handlers. recorded by high-priority handler. It then clears the dev ;Ce interrupt begins the subsequently

5.4 p) Interrupt Service Routine (ISR)

Interrupt means event, which invites attention of the processor on occurrence of some action at hardware or software interrupt instruction event. In response to the interrupts the routine or program, which is running presently interrupts and an interrupt service routine (ISR) executes. ISR is also called device driver in case of the devices and called exception or signal or trap handler in case of software interrupt.

Processor executes the program, called interrupt service routine or signal handler or trap handler or exception handler or device driver, related to input or output from the port or device or related to a device function on an interrupt and does not wait and look for the input ready or output completion or device-status ready or set.

The programmer may register Interrupt Service Routines (ISRs) to handle hardware interrupts, re like routines are not independent threads, but mo signals. Any currently executing thread is suspended by an interrupt, and the ISR called. The ISR n»« on a Wte interrupt stack only if the hed onto the Otherwise, the ISR stack frames are pus stack of the interrupted thread.

Most of the thread manipulation within interrupt service 'hllads een ThreadOu T routine. example, in implementing a time-slice scheduler. However the ISR, although more flexible then in typical OSs, still tool ISRs can use CTbcISync daw is provided especially foi ISR routines.)

5.“bh, J ory Accent (DMA)

Q. Explain steps in DMA transfer.

When an Permit transfp , unit will be provided to external device° and lhe primary memory, without Access mu Of DMA.

Disk involves the transfer of large size information in a singl operation. This transfer of information between disk and main memory involves DMA operation. When used together with an interrupt, the processor is conveyed only after the whole block of data has been moved.

Information is transferred in the form of bytes or words and for these it must supply the address of memory location and every bus signals that control the data transfer. Communication with a device controller is handled through a device driver. Device drivers are part of the operating system.

The operating system provides a simplified view of the device to user applications (e.g,, character devices vs. block devices in UNIX). In some operating systems (e.g., Linux), devices arc also accessible through the /dev file system.

In some cases, the operating system buffers data that are rrPd between a device and a user space network buffer). This usually performance but not always. The DMA controller tas access to the system j bus independent of the CPU registers dw P rocessor can -d or write. These renters compn several control registers . countregister,ands gi

- The function of the control registers is to detail about the facts like I/O port to utilize, reading from or writing to the I/O device, the unit of data or information transferfin terms of byte or word at a time), and the total bytes to transport in one burst.
- Without DMA, let us see how disk read occurs. Initially the controller reads the block from the drive serially, bit by bit, until the complete block is in the controller's internal buffer.
- Next, it calculates the checksum to confirm that no read errors have occurred. Then the controller causes an interrupt.
- After this, operating system can read the disk block from the controller s buffer a byte or a word at a time by executing a loop.
- In one byte or word is read from controller device register and storing it in main memory.
- With DMA, the process is different. First the CPU programs the DMA controller by setting its registers so K becomes aware about what to transfer where (step] in Fig. 5.4.1). It also instructs to the disk controller to read dau from the disk into its internal bu er and confirm the checksum. DMA can start only which valid data are in the disk controller's buffer.
- The DMA controller begins the transfer by issuing a read request over the bus to the disk controller (step 2). The disk controller remains unknown and does not care whether the read request came from the CPU or from a DMA controller.
- Naturally, the memory address to write to is on the bus' address lines so when the disk controHer fetches the next word from iqteniaJ buffcr ft knows where to write it. The write to memory is another standard bus cycle (step 3).
- When the write is ends, the disk controller sends an acknowledgement signal to the disk controller, also over the bus (step 4). The DMA controller then increments the memory address to use and decrements the byte count.

If the byte count is still greater than 0, repeated until the count reaches 0. At that time, the DMA controller interrupts the CPU to let it know that the transfer is now complete. When the operating system starts ... disk block to memory- L a? a ot have ‹o Co Py ,he shows DMA operation.' there. Rg , 5 ,4 -<

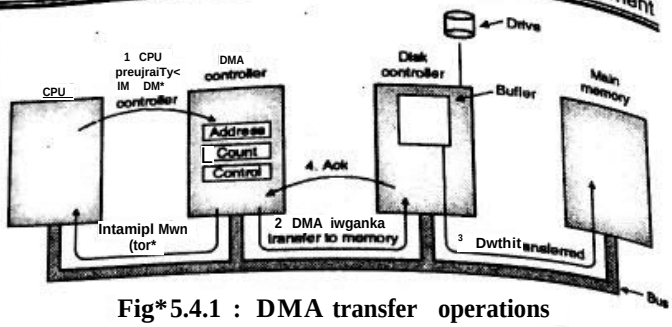


Fig* 5.4.1 : DMA transfer operations

Syllabus Topic : Application I/O Interface

5.4*3 Application I/O Interface

- Due to interfaces, operating system treats I/O devices in a standard and uniform way. In VO related part of kernel module, the device-driver layer hides the differences among device controllers from the subsystem of the kernel, much as the I/O encapsulate the behavior of devices in a few classes that hide hardware differences from applications.
- input-output the hardware, then the job of the operating system developer will become simple.
- Every type of operating system has its own for the device-driver interface. It is the difficult device-hardware manufacturers.
- A given device may ship with multiple device drivers- for instance, drivers for MS-DOS, Windows 95/98, Windows NT/2000, and Solaris. Devices vary on many dimensions.
  - **Character- stream or block** : A character stream device transfers bytes one by one, while a block device transfers a block of bytes as a unit.
  - **Sequential or random-access** : A sequential device transfers data in a fixed 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.
  - **Synchronous or asynchronous** : a synchronous device exhibits predictable response times. An asynchronous device exhibits irregular or unpredictable response times.
  - **Sharable or dedicated** : A sharable device can be used concurrently by several processes or threads; not.

Speed of operation :Device speeds range from a few bytes per second to a few kgabytes per second.

ge»d •wr'1*' «»d only, Or » rite only, vic* perform both input and output. Some support only one data transfo dil but others

ock or Character D

4(A) 01 ~' =V, CBB

r h iock device interface holds f t* "8 disk drives and other b1 tS eSSen ia pt The device must recognue co X:k 0 "ente d devic L writeO ;if it is u random such w read(Lt have a seek 0 command to de ce - " « L l0 transfer next. These devices are *' *'"* Citation through file system interfaces. S,ed by

Sing sy m jmd Database management system , Ls** the b "* d a S mplc "ear blocks. H application carries out its own buffe * ing then raw raW deV C r aC 5S Pass nontm! of * device directly to the application.

Si help to step out the OS to carry out salne - Lfiott- W is CaHed raw UO - So OS will not be Led to perform these operations. UNIX allows a ode of operation on a file that disables buffering and firing called as Direct I/O.

advantageous to have memory -mapped file access f on top of block-device drivers. A memory-capped interface Offers access to disk storage via an array of bytes in main memory.

The system call that maps a file into memory returns the virtual memory address that contains a copy of the file. The real data transfers are carried out only when required to satisfy access to the memory image.

As the transfers are handled by the same means as that used for demand-paged virtual memory access * memory mapped I/O is efficient. The i example of a device that is accessed throug a interface is keyboard.

- The basic system calls in this ,mcrf ** ubraries application to get() or pulO one . crface that offers can be implemented on top o editing line-al-a-time access with buffering and services.

W) Network Devices

y ioten ns

Network VO and disk VO diffeIS /°L r ist ics of performance and addressing ch n

The majority of operating interface In M * * LLT* UI1ers * OTk vo available in many °T*rai in g systems. The m ** k * Wrf ace offers of a or functitm * 0 which infor wl ** seleci(UWon vhe mptemcnution called, to be rece ivedatvi Z ktt havii g a packet waiting packet to be sent. By using select() the polling and busy waiting is eliminated that would otherwise be essential for network I/O.

5.4.3(C) Clocks and Timers

The hardware clocks and timers available on most of the computer performs three basic functions :

- Pro Vlde the cutrem time.
- the time.
- Stt a 10 cause operation X at time T.

The operating system makes heavy use of these fu nc tioiis. The time sensitive applications also make hea vy use of these functions.

Unfortunately, the standardization of the system calls that implement these functions is not done across operating systems.

The hardware to determine elapsed time and to trigger operations is called programmable interval lime. This can be set to wait a definite amount of time and then cause an interrupt.

The setting can be for generating interrupt once or to repeat the process to cause periodic interrupts. When process is al the end of its time slice, it should be preempted.

The scheduler uses above method to generate an interrupt to preempt such process. The disk I/O subsystem uses it to invoke the periodic flushing of dirty cache buffers to disk, and the network subsystem uses U to cane JT rations that are proceeding too slowly because of work conges fion M failures.

The operating system may also provide h ta user processes to use timers.

5.4.3(D) Blocking and Non-blocking I/O

Q. Qflerentiate t een blocking and nonblocking VO.

[Toperaang System (MU - IT)

nce of I/O

o. , Explain issues related lo performs system.

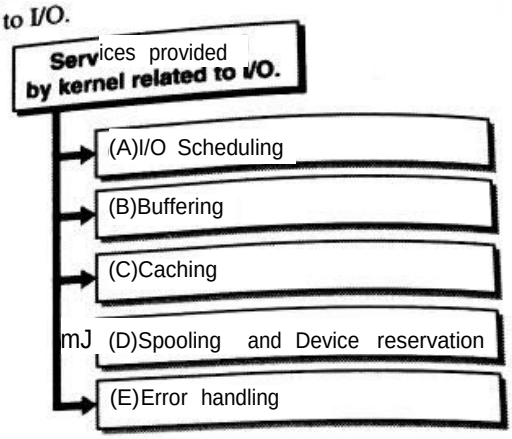
- The execution of the application ets suspended after it while executing 'he issues a blocking a system c . blocking it will application was in run queue - move to wait queue,
- After the system call finishes, the application is moved back to the run queue, where it resume execution.
- When it resumes execution, it will receive the values returned by the system call. The physical actions performed by VO devices are usually asynchronous they take a varying or unpredictable amount of time.
- However, most operating systems use blocking system calls for the application interface. This is because the blocking application code is easier to understand than non-blocking application code.
- Apart from blocking VO, some of the user-level processes require non-blocking I/O. One example is a user interface that receives keyboard and mouse input while processing and displaying data on the screen.
- Overlapping of execution with VO can be achieved by writing multithreaded application. While some threads cany out blocking system calls, at the same time others carry on execution. The Solaris developers used this technique to implement a user-level library for asynchronous I/O, freeing the application writer from that task. Some operating systems provide non-blocking VO system calls.
- If the call is non-blocking, then it does nut stop the execution of the program for an extensive time. It returns promptly, with a return value that shows number of bytes transferred.
- The substitute to a non-blocking system call is an asynchronous system call. An asynchronous call returns immediately, without waiting for the VO to complete. The application continues to execute its code.
- The completion of the VO at some future time is communicated to the application, either through the setting of some variable in the address space of the application or through the triggering of a signal or software interrupt or a call-back routine that is executed outside the linear control flow of the application.
- As compared to asynchronous system calls a non-Wocking readQ returns without delay with whatsoever data are available. This can be full number of bytes guested. wer, or none at all. On the other hand an asynchr" 0US its entirety but will complete at some nrmcd m future time.

Tropic : K.mel I/O Subsy.teTT

5.4.4 Kernel VO S ufasyStem

Q. Explain various services provided by kernel are related to I/O.

Following services are provided by kernel which are related to I/O.



Fig, C5.13 : Services provided by kernel

➜ 5.4.4(A) I/O Scheduling

- It involves deciding the good order in which I/O requests should be executed. Scheduling can enhance the overall system performance. Processes also shares the device accesses in fair manner due to scheduling. The average waiting time for I/O to complete also can be minimized.
- There is a wait queue of requests maintained for each device. The I/O request of the application is placed on the wait queue for that device when application issues blocking I/O system call. All these requests are then arranged in a manner such that, the overall system efficiency and the average response time experienced by applications will be improved.
- Operating system tries to give equal service to all the applications. Delay-sensitive requests may be serviced on the basis of priority. For asynchronous I/O, kernel should keep track of many I/O requests simultaneously Hence, the OS might attach the wait queue to a device' status table. This table keeps the entry for each VO device and its management is carried out by kernef

- ... table entry shows the device's type, address, and state of the device. If the device is busy in processing a request, the parameters will be stored in the table entry. Scheduling of I/O operations improves performance of the I/O operations by improving the performance of the system.

- ... main memory or on disk via techniques called buffering, caching and spooling.

5.4.4(B) Buffering

- First reason for buffering is to handle speed mismatch between the producer and consumer of a data stream.
- The second reason for buffering is to adapt between devices that have different data-transfer sizes. These disparities are particularly common in computer networking. Where for fragmentation and reassembly of messages, buffers are heavily used. Buffering is also done to support copy semantics for application I/O.
- Buffering needs to be consider for block and character devices for a diversity of reasons. For example when process reads data from a modem, user process do a read system call and block waiting for one character.
- Each incoming character causes an interrupt. The interrupt service procedure gives the character to the user process and unblocks it. After pulling the character somewhere, the process reads a further character and blocks another time.
- The problem with this approach is that the user process needs be started up for every arriving character. Permitting a process to run again and again for short runs is inefficient, so this design is not a good one.
- An enhancement on above approach is to have user process an *n* -character buffer in user space and does a read of *n* characters. The interrupt service procedure places arriving characters in this buffer until it fills up.
- After this, it wakes up the user process. This proposal seems to be more efficient than the previous one. The limitation of this scheme is that if buffer is paged out when a character arrives, the buffer could be locked memory. In this way, if many processes initiate locking pages in memory, the pool of available pages reduce in size and performance will degrade.
- In other approach, a buffer can be created within the kernel and the interrupt handler places the characters in this buffer, the page ...

- ... with the user buffer is ... buffer ... when needed, and the ...
- This approach ... arrive while ... brought in from ... the buffer ... As the buffer ... then, ... over this is to have a second buffer ... When the second buffer is used ... copied to the ... liable to be ... copied to user space, the second buffer is being ... one can be used for new characters. In this way, the two buffers can be used alternatively, ... to user space, like this ... new input. A buffering scheme called double buffering ... input, also important on output.

5.4.4(C) Caching

- The cached data is accessed more efficiently compared to original, for example, disk stores instructions of currently executing process. Same instructions are then cached in physical memory, and copied again in the CPU's secondary and primary caches.
- Buffer holds the only existing copy of a data item, whereas, cache holds, a copy on faster storage of an data item that resides in another place. Sometimes, a part of memory area can be used for both buffering and caching. For efficient scheduling of disk I/O, operating system uses buffers in primary memory to hold disk data. The I/O efficiency for files shared by many applications is improved by using these buffer maintained in main memory as a cache. The files being written or reread speedily also makes use of these buffer as cache.
- For the file I/O request, kernel first check buffer cache in main memory to avoid physical disk I/O. In addition, disk writes are collected in the buffer cache for several seconds. This helps to collect the large transfers to permit efficient write schedules.

5.4.4(D) Spooling and Device Reservation

- Some devices, such as printer cannot accept interleaved data streams. A buffer that holds output for a device ... Printer can print a single job at a time. ... want to print their output ...

simultaneously, then each ones output =, spooled to « separate disk file. Once the printing ' queued the output is dine, the spooling syster corresponding spool f or output to the printer.

– These queued spool files then are cop icd b y spools system to the poorer b * agcm e|.l of *t»°" n « precess is responsible for In other opera ting system*. Zme openmag system. is managed by enm-kentd thread.

– For this purpose. opera "J d ,y SBnia dmini*ire K ,R , o interface that allows users eliminate unwanted j ob s prior to display the queue, to whjle lhe printer is those jobs print, to suspend pnm g serviced, and so on.

– Devices like tape drives and printers, cannot multiplex the I/O requests of multiple parallel applications. Spooling is one means operating systems can manage concur - r output. Other method to handle concurrent device access is to offer explicit for coordination.

■4 5.4.4(E) Error Handling

- If an operating system uses protected n]em< , 1C d can protect against many types of hard is application errors. Hence, complete system failure avoided if minor failure occurs. There can be a transient reason to fail devices and *I/O transfers* sue as network becomes congested . The permanent reason for such failure can be, for example disk controller becomes faulty.

– In case of transient failure, operations can be repeated to recover from failure. But it is not possible to recover in case of permanent failure.

– An *I/O* system call returns single bit as a status indicating success or failure. UNTX uses integer variable named ermo to return an error code. Near about hundred values are used to indicate the nature of the failure.

5.4.5 I/O Protection

– User process should not issue any illegal I/O instruction to disturp the operation of system. To prevent the users fiom performing illegal I/O operations, all I/O instructions are defined as privileged instructions.

– User program has to execute system call in order to o7Z' ne r a ,,,, 8 SyStem Cany 0Ut 1/0 On behalf ehZ JJeth 8 SyStem , n,nning « m o nitOT mode request is the valid or invalid. If it is valid, then carry out trie I/O requested.

– Memo* protection rt fiVS tem should protect any memory-mapped and I/O po« memory locations from r reject ah user access. Kernel cannot just «ceCSS

5.4.6 Kernel ™ta Structures

– Kernel kccpS state information regarding the use of I/O cO mpo nLS oDen -file table structures, kernel Uses just Hke the r, dif&rcnt VO activities like netw ork m any structure vicc communications etc. connections, cha

Syllabus Topic : Irfa ,.itlJiin*nil 1/0 Requests to Hardware Operations

5.4.5 Transforming I/O Requests to Hardware Operations

Q. Explain i® - RWU es , is transto,m ,o hard *are operations.

– This «P lainS the t° W o S Co D neCtS « P l 08ram rcq «e S' ■>set of network wires or to a specific disk r ln or der to read a file from disk, the application JX W <he data by a file name. Instde the disk, this file name is mapped by file system through the file-system directories to get the space allocation of the file.

– In MS-DOS, file name's first part preceding the colon, for example C: indicate that it is a primary hard disk. Then, c: is mapped to exact port address through a device table. Colon separates the device name space from the file-system name space within each device.

– Modem operating systems get important flexibility from the several stages of lookup tables in the path between a request and a physical device controller. The methods to pass requests between applications and drivers are common. Hence, new devices and drivers can be added without recompiling the kernel. Some operating systems can load device drivers on demand.

Following are the steps involved in typical life cycle of a blocking read request. It shows I/O operations consumes many CPU cycles.

1. A process makes a blocking read () system call to a file descriptor of a file that has been opened previously.
2. Kernel code related to system call verifies the parameters for correctness. If required data is in the buffer cache, then it is returned to the process which completes the I/O operation.
3. If required data is not in buffer cache then a physical I/O must ** carri ed out. The requesting process is then

moved from the run queue to the wait queue of the device, and the I/O request is scheduled. Finally, the I/O subsystem sends the request to the device driver. The request is either sent via a subroutine call or an in-kernel message depending on operating system.

4. The kernel buffer space is allocated by device driver s to take delivery of the data and schedules the I/O. Finally, the driver issues commands to the device controller by writing into the device-control registers.

5. The device controller controls the device hardware to carry out the data transfer.

6. The status and data may be polled by driver, or it may have initiated a DMA transfer into kernel memory. DMA controller manages the transfer, which generates interrupt after completion of transfer.

7. The correct interrupt handler gets ... through the interrupt vector table. ... signals the device driver, and returns ... essential data from t...

8. The device driver gets the signal, ... request has finished, finds the status of request, which signals the kernel I/O subsystem that the request has been accomplished.

9. The address space of the requesting process ... data or code from kernel and process gets ... from wait queue to ready queue by kernel.

10. As process is in ready queue, it is now ... When process gets CPU, it resumes execution at completion of the system call.

Syllabus Topic : STREAMS

5.4.6 STREAMS

Q. What is STREAMS? Explain.

- STREAMS are a mechanism offered by UNIX V system. This mechanism allows an application to assemble pipelines of driver code dynamically. A stream is a full-duplex link between a device driver and a user-level process. It consists of a stream head, a driver end, and zero or more stream modules. Stream head interfaces with user processes. The device is controlled by driver end. Zero or more stream modules exist between stream head and driver end as shown in Fig. 5.4.2.
- The read queue and write queue is associated with each module, stream head and driver end. The functionality of STREAMS processing is ... The ioctl( ) system call is used to push modules onto a streams.

- The exchanges of messages take place between queues in adjacent modules. Therefore, queue present in one module may overflow an adjacent queue. Hence queue should support for flow control.

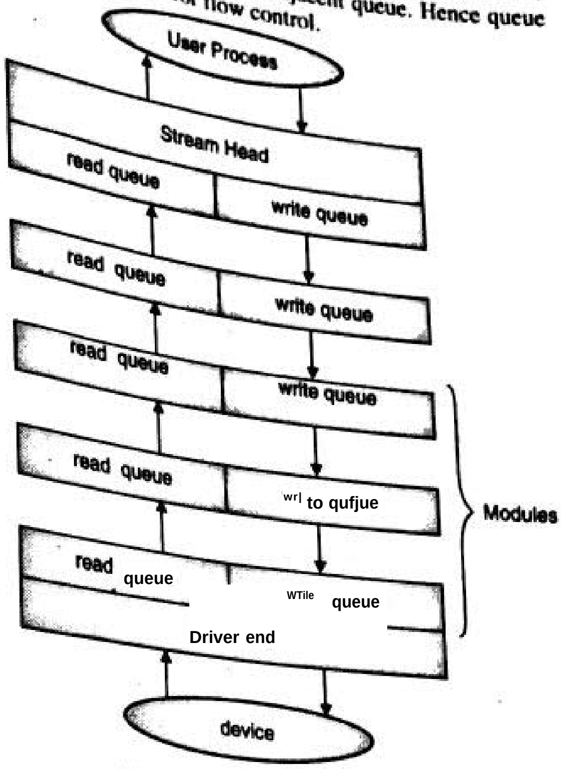


Fig. 5.4.2 : The STREAM structure

- If sufficient ... that support ... available ... messages also ... exchanged among queues in adjacent modules.
- The write( ) or putmsg( ) system call is executed by Process for writing the data to device. The write( ) system call is executed to write raw data to the stream, while putmsg( ) permits the user process to state a message. For any of this system call the stream head copies the data into a message and sends it to the queue for the next module in line.
- This copying of messages carries on till the message is copied to the driver end and so the device. In the same way, the read( ) or getmsg( ) system call is executed by process to read data from the stream head.
- STREAMS I/O is asynchronous (or nonblocking) excluding when the user process communicates with the stream head. The driver end also has a read and write queue. It is necessary that driver end must respond to interrupts, for example, one triggered when a frame is ready to be read from a network.

Syllabus Topic ; Performance

5.4.7 Performance

- **The performance of the system is affected by the 1/0 <uid hence it is important factor to consider in performance. CPU has to execute device driver code and a/so ft has to carry out scheduling of the processes in fair and efficient manner. Also due to I/O. context switches occurs and hence, it burden the CPU and its hardware caches.**
- **If there arc some inefficiencies in the interrupt- handling mechanisms in the kernel the, I/O can expose them. Also due to I/O, memory bus gets heavily loaded during data copy between controllers and physical memory. It also happens when copy takes place between kernel buffers and application data space.**
- In interrupt handling, it is require carrying out several operations. Hence interrupt handling is expensive task. During completion of I/O, process gets blocked. As a result of this blocking, context switching is required which lead to certain overhead on system performance. Several context switches and state switches takes place when there is communication between the machines in network. For example, during remote login many context and state switches takes place.

Following are the measures to improve I/O efficiency.

I- It is required to lessen the number of context switches.

2. *The* frequency *of* copying the data in memory While passing between device and application should be reduced.

3. Use large transfers, smart controllers, and polling to lessen the frequency of interrupts.

4. Use DMA-knowledgeable controllers or channels to relieve the CPU from simple data transfer in order to improve the concurrency.

5. Processing primitives should be in hardware to permit their operation in device controllers in parallel with CPU and bus operation.

6. Try to keep balance in performance related to mernoty. CPU, bus and I/O. If one area gets overloaded then it will cause the area to be idle.

5.5 Exam Pack (University and Review Questi Ori s)

☞ **Syllabus Topic : File System - File Con te Pt**

Q. **Explain various file attributes in brief** *(Refer section 5.1.2)*

Q. **Explain various tile operations in brief** *(Refer section 5.1.3)*

Q. **Write short note on File Types.** *(Refer sectio 1 5.1.5)*

Q. **Explain the different techniques to a»n** files. *(Refer section 5.1.6)* CtUr o 'be

☞ **Syllabus Topic : Access Methods**

Q- Explain different file access methods *(Refer section 5.1.7)* (10 Marks) **(June 2015)**

Q. Write short note on File Access (sequential anr *(Refer section 5.1.7(1))* -ess).

Q. Write short note on File Access (Random acce *(Refer section 5,1.7(2))* ss).

☞ **Syllabus Topic : Directory Structures**

Q. Explain different methods for defining the logical structure of a directory. *(Refer section 5.1.8)*

Q. Explain different directory operations. *(Refer section 5.1.10)*

☞ **Syllabus Topic : File Sharing**

Q. Write note on "access rights". *(Refer section 5.1.13)*

Q ' Explain issues related to file sharing. *(Refer section 5.1.13(A))*

0- Explain different access rights to the tile. *(Refer section 5.1.13(A))*

Explain in detail operation of remote file system. *(Refer section 5.1.13(B))*

☞ **Syllabus Topic : Protection**

Q* Explain file system protection in detail. *(Refer section 5,1.14)*

☞ **Syllabus Topic : File System Implementation**

Q. Explain file system structure. *(Refer section 5.2.1)*

☞ **Syllabus Topic : Implementing File System**

Q. Explain the implementation of file system in detail. *(Refer section 5,2.2)*

Write notes on virtual file system. *(Refer section 5.2.2(B))*

Syllabus Topic : Directory Implementation

Explain approaches for implementation of directory. *(Refer section 5.2.3)*

What are the different allocation methods to File System? Explain with example. (10 Marks)

Explain contiguous file allocation method and its advantages and disadvantages. *(Refer section 5.2.4(A))*

Explain linked list file allocation method and its advantages and disadvantages. *(Refer section 5.2.4(B))* **(B))**

Explain indexed file allocation and its advantages and disadvantages. *(Refer section 5.2.4(D))*

Explain the structure of i- node in detail. *(Refer section 5.2.4(D))*

How i-node is used to allocate the file ? *(Refer section 5.2.4(E))*

Syllabi Topic : Free Space Management

List methods of free space management. *(Refer section 5.2.5)*

Explain bit map method in detail. *(Refer section 5.2.5(A))*

Describe linked list of disk blocks method. *(Refer section 5.2.5(B))*

Syllabus Topic : Efficiency and Performance

Explain various techniques to improve efficiency and performance of secondary storage. *(Refer section 5,2.6)*

Syllabus Topic : Recovery

Explain file system recovery in detail. *(Refer section 5.2. 7)*

Whjs Topic : NFS

Explain operation of network file system in detail. *(Peter section 5.2.8(A))*

Explain the layered structure of NFS. *(Peter section 5.2.8(C))*

Syllabus Topic : Storage Management

Explain physical structure of magnetic disk. *(Refer section 5.3.1)*

Explain physical structure of magnetic tapes. *(Refer section 5.3.2)*

Explain host attached storage. *(Refer section 5.3.3(A))*

Explain network-attached storage. *(Refer section 5.3.3(B))*

Syllabus Topic : Disk Scheduling

Explain techniques of disk scheduling. *(Refer section 5.3.4(A) to 5.3.4(E))* (10 Marks) **(Dk. 2016)**

Explain various Disk Scheduling algorithms using following example. Took FCFS, SCAN, C-SCAN, SSTF, LOOK, C-LOOK. (10 Marks) **(Dec. 2014)**

Example 5.3.3 (10 Marks) **(May 2016)**

Example 5.3.11 (10 Marks) **(June 2015)**

Example 5,3, 12 (10 Marks) **(Dec. 201 5)**

Syllabus Topic : RAID Structure

Q. **Explain various RAID levels.** ***(Refer section 5. 3.7(A))***

Syllabus Topic : Tertiary-Storage Structure

Q> Explain various Tertiary-Storage Devices. *(Refer section 5.3.9(A))*

Syllabus Topic : Swap-Space Management

Q. Explain management of the swap space in detail. *(Refer section 5.3.10)*

Syllabus Topic : I/O systems

Q. Explain different types of I/O devices. *(Refer section 5.4.1(A))*

Explain different parameters to distinguish I/O devices. *(Peter section 5.4.1(B))*

Syllabus Topic: Overview VO Hardware

Write note on "Polling". *(Pets' on 5.4.2(B))*

Q. What is interrupt? Explain the tasks carried out by interrupt handler. *(Refer section 5.4.2(C))*

Q. Explain steps in *DMA* transfer. *(Refer section 5.4.2(E))*

☞ **Syllabus Topic : Application I/O Interface**

Q. Differentiate between blocking and nonblocking I/O. *(Refer section 5. 4. 3(D))*

Q. *Explain issues r81at6d to perfofmance Qf I/o* system, *(Refer section 5. 4.3(D))*

☞ **Syllabus Topic : Kernel I/O Subsystem**

Q. Explain various services provided by k... nel ar e related to I/O. *(Refer section 5.4.4)*

☞ **Syllabus Topic : Transforming I/O Requ... to Hardware Operations**

Q. Explain I/O Request is transform to h... ar... vi/are operations. *(Refer section 5.4.5)*

☞ **Syllabus Topic : STREAMS**

Q. What is STREAMS? Explain. *(Refer section 5 4 ..6)*

□□□

CHAPTER 6

Module VI

Distributed Systems

Syllabus Topics

Distributed Operating System : Network h Structure and Protocols; Distributed ...-based OS ... Versus 3tct=!"S Service, File p-- ... Naming and tr... and To P° lo gy. Communication C trof and Deadlock Handling, ... distributed Svnrk ... ynchrontzahon : Mutual Exclusion, Concurrency

Syllabus Topic : Distributed Operating System

6.1 retributed Systems

Q. g fjne distributed system.

The development of powerful microprocessors and invention of the high speed networks arc the two major I velopments in computer technology.

Many machines in the same organization can be connected together through local area network and iflformalion can be transferred between machines in a very small amount of time.

As a result of these developments, it became easy and f Unable to organize computing system comprising large number of machines connected by high speed I networks.

;■ Over the period of last thirty years, the price of I microprocessors and communications technology ... constantly reduced in real terms. Because o ... ractical I distributed computer systems appeared as a p... substitute to uniprocessor and centralized sy... stems.

r The networks of computers are present all over. I Internet is composed of many netwo... rks. All these networks separately and in cofnbindt ... ll pertinent the necessary characteristics that m... topics to focus under distributed system

6.1.1 Definition

A computer net work *s defm ... □ together by ... re connect ... communicating devices ...

communication links. These devices include computers, printers and other devices capable of sending and/or receiving information from other devices on the network. These devices often called as node in the network. So computer network is interconnected set of autonomous computers.

A distributed system is defined as set of autonomous computers that appears to its users as a single coherent system.

Users of distributed system feel that, they are working with a single system.

Main characteristics of distributed system

Q. What are its characteristics? Explain.

Following are the main characteristics of distributed system,

- A distributed system comprises computers with distinct architecture and data representation. These dissimilarities and the ways all these machines communicate are hidden from users.
- The manner in which distributed system is organized internally is also hidden from the users of the distributed system.
- The interaction ... nf users and applications with ... distributed system is in consistent ... sptfe of where and when interaction occurs ... and identical way. in
- A distn... n ...ri system should allow for scaling it ... suppo rt for availabilit y. It
- Distributed system should ... should be aiw®j ... arable to the users and applications of failures.
- X ... handling should be hidden ... and application ...

6.1.2 Motivation

Q. What are objectives behind building rhe distributed system?

Following are the objective behind building distributed system.

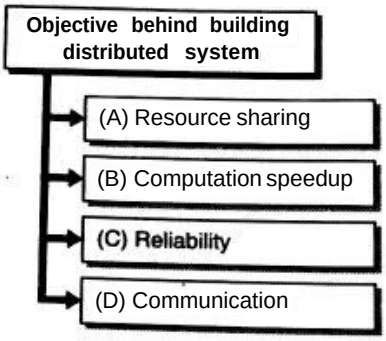


Fig. C6.1 r Objective behind building distributed system

➜ 6.1.2(A) Resource Sharing

- Resource sharing offers saving in cost. One printer can be shared among many users in office instead of having one printer to each individual user. There can be more saving in cost if expensive resources are shared.
- The increase in connectivity and sharing also increases the security risk and to deal with it is equally important. Presently, systems offer fewer defenses against eavesdropping or intrusion on communication,
- A communication can be tracked to construct a favorite profile of a particular user. This clearly violates privacy, particularly if it is done without informing the user, A allied problem with increased connectivity can also cause unnecessary communication, for example electronic junk mail, called as spam. Special information filters can be used to select inward messages based on their content.

➜ 6.1.2(B) Computation Speedup

- Many big computations can be divided in sub computations which can be assigned to different nodes in network so that work can be carried out in parallel.
- In this case, speed of computation increases. As another example, process may be migrated from heavily loaded machine to lightly loaded machine in network.

➜ 6.1.2(C) Reliability

- System does not affeci with failure of one node in network as other nodes are available in system.
- In case of failure, other nodes can take over computations to achieve reliability. This failure should be detected and recovered by system itself. Once the recovery is done, final computation must result iQ consistence state.

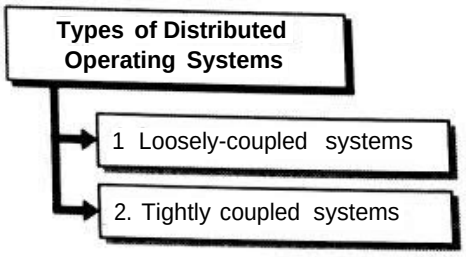
➜ 6.1.2(D) Communication

- Machines in network communicate with each other by exchanging messages. At low level, this communication takes place through protocol stack between machines.
- At highest level, applications exchange these messages Due to such communication, many users can be part of a single work to be carried out by sharing the information.
- By connecting users and resources, it becomes easier to work together and exchange information. The success of the Internet is due to its straightforward protocols for exchanging files, mail, documents, audio, and video.
- The worldwide spread people can work together by means of groupware. Electronic commerce permits us to purchase and sell verity of goods without going to shop or even leaving home.

6.2 Types of Distributed Operating Systems

Q. Explain different types of distributed operating systems.

- In a network, multiple machines may have different operating system installed on it. Operating systems for distributed computers are categorized as :



t ig. C6.2 : Types of Distributed Operating Systems

➜ 1. Loosely -coup led systems

In a set of computers, each has its own OS and there js coordination between operating systems to make their own services and resources available to the others. This loosely coupled OS is called as **network operating**

...ystem (NOS) and is used for heterogeneous multicomputer systems,

I **Tightly coupled systems**

Jf JteCP[5] a sinfi,e ' global v]ew of the fljtf*® [68] * SUCh tiEhtl * c ou pled ...alled **distributed operating** s v ...

... js useful for the manage, ... joiuogeneous multicomputer. ... underlying hardware. This hardware ... many processes and details ... hidden.

Z **yllabus Topic ; Network Based OS**

6.2.1 Network Operating System (NOS)

ffusers have NOS installed on machjn ... either access resources on other ... cy can login or transfer the data frnm ... by remote ... machine t0 their ... machine.

6.2.1(A) Remote Login

, Network operating system allows users to |0 remotely- The Internet offers the **telnet** facing fo" remote login. NOS permit users (0 use services available on a particular machine in the network Remote login service provided by NOS allows the user to Jog in remote machine from his/her terminal.

- Using command for remote copy, user can copy the file from one machine to other.

621(B) Remote File Transfer

- NOS offer a mechanism for remote file transfer from one computer to another. Here, each computer has its own local file system. If a user working on computer A wants to access a file on another computer B, then the file must be copied explicitly from the computer B to computer A. File Transfer Protocol (FTP) is used for such transfer in internet environment. User invokes the FTP program as:

ftp name of computer B

After entering the correct username and passwor ... user should connect to the right subdirectory ... to access the file required file. Suppose user wan ... ed from computer B abc.frt then this file must be copi ... man d. tocomputer A by executing following co

get abc.txt

- The location of M ... hl ... transparent to user. ... r,ie ... c °pits •X 10 copy from ... Whkh Wa *te th hmc 10 Other ... man > c °nnect 10 ... nJnt,in & on J m ... Thc au heruj ... rCqUCSl FTP COmpulcr Watchcs of fo]l ... 1O„ o f This d ... „n<1 completes ... -Wln & predefined ... aCm o n res P° nds the «:t b get' « ... q commands.

Distributed S...

Foster fife f...

2. Put: transfer fi] ... e from ... 10ca1 COrn Puter t0 remote computer, ... o m remo|c computer to local computer.

3. Wdir: Uslfik ... s in the current directory on the re °W computer.

4. cd: Change current ... d,recto ™ *e remote computer. IIB

6.2.2 Distributed Operating Systems (DOS)

local resourc[1]... 1 , 11 o pera,m * 'kstem. access to remote and sy...? ... UrceS '» be I" the same way. Operating ... controls the data and pro... xss migration from one site to another.

(A) Data Migration
(B) Computation Migration
<S rocess Migration

6.2.2(A) Data Migration

- User working on machine A can access data (such as a file) that reside at machine B, One method to data migration is to transfer the complete file to machine A. After this all access to the file is local. When the user need is completed, a copy of the file if it has been modified is sent back to machine B.
- Although a small change has been made to a large file, all the data needs to be transferred. This method was used in the Andrew file system. This method of file transfer is inefficient. In other method of data transfer only those portions of the file that are actually needed for the immediate task gets transferred.
- Jf another part of the file is needed later on, another transfer will take place. When the user no longer wants to access the file, any part.of it that has been modified ... he sent back to machine B. The Sun Xylems Network File System (NFS) protocol M L method. The Microsoft SMB protocol ... SeS • ... n top of either TCP/IP or the Microsoft (running NetBEUI protocol) 3150 a ... ows fi,e sharing o wr a network.

6.2.2(B) Computation Migration

- The computation rather than the data can be transfer across the system. This approach is called computation migration.
- If applications needs to access various large files that reside al different machines, to obtain a summary of those files. In this case, it would be mare beneficial and efficient to access the files at the machine where they reside and return needed results to the machine that initiated the computation.
- If the time required to transfer the data is longer than the time to execute the remote command, the remote command should be executed.

6.2.2(C) Process Migration

Q. What are the reasons behind process migration?

Process migration involves transfer of process to other machine for execution purpose. The whole process, or portion of it, may be executed at different machines. The reason for migration is :

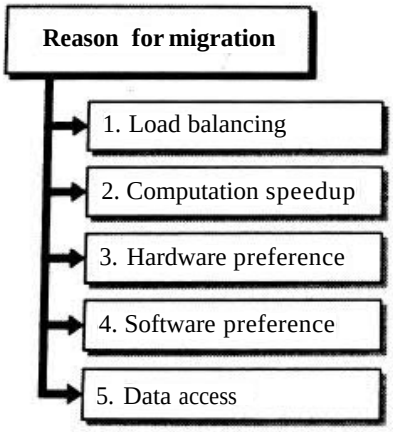


Fig. C6.3 : Reason for migration

1. Load balancing

The processes (or sub processes) may be distributed across the network on different machines to balance the workload.

2. Computation speedup

The subprocesses of the single process can run concurrently on different machines then the total process turnaround time can be reduced.

3. **Hardware preference**

The particular process may need some specialized processor for execution.

4. Software preference

The needed software by process may available at only a particular machine, and either the software cannoi be moved, or it is less expensive to move the process.

5. Data access

If the huge amount of data requires in the computation it may be more efficient to have a process run remotely than to transfer all the data.

Syllabus Topic : Network Structure

6.3 Network Structure

- Distributed systems are built on top of computer networks, which are of two types: LANs (I.ocal Area Networks) and WANs (Wide Area Networks).
- LAN covers one room, building or campus, WAN covers a city, country, or whole world. Ethernet is example of LAN.
- Internet is collection of thousands of separate networks and can be considered as one WAN.

6.3.1 Local-Area Networks (LANs)

Q. Explain LAN type of network.

- Instead of having large mainframe computer, it seemed to be more economical to have many several computers. Hence, LANs emerged in early 1970 as a substitute to for large mainframe systems which many organizations were using for their compulation.
- As LANs cover small area such as LAN covers one room, building or campus, the communication links between nodes are having more speed and less error rate.
- The reliability and high speed can be achieved by using high quality cables to connect the different computers in network.
- Twisted pair cables or fiber optics cables are most commonly used in local area network.
- IEEE Standard 802.3 describes classic Ethernet in which a coaxial cable is used to connect many computers. The cable is called the Ethernet. In the very first version of Ethernet, a **vampire tap** was used to attach a computer to the cable. Connecting the machine to cable was carried out by drilling a bole halfway through the cable and screwing in a wire leading to the computer.

Multiaccess bus, ring, and star networks are the most common configurations for local area network. Infrared and Bluetooth network have communication speed of 1 megabit per second.

Gigabit Ethernet have speed of 1 gigabit per second. 10BaseT Ethernet have speed of Ten megabits per second. 100BaseT Ethernet requires a higher-quality cable but runs at 100 megabits per second and is becoming common.

LAN contains several small and large computers, peripheral devices which share... and gateways to connect to other networks.

6.3.2 Wide-Area Networks (WANs)

Q. Explain WAN type of network.

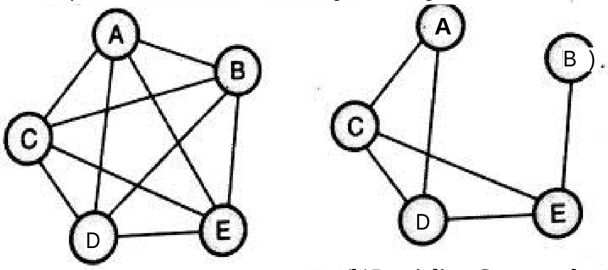
- WANs cover a city, country, or the whole world. Internet is a collection of thousands of separate networks and can be considered as one WAN. WAN was emerged in late 1960 as an academic research project to offer efficient communication among machines. The main objective behind this project was to share hardware and software in convenient and economical manner by a wide community of users. Arpanet is the example of first WAN developed.
- In WAN, machines are physically distributed over a large geographical area. Therefore communication links are relatively slower compared to LAN and links are also not reliable. Internet WAN offer capability for communication between different machines which are geographically distributed.
- The Internet comprises hosts and routers which are also computers. Hosts are PCs, notebooks, handhelds, servers, mainframes, and other computers owned by individuals or companies that want to connect to the Internet.
- Routers are also computers with many incoming and outgoing lines. It accept packets on one of many incoming lines, process it and send them along one of many outgoing lines. Routers are connected together in large networks.
- Backbone networks are large national or worldwide networks operated by telephone companies and ISPs (Internet Service Providers) for their customers. Internet is organized with backbones at the top. many backbone operator. Backbone central faceted by high bandwidth fiber optics. Telephone companies have connections to backbone.
- Host are networks and ... connected to backbone. Regional backbone by ... at ISPs are connected to routers in medium-speed fiber.
- Corporate networks routers are connected to regional modem bank routers at ISPs are connected to every host. ISPs customers. In this way, often on internet has at least one path, and many paths, to every other host.

Syllabus Topic : Network Topology

6.4 NetWork Topology

Q. Explain different topology for network with its advantages and disadvantages.

- Machines in distributed system can be connected in different ways called as topology for that network. Each topology has its advantages and disadvantages. Installation cost, communication cost and availability criteria are used to compare different configurations.
- Various topologies are shown in Figs. 6.4.1, 6.4.2, 6.4.3 and 6.4.4. In fully connected networks, each node is connected to every other node in network. Hence, number of links increases as square of number of sites. Also installation cost is high. Hence, this topology is impractical to consider building the large network.



(a) Fully Connected Network (b) Partially Connected Network

Fig. 6.4.1

- In partially connected networks, there exist direct link between nodes, but not between all nodes as in fully connected network. Therefore installation cost is less than fully connected network. As there is no direct link between ... must be routed via other nodes. This ... higher communication cost. Because of failure of some links, some of the nodes may become unreachable.
- Tree structured networks, ring networks, and star networks are examples of partially connected networks.

networks. Tree-smictured network relatively have low installation and communication cost. However, due to tbc failure of a single link, network may become partitioned.

- In ring network, if two links fails then and then network can become partitioned. So availability of ring network is higher than tree-structured network. Burring network incur high communication cost as messages needed to cross large number of links to reach to destination.
- In star network, if single link fails, then only single node will be unreachable. In star network, communication cost is low but if central node fails, then all nodes in the system became disconnected.

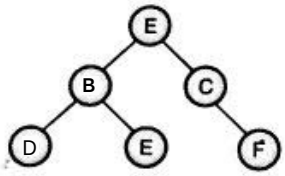


Fig. 6.4 J; Tree-structured Network

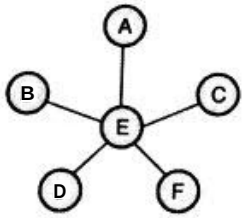


Fig. 6.43: Star Network

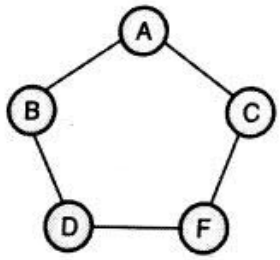


Fig. 6.44 : Ring Network

jy flhusJTopic: Communication Structure

6.5 Communication Structure

Following five different issues are important to address while designing the communication network.

'different while designing 1
Itto communication network |

1. Naming and name resolution
2. Routing strategies
3. Packet strategies
4. Connection strategies
5. Contention

Fig. C6.4 : Issues while designing communication network

-4 6.5.1 Naming and Name Resolution

F Q> what is naming? How names are resotved?[1]| Explain .

- Processes running on different machines must be able to specify each other. On each machine, process has process identifier- The messages to be sent to the processes may be addressed to the process identifier As machines do not share memory, initially machines have no knowledge about processes on remote machines.
- To solve this problem, a pair <host name, identifier* is used to identify the process on remote machine. Host name is unique in network and is alphanumeric so that it is easy to recognize it. Within any host, identifier is either process identifier or other unique number used in that host to identify the process.
- Names are used to identify the hosts as they are easy to remember. But machines (hosts) use an identifier for simplicity and speedup. Hence. It is necessary to have some mechanism to resolve host name into host*id. 1wo solutions exist. First each host may have data file which contains host name and addresses of all hosts in network. This was the approach used initially in internet. But this approach requires updating the data files on all machines whenever host are added or removed from network. Due to growth of internet, this approach is not used for name resolution.
- The second approach is domain-name system (DNS) which specifies the naming structure of the hosts, and also name-to-address resolution. Names starts with most specific part, where each part is separated by dot operator and ends with general part. For example, *aihu.ityale.edu* indicates host name athu in information

technology department at y ale – Uln \eisity in the domain edu.

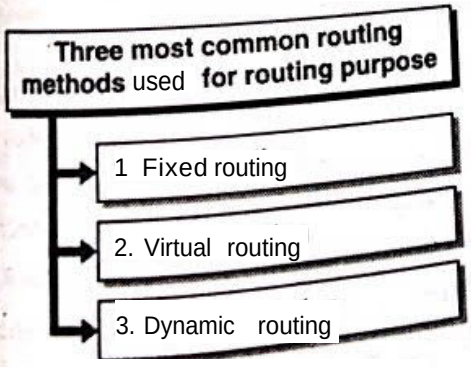
...bove address each component h£ ss i0 system) which takes a name ...s a name server C the "" T" rcSPOn *«>le I?"* the Finally, name server for the host is . ° , hat -ante. ...then host-id. Consider Machine J'' 6,1 which X t is , issued by process X on whi ch ...athu.it.yale.edu. Following steps t lhe host resolve the name *athu.it.yale.edu.* lnv °lved to

- Kernel on host X issue request to nanie server of edu domain. It ask to name server about ...wale. -The edu name server's address U v of i fice initial request was issued. ' know °>
- 'jjK?w ------ne server r ------~ "duress Of *yale.edu.*
- Kernel then request to name server *v„i* , asking a... *it-pcc rtf it x- j j*
- To the return address by *it.yale.edu* kernel issues request for address of *athu.it.yale.edu.* Finally IP address (host-id) of athu is returned.

465-[2] Routing Strategies

Q. Explain different routing strategies.

- Machines in internet communicates by exchanging packets. Each packet contains address of source and I destination. Router contains routing tables. It extracts address in incoming packet and look up the table to find which outgoing line to send the packet on and thus to which router.
- This process is repeated until the packet is delivered to !he destination host. The routing tables arc very dynamic and are updated continuously as routers an ... links go down and conic back up and as ... conditions change.
- Following are the three most common routing methods used for routing purpose.



Flg.C6 : R o u « W m£lh " dS

1. Fixed routing

In this type of routing, path from sender machine to destination machine is specified in advance. This path changes if and only if a hardware failure disables it.

Otherwise, it does ... is cl, Q5w ta t y hj ''8'Always theshorIt ·5t lh ... communication costs.

2. Virtual routing

In 'his type of ... destination ntJ™' 1„8 , P>fll f„>ra "„■>« machine to S*s®lon. ... fixed for the duration of one

- Different paths ... the messa... f USed y different sessions to send '
- U • OmSOurct destination.

involved in°r ** tht Shorl period just like pcriod Period 1 5 trans tr or 85 h)ng as a remote-login

** 3' **Dynamic routing**

- In dynamic routing, the path used to send a message source to destination machine is selected only when a message is sent. As decision is taken dynamically, different paths are used by different messages.
- in general, a source machine sends a message to destination machine on whatever link is the least used at that particular time.

65.3 Packet Strategies

- There are two types of services : Connection-oriented service and connectionless service. With connection-oriented network service, user initially establishes a connection, uses the connection, and then releases it. Over this connection, the sender sends bits at one end. and the receiver takes them out in the same order at the other end.
- On the contrary, with connectionless service packets ... routes to reach the same ... arrive in out of order at ... r . " " - zz x. z- - — ... oriented service.
- Services are characterized by a quality of service. Reliable service is implemented by giving a guarantee to sender that packet is correctly delivered to receiver.
- Receiver scM 's Acknowledgement packet as a cohfiraaho" of delivery of pacle t to the sender. wledgements introteC s delay and are essential to detect SC Xtl 2 paCte * l0SS bU ,

slow the communication. File transfer is example of reliable connection-oriented service.

— Reliable connection -oriented service supports either message sequences or byte streams.

- In message sequences message boundaries are preserved, fn byte streams, the connection is simply a stream of bytes, wilh no message boundaries.

- For application such as digitized voice traffic, the delays introduced by acknowledgements are unacceptable. Here noise on telephone line or a garbled word from time to time user can prefer with compare to delay due to waiting for acknowledgements.

– Unreliable connectionless service is called as datagram service and does not offer acknowledgement as a confirmation of delivery of packet to die sender,

— The acknowledged datagram service can be used to send one short message instead of establishing the connection. The request-reply service can be used for client server communication. Following are the six types of network services.

*+ 6.5.4 Connection Strategies

Q. What are the different connection strategies used in network communication?

– Once connection is established between source and destinations, processes can set up communications sessions to exchange information. There are number of ways by which two communicating processes may connect. Following are the three most common schemes used.

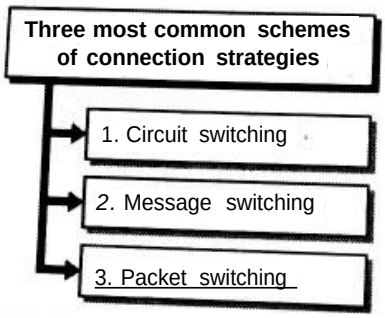


Fig- C6.6 : Schemes of connection strategies

1. Circuit switching

– It is similar to system ,0 COnneCti o n estab,isl »“nt in telephone

– Two communicating parties continue to talk over a communication line till connection i$ terminated Dunng communication period, no one can use this line

– A permanent physical link is established between the two processes that want to communicate with each other.

– Any other process cannot use this established 1ink till communication session is completed. Hence in cixcujt switching, link is allocated to for the duration of communication session.

2. Message switching

– In message passing, a temporary link is established f$_{Or}$ the duration of one message transfer between iw$_o$ communicating processes.

– As per need, physical links are allocated dynamically among correspondents. These links are allocated f$_{Or}$ only short duration.

– Each message contains data with other system information like source, the destination, and Error Correction Codes (ECC). This information is used by the network for correct delivery of the message to the destination.

3. Packet switching

- Each message from application is first broken up in small chunks called a packet.

- Each packet contains data with source and destination address as each packet may be sent to its destination separately. Each packet reaches to the destination machine through separate path.

– As packets takes different route to reach the destination, all must be reassembled into messages as they arrive.

6.5.5 Contention

It several computers want to transmit over a same communication link then collision occurs. This situation occurs in ring and multi-access bus network. To avoid the collision as it degrades performance, several techniques have been developed.

CSMA/CD

– In this technique, every computer sense the channel before transmitting to check that other machine message is currently being transmitted or not. This is called carrier sense multiple access technique. If collision is detected, then transmission gets terminated.

– On Ethernet, computer has to first see whether other computer is transmitting packet or not. Then it can send the packet. If two or more computer sends packet at the

...me time collision occur and both detect it. So until Lfijst computer finishes sending, other computer can ...t sebdPet.

...Vision occurs, both the ... issions. Here both ...nates their ... amount of time between 0 and T Microsecond ... sends the packet. ... Vision occurs, all colliding ... into the interval 0 to 2T ... again. On each ... is doubled ... Visions. This algorithm ... backoff. ... as binary ...

... Sisseton Ethernet ... also a maximum number ... connected to it, a multiple ... entire campus- ...

A bridge is used to connect these ... which permits traffic ... another when the source and destin... side. ... Ethernets use switches to avoid collisions. Each ... contains ports, to which computer ... another switch can be attached. A ... toffeted in switch and sent out on the port ... destination machine stays. As each ... connected to separate port ... but requires bigger ...

Token Passing

- Token is a unique message that is circulated in network. Any machine want to transmit holds the token. Once transmission completes, it releases token so that other computer can use it. Token may loss so system should be able to generate new token.

Syllabus Topic : Communication Protocols

6.6 .-Communication Protocols

< Explain different communication protocols.

Protocol is set of rules by which ... coWunicaies with each other. Many pf... host-host ... such as router-router protocols, different ... ocols, and others. Protocol stack layers ... protocols on top of one another. This protocol stack is ... used by all modern ... protocol stack deal with ... International ... interconnection ... Fem ... owing are the functions of each ... communication between two machines.

Physical layer: This layer deal with ... physical ... any volts to represent 0 and ... of bits per ... these ... simple, ... whether ... duplex or duplex; all ... are defined by physical layer. It is implemented in hardware of ...

Data link layer: it ... channel ... occurred ... during ... mission.

3. **NetWork layer :** All,k ... layer offer, ... The ... for routing the ... ebons ... is ... P...siblc ... on network, ... versa ... address to logical address and ... mainta...

4. **Tranenew** 1 ... provides low-level access to the network. It is responsible for ... sfer of messages between clients, includir... ing breaking the messages into ... mets. sequencing of packets, ... control, and generating physical addresses

5. **ession layer-.** The session layer is responsible for establishing the sessions between users. It also implements process-to-process communication protocols. It provides services like dialog control, synchronization, token management.

6. **Presentation layer:** This layer deal with syntax and semantics of information exchanged between machines. As machines may have different data representations, presentation layer resolve the differences in formats among the various machines in the network, including character conversions and half duplex-full duplex modes (character echoing).

7. **AuoUcution tayen** This layer inUuacb directly with ... It contains different protocols needed by users such ... j pyP TTP, Telnet, DNS, and SMTP, This layer ... tals with Gl... ttansfcr ... remolc ... logm ... Protoco,s ... and electronic mail.

- Most of the distributed systems use the Internet as a base. Hence, these system uses two important Internet protocols: IP (Internet protocol) and TCP (Transmission Control Protocol), IP (Internet Protocol) is a datagram protocol. In IP, a sender sends datagram, of up to 64 KB over the network and no guarantees are given for its delivery. The datagram may be fragmented into smaller packets and travel independently, possibly along different routes. At destination host when all packets reaches, they are assembled in correct order as per sequence number and delivered to the application.
- IP protocol has two versions, v4 and v6. Version 4(V4) are currently in use and v6 is up and coming. IP v4 packet starts with a 4O-byte header that contains each 32-bit source and destination address with other fields. These are called IP addresses and routing is carried out using these addresses.
- IP does not offer reliable communication in the Internet. To offer reliable communication, TCP (Transmission Control Protocol), is present on top of IP. TCP makes use of IP to offer connection-oriented streams. The remote process always listen the incoming connection on port number. Remote process is specified by IP address of machine and port number. Sender first establishes the connection and sends bytes over that connection which is guaranteed to come out the other end undamaged and in the correct order. The TCP gives this guarantee by using sequence numbers, checksums, and retransmissions of incorrectly received packets.

Syllabus Topic : Distributed File Systems

6.7 Distributed File Systems (DFS)

Q, Explain working of distributed file system (DFS).

- In distributed system, files are available on several computers. Computers in distributed system can share these physically dispersed files by using distributed file system. Service offers particular function to client and it is a software entity running on some machines. Server runs service software on single machine. Client process invokes the service through some set of defined operations called as client interface.
- File system offers file services to clients. ITie set of primitive file operations are create a file, delete a file, read from a file, and write to a file. Client interface is formed with set of these operations. File server controls secondary-storage devices on which files are stored. These files are accessed from these devices as per request of client.
- There can be different implementation for DFS. Server may run on dedicated machines. In other implementation both client and server may run on same machine DFS can be part of DOS or it can be a software layer managing communication between file system and NOS. DFS should come into view to its client as conventional centralized file system. The servers and storage devices which are on different machines in network should be invisible to clients. DFS should fulfill the request of client by arranging the required files or data.
- As data transfer is involved in operation of DFS, its performance is measured with amount of time required to service the client request. Storage space managed by a DFS includes different and remotely located small storage spaces.

Syllabus Topic : Naming and Transparency

6.8 Naming and Transparency

- Mapping between logical object and physical object is naming. User always knows logical name of the file but system has to manipulate the data blocks which is actually stored on disk sectors or tracks. Actually text name of file gets mapped to numerical identifier which in turn again mapped to disk block. Due to such mapping user remains unknown about location of file on secondary storage.
- In case of DFS, location of file in network remains unknown to the user. This is the new dimension added to the abstraction with compare to conventional file system. In conventional file system, file remains on disk of same machine whereas in DFS location of file can be in disk of any machine in the network. File replication is supported in DFS so that several replica of file exist. By given file name then set of locations of the file is returned by mapping.

6.8.1 Naming Structures

Name mapping should support location transparency and location independence.

1. **Location transparency :** Physical storage location of file cannot be known by using file name.

[J Ctun independence , AhhZr** »., 1

r location of the Chan8eS , lherc « »» need to change (k.

Sk"ı»16 , n* clHie nt DFS system , upports S‹a ,ic PPir

" ted by these systems. Henri ,n,grj "ion is 8 ot

Independence is immaterial for these , noil on of location sterns.

' tion independence is suptxjrted ltu Ülen files o

, « lo caJ data TMn " ,,crs that ar p Can be

" , speeifv borage location. If on Cached

parency is supported, the file kx ,

"f of physical disk blocks although J l**1 to remain hidden from users. gn the * block,

With location transparency, users can sharp r)1

files simply by naming the files in the m0le

manner j t B® *e files are local. In this "X "'

the storage space is cumbersome. , Sdarin g

Location independence allows sharing of both

end data objects as well. Hence, it is u "o" ge

balancing the disk utilization across the system , n th«

case, naming hierarchy is separated from lhe hierarchy

of storage devices hierarchy and front the inter computer structure. Once separation between file name

and storage space completes, client can now access file from remote server also.

If clients are diskless then it relies on servers to provide afl files and OS kernel. This is needed as diskless client cannot use DFS code to get back the kernel. There is a special boot protocol in ROM which is invoked by client This facilitates networking and get backs either boot code or kernel from fixed location. As soon as kernel gets copied over network and loaded, DFS makes available all other OS files.

* In current systems, client uses both local disks and remote file servers. Operating systems and networking software are stored locally. File systems holding user cUra and sometimes applications are stored on remote file systems.

W Naming Schemes

Q1 Ptain various naming schemes.

Ttere are three approaches available for narru ng scheme in DFS.

In frtt approach, combination of f,ic s stein -wide JocaJ name is used. This ensures a unique name. In Ibis, host: local -name combination!

Th-

Distrihirtmj Systems

d, v, l Lld' 1 L PprDKh is used in network file system "'•Wortcin. , Su " " yaterns. QNC+ a

vendors. NFS package "PPorted by many UNIX

package. NFS xr po,K "1 of — i networking

cok tor " s to local a-" ra " hamsm to attach remote

intrudUc dircctOT ed now jn V u ? s Which autOrn appeare to u °unt feature is

based on a table of NFS, m °unts arc done on demand, names. mou nt points and Restructure

h third apDrn ach, all the files in the system

in a single global name stru are covered Saturn s n)s , cture. This file-system

c entional file lsom <>rphic to the structure of a

tTUkes « difficu i7t tem PraCt ‹Cally *some special files 81<Jbal name structure ** om <teh lhe 8 of single

WoLh ? approaches , 1,1051 complex is the

Really by WS slructure is complex and practicall most diffici)lt (o As maintain.

remote ditectQry to local one. hierarchy

becomes more unstructured, Due to non availability of server, some directories on different computers also become unavailable,

6 8.3 Implementation Techniques

- As transparency is important, implementation should include the mapping of a file name to the associated location. It is necessary to combine files in component units instead of keeping a single file, so that resultant mapping would be manageable. It also helpful from administration point of view.
- Replication, local caching, or both can be used for improving the availability of mapping information. If location independence is supported then consistent update of this mapping is not possible. This is due to change of locations. Solution to this problem is to use low-level location-independent file identifiers.
- File names which are in text form are mapped to lower-level file identifiers that specify to which component unit the file belongs. As the identifiers are locations independent, its replication and caching can be earned out freely without being invalidated by migration of component units. This will result in need of a second Lei of mapping. which mapS COmPO « n, " Nations and needs a simple yet consent update mechanism.

— ***Structured*** **names arc commonly used for low-level identifiers. These names are bit strings with two parts. The first part is component unit to which the file belongs. The second part indicates a file within the unit. Implementation with more than two parts is also possible. This explained technique is used in Andrew File System (AFS).**

Syllabus Topic : Remote File Access

6.9 Remote File Access

– Suppose, client forward the request to server to access remote file. Naming scheme locates the server and actual data transfer between client and server is achieved through remote-service mechanism. In this mechanism, request for accesses is forwarded to server, which then performs accesses and returns (he result to user or client. This is similar to disk accesses in conventional file system.

- Caching can be used to improve performance of remote-service mechanism. In conventional file system, caching is used to reduce disk I/O. The main goal behind caching in remote-service mechanism is to reduce network traffic and disk I/O. Following are basic caching schemes in DFS.

6.9.1 Basic Caching Scheme

- If data needed for operation is not cached by client then it send request to server for the same. Now, client performs the accesses on cached data which is sent by server. Ail the future repeated accesses to this recently cached data can be carried out locally. This will reduce additional network traffic.

– Least Recently Used (LRU) algorithm can be used for replacing the cached data. Master copy of file is available on server and its part is scattered on many client machines. If copy of the file at client side modifies then its master copy at server should be updated accordingly called as cache-consistency problem. DFS caching is network virtual memory which works similar to demand-paged virtual memory having remote server as backing store.

– In DFS, data cached at client side can be disk blocks or it can be entire file. Actually more data are cached by than actually needed so that most of the operations are carried out locally.

- **If granularity of caching is large size chunks then hit ratio increases. But, if miss occurs then more data is needed to retrieve from server which increases network traffic. It in turn will increase the possibility of consistency problems as well. The network transfer unit and the RPC protocol service unit should be taken into account wile selecting unit of caching. For large caches large block sizes are beneficial.**

6.9.2 Cache Location

– Cache location can either be main memory or disk If cache is kept in main memory then modifications done on cached data will lost due crash. If caches are kept in disk then they are reliable. No need to fetch the data during recovery as data resides on disk. Following are the advantages of main memory caches.

- o II permits for diskless workstations
- o Data access takes less time from main memory compared to access from disk.
- o Performance speed up is achieved with larger and inexpensive memory which technology demands today.
- o To speedup I/O server caches are kept in main memory. If both server caches and user caches are kept in main memory then single caching mechanism can be used for both.

– Many remote-access implementations take the hybrid approach considering both, caching and remote service. In NFS, for example, the implementation is based on remote service but is improved with client- and server side memory caching for performance.

6.9.3 Cache-Update Policy

- System's performance and reliability depends on the policy used to write back updated data to the master copy of file which is available on server. Following policies are used.

Write-through policy

– This policy write data through to server copy (disk) the moment they are placed in any cache. This policy is more reliable as little information is lost when client system crashes. The write performance is poor as each write access has to wait until the information is sent to the server.

...yed-write policy or write ... caching

[S P...olicy there is dela > - writin, « th e modifications to the master copy on server. M ...ications are ... in cache and then later time ...

I L advan,a 8c <>f this J dooe on n. ten f accesses complete in less time as writes are made to cache. Second, the data which may be over... ...lten prior

I fatten. The limitation thi ; "Mate needs to crashes then "-™ data are lost and, 'Client reliable. hcnce less

There a,e variations of delayed ... choice is w fluSh a block « soon as " Mlc >- One I ...ected from the chenfs cache. This altern " "» be ,'jood performance, hut some blocks c.""""" lc « I dial's cache a long time before they ... exist in the [ng server. A negotiation between this eh*""* 11 back wriK-through policy is to scan the cach ...olce and ,be Rivals and to flush blocks that have iJ " rcEular siare the most recent scan. So far one " mo<1,fi ed 0 delayed write is to write data back to the " when the file is closed. This write-on-close policy is ysed in Andrew File System (AFS),

L Consistency

I Client machine always use cached data for accesses fcjj is consistent with master copy at server. If client amines that its cached copy is out of date, then it should it the up-to-date copy of data. Following two oacbes are used.

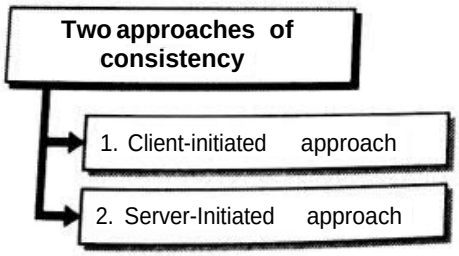


Fig. C6.7 : Approaches of consistency

f Client-Initiated approach

I client starts a validity check in which it contacts / server. In this check, client checks consistency 1 * hed data with the data in master copy. decided by I resulting consistency semantics is can I / cy of validity checking. Validity c ess {0 I * carried out prior to *every* access *or* on ...ile.

- Every access accompanied by a validity check is delayed, compared with an access served by the cache
- h? """"hers and, checks can be started at ...vals. The load on ... frequency of the va 'dity checj nt WDr ,Q Spends on

2. ...erver-initiated approach

The server maintains the records of the files or portion of files that each client cat... ...s. Server immediate... take "*°™>«lencr Whe "«r n detects a potential If two different clients in conflicting "xxles cache a Bte ,hen there is possib... A notif ication yiflnc ™sista>cy. OP "' ed ' « the intend**.*™ " «ben a file is ...ified for every open ... of write mode must ** Whenever server rt... cts that file bas bcen opened s iniultaneously in rnnft fl ,Cllng modes - « can take action b ? disabling cach c "*g is d iMbled g lbe" p m T i<:ular n,e When operation becomes active m « le <*

6-9 5 A Comparison of Caching and Remote Service

Q. ... racing? dlflefenCe betwe8n rem ote service and

Sr. No.	Caching	Remote Service
I.	It allows to serve remote accesses locally so that they can be faster as local accesses	Remote access is handled across the network, so slower compared to caching.
2.	It reduces the network traffic, server load and improves scalability as server is contacted occasionally.	It increases network traffic, server load and degrades performance.
3.	In case of caching, for transmission of big chunks of data, overall network overload required is at lower side compared » remotew ------	In case of remote service, transmission of series of responses to specific requests involve higher network overhead as compared to caching.

Sr. No	Caching	Remote Service
4.	*Caching is better* in case of infrequent writes but if writes are frequent then mechanism used to overcome consistency problem incur large overhead in terms of performance. network traffic, and server load.	In remote service, there is always communication between client and server to have a master copy consistent with client's cached copy.
5.	Caching is better option for machines with disk or large main memory.	If machines are diskless and with small main memory then remote service should be carried out.
6.	In case of caching, the lower-level inter-machine interface is different from the upper-level user interface.	The remote service concept is just an extension of the local file-system interface across the network.

Syllabus Topic : Stateful Versus Stateless Service

6JO Stateful Versus Stateless Service

- Client access the remote files from server. If server keeps track about each file being accessed by each client then the service is called stateful. If server is simply provides requested blocks of data to client and does not keep track about how client makes use of them then service is called stateless.

Stateful File Service

- First client must cany out open( ) operation on file prior to accessing it. Server the access information about file from disk and store in main memory.
- Server then gives unique connection identifier to client and opens the file for client. This identifier is used by client to access the file throughout the session.
- On closing of the file. or by garbage collection mechanism, the server should free the main memory space used by client when it is no longer active. The mformatton maintained by server regarding client is

edm fault toterance, AFS uses stateful approach

Stateless File Service

- in this case server does not keep any information in main memory about opened files. Here, each request identifies the file. There is no need to open arid close the file with operations open( ) and closef).
- Each file operation in this case is not a part of session but it is on its own separately. Closing the file at last is also a remote operation, NFS is example of stateless file service approach.
- Stateful service gives more performance. As connection identifier is used to access information maintained in main memory, disk accesses are reduced. Moreover, as server knows that file is open for sequential access, then read ahead next block improves the performance. In stateful case, if server crashes then recovery of volatile state information is complex. A dialog with client is used by recovery protocol. Server also needs to know about crash of client to free the memory space allocated to it. All the operation underway during crash should be aborted.
- In case of stateless service, the effect of server crash and recovery is invisible. Client keeps retransmitting the requests if server crash and no response from it.

Syllabus Topic : File Replication

6J1 File Replication

- Replicating the file on many machines improves availability. If nearest replica is used then service time also reduces. The replica of the same file should be kept on failure independent machines so that availability of one replica is not affected by availability of remaining other replica. Replication of files should be hidden from users.
- It is the responsibility of naming scheme to a replicated file name to a particular replica. Higher levels should remain invisible from existence of replicas.
- At lower-level different lower-level names are used for different replicas. Replication control such as determination of the degree of replication and placement of replicas should be provided to higher levels. As one replica updates then from user's point of view, the changes should be reflected in other copies as well.

6.12 Distributed Synchronization

centralized synchronization nie .

I Lj to distributed environment also C can be

6.12.1 Mutual Exclusion

f gxpiafnddferent algorithms for mijh ual exclusion in

distributed system.

. U T rcnt algorithm, Io achlew J

n in distnbuted environment. There are su mUtual

her *of processes* in system nnntbered froni , "PP0Se "

each process has its own processor .

6.12.1(A) Centralized Approach

Q. Explain centralized algorithm for mutual exclusion in distributed system.

ID this algorithm, one of the proceS5 acts as coordinator. Each process want to enter in critical section sends request message to coordinator. If process then receives reply message from coordinator then proceed to enter in critical section. Once process finish execution in critical section, it sends release message to coordinator and continue its execution.

After receiving the request, first coordinator checks whether critical Section is empty or not. If empty then coordinator sends reply message to requesting process and process enters in critical section. If other process is already in critical section then request of requesting process is queued.

When coordinator receive release message from process exiling from critical section, one of the processes from queue is chosen to enter in critical section based on some scheduling algorithm. If FIFO strategy is used then no starvation can occur. Three messages request, reply and release are used in this algorithm.

6.12.1(B) Fully Distributed Approach

1 Explain distributed algorithm for mutual exclusion in distributed system.

-yolved in decision

fa this algorithm all processes arc mv x* th efliC*' in critical

to allow the requesting process ter in cr itical

xwion. Process Pi which want to cn olher

L *«ion. sends **request (Pi, TS)** message to

(n _ (J Distributed Systems

processes, ltldudin 8 itself, in this request

me5 *ge **TS is a time** SLarT1p 'Seating the time at

*hich request nie ssage, ot |ler SCnt‹ receiving this request

imtnc diate|vn r ;/ r<1CeSS may 5410(1 reply message

If P ess w ydefCTit

P' °ce$s i$ alre d request message from other

Queued. When . cr l‹ical section, then request is

reply lo reou ‹ oOr, ,eS oul critical section, it sends

reply from a|R nfi process ‹ If requesting process gets

critical section " ' pr o cesses lt can enter in

Whethe reply is sent

"'Pends ' on following factors. immOlja dy » it deferred

1 [f process receiving request message is in critical sec •on then it defers the reply to requesting process.

2. If process receiving request message do not want to enter in critical section then it immediately sends reply to requesting process.

3. If process receiving request message also want enter in critical section then, it compare its time stamp with time stamp of requesting process. If its time stamp is greater than lime stamp of incoming request then it immediately sends request message. Otherwise the reply is postponed.

In this algorithm, mutual exclusion and no deadlock is guaranteed. There is no starvation as time stamp is used to ensure FCFS strategy to serve. Exact 2(n-l) number of messages arc required to exchange to enter in critical section which is less compared to if processes act independently and concurrently.

Following are the limitations of this algorithm.

1. Processes should know names of all other processes in group. There is problem when new process joins the group. In this case, processes must receive names of all other processes in group. Suppose, request and reply messages are in transit and new process joins the group, then problem definitely will arise.

2. If anyone process fails, then requesting process wi jl not receive reply message from it. The monitoring of all processes slate is required and notification regarding fail process need to be sent to all other processes in group. Thts is required so thaX P esS sh0UW Send thC req «eSt lo failed process.

3. h is necessary to pause (he processes frequently that have not taken entry in their critical section. This is required to guarantee other processes that they want to enter the critical section.

- And hence, this algorithm is better suited to small and stable sets of cooperating processes.

6.12.1(C) Token-Passing Approach

- In this algorithm, a token which is a special type of message circulated in the system. Process acquiring token can enter in the critical section which ensures a single process in critical section at a time. In this algorithm, processes in the system are assumed as logically organized in a ring structure. This logical arrangement can be based on some parameter, for example, integer value of IP address of the machine on which process is running.
- Token circulates around the ring and process which want to enter in critical section holds the token and enter in critical section. After the process exits its critical section, the token is circulated again in ring. In this scheme, process exiting from critical section passes the token to its neighbour. In this algorithm, unidirectional link guarantees the freedom from starvation.
- One message per entry is required to enter in critical if all the processes want to enter in critical section, if no one want io enter in critical section then infinite number of messages may require. Limitations of this algorithm are:
  1. Token may lost. In this case election needs to be called to generate new token.
  2. Failure of any process requires establishing a new ring.

Syllabus Topic : Concurrency Control

6.13 Concurrency Control

- This section explains concurrency control schemes for distributed environment. The management of the transactions or subtrabsactions accessing database on the local site is earned out by transaction manager of distributed database system.
- Transaction can be local or it can be part of the transaction running at other sites. The concurrency control is the responsibility of transaction manager. i$_t$ also maintains log for future recovery purpose.

6.13.1 Locking Protocols

- Two-phase locking protocol also works for distributed environment. In this case implementation of loc$_k$ manager differ compare to single system environment

6.13.1(A) Non replicated Scheme

- This scheme is used when data is not repli$_{Catfd}$ in system. In this scheme, there is a local lock manager $_{Un}$ each site to manage the lock and unlock requests for those data items stored in that site. In order to lock the data item X on site Si, process simply send message to lock manager on that site.
- If X is locked in incompatible mode then the request is delayed until that request can be granted. Once it has been decided that the lock request can be approved, the lock manager sends a message back to the initiator showing that the lock request has been granted.
- The implementation of this scheme is easy. It needs only two message transfers for handling lock requests and one message transfer for handling unlock requests But, deadlock handling is more complex.

6.13.1(B) Single-Coordinator Approach

- This scheme is used when data is replicated. In this scheme, single lock manager resides on particular selected site Si. For locking and unlocking the data item, all request are sent to this site. Any transaction want io lock ihe data item, sends lock request to Si and lock manager decides whether to grant lock without delay or not.
- It immediately lock request is granted, then lock manager instantly sends message to the site from where request was sent.
- If request not granted, then it is delayed till it is granted and again message is sent to the requesting site.
- The transaction can read the data item from any one of the sites at which a replica of the data item resides.
- In case of write operation, all the sites having replica should take part in the writing.

Advantages

- Implementation is simple as only two messages are required to handle lock request and one message to handle unlock request. Algorithms used to handle

deadlock on stn 8 Je system can be used " -a 1Qck and unlock request are from same's ite.

pfM****"

.. sing* '<"* mana ger reside On "articular «, site si , site Si may be come ... cessing of a*1 *e request, then . """ck due ? > » * done or processing must stOp ' f* ,s recovery another solution to avoid bottle neck is t. mnk... z j; rta tor approach. Lock manager run on several site and each handles lock and unlock requests for subset of data items. Since several sites are involved, dead lock handling is complex.

6.13 1(C) M a J°rity Protocol

Q. Describe majority protocol jn distributed system.

I%is approach is modification of nonrcnli ' srheoie. In "heme lock manager is pra ,c?. da,a iile . The locks for all data or replica of data ™ sX the site are controlled by lock manager inning on that site-

If data item is replicated on n number of sitcs transaction want to lock this data item. Then it is necessary to send the lock request to more than one-half of the n sites. Lock manager decides whether to grant the lock or should response be delayed. The transaction does not operate on replicated data item until it has successfully obtained a lock on a majority of the replicas of data item.

Advantages

- Avoids the drawback of central control approach.

Disadvantages

- implementation is complex. It requires 2(n/2 + 1) messages for handling lock requests and (n/2 + 1) messages for handling unlock requests. As many sites are involved, different types of implementation is required for handling the deadlock.

613.1(D) Biased Protocol

Tfe working of biased protocol is close to the a Jori ty Protocol. In biased protocol, requests for s hared locks get more favorable action than requests for exclusive :r which ,Ocks - Each site contains a lock manag5 items stored at that *** the locks *for* all the data ed to site. Shared and exclusive locks are man g dif fertnt manner.

Shi ared locks

To acquire the I ock " onsite hT d n« 'X' x" X , ,, an5Mion s 'Wiy E ,«u.lv. X « " P,iCa of that data item x.~

When □ transaction wants to lock data item x. u requests a lock on X h ... o lngare Plica of X lock " ager at each site

If * e *que st b priest is delayed tv eRntCd lhen response to that on read operation ■ S SC eme ,m POses less overhead advantage E ive _ mpared lo majority protocol. This there compared t reSU ,tS When morc reads arc ■s incurred a Writts « extra overhead on write handling com i S many Sltes involved, deadlock m Ecomplexi Ues exists.

6.13/1 (E) Primary Copy

In this aDDroach, for each data item X there is a primary cop ... tv - zv wmcn ls kept on exactly on one site . For this data 1 m X, this is called as primary site of X. When transaction needs to lock data item X, it should request lock st primary site of X, If the request is not granted then response to that request is delayed.

The concurrency control for replicated and nonreplicated data is handled in the same way. Implementation is straightforward. If primary site of data item X fails then although replica is available, data item X is accessible.

6.13.2 Timestamping

A unique timestamp is given to each transaction based on which serialization order is decided.

6.1 3.2(A) Generation of Unique Timestamps

Centralized and distributed methods are used to generate the unique timestamp. In centralized approach, single site distributes the time stamps which are generated using logical counter or site's own clock.

In the distributed method, every site generates a local timestamp which is unique. For this, either a logical counter or the local, clock is used. The global unique timestamp >s further 8enerated by concatenat,on of ,he local unique timestamp with the site identifier. While loc « ,ln » site identifier is kept in the least "X i«sito « that the global tifflesamps

generated in one site are not / * venerated in another site.

- Any site should not generate local timestamps at a faster rar eomparel o other sites. Otherwise. faster reater than limeslwn ps ste ·s nmestw", „ s« mechanism 5hould be generatcu there to guarantee mat, X hat umestarups will be generated >» fair manner across the system.
- If system clocks are used to generate local timestamps then fair generation is possible if all clocks tick « — speed. Any faster or slower clock may affect fair generation of timestamps.

6.13.2(B) Timestamp-Ordering Scheme

- In centralized approach, there can be cascading rollbacks if no mechanism is applied to stop a transaction from reading a data item value that is not until now committed. The basic timestamp scheme experiences through the unwanted property that conflicts between transactions are resolved through rollbacks, rather than through wait.
- This problem can be resolved by buffering the different read and write operations (delay) until a time when we arc guaranteed that these operations can happen without causing aborts.

Syllabus Topic : Deadlock Handling

6,14 Deadlock Handling <

- Deadlock prevention, deadlock avoidance and deadlock detection algorithms for single system can be extended to distributed system.

6.14.1 Deadlock Prevention and Avoidance

Q. Explain deadlock prevention and avoidance in distributed system.

- In distributed system, resource-ordering deadlock-prevention technique can be used by simply defining a global ordering among the system resources. A unique number is assigned to each resource in system and process request the resource with unique number i only if it is not holding a resource with a unique number greater than i.
- Banker's algorithm can also be used in distributed system provided one of the processes is selected as process which keeps all the information required to carry out the ban dlouk-P-ven " 'Xgh t ordering dea overhead and simple to banker's algorithm involves large overhead, its impleme« wUon ls simple.
- The number me5S |afge , Hence, it may become as banker may ult, the banker's algorithm does not bottleneck M al use in & distribdted systenv aDuear to be of praccre .
- Padlock-prevention scheme that considers a A " Xng approach with resource preemption "X any deblock situation that may happen in a can h red system Consider only a ease of a single Xce of each resource type. A unique priority number is given to each process in order <0 control the prevention.
- These priorities decide whether process Pi should for Pi If Pi has highest pnomy j ---------- for Pi This method prevents deadlocks since, for every edge from Pi to Pj in the wait-for graph, Pi has a higher priority than *Pj.* Thus, a cycle cannot exist. This scheme lead to starvation as always lower priority processes will be rolled back. Following are the two deadlock-prevention schemes based on timestamps.

The wait-die scheme

- This scheme is based on a non-preemptive approach. When process *Pi* requests a resource presently held by *Pj, Pi* is permitted to wait only if it has a lesser timestamp than does Pj (that is. *Pi* is older than *Pj).* Otherwise, *Pi;* is rolled back (dies).

The wound-wait scheme

- This scheme is based on a preemptive approach and is a complement to the wait-die scheme. When process Pi requests a resource presently held by Pj, Pi is permitted to wait only if it has a larger timestamp than does Pi (that is, Pi is younger than Pj). Otherwise, Pi is rolled back (Pj is wounded by Pi).
- If rolled back process does not get new timestamp, then above both schemes can avoid starvation. The rolled back process will always receive smaller timestamp as time always increases.
- In the wait-die approach, an older process has to wait for a younger one to free its resource. Therefore, the older the process gets, the more it tends to wait. On the other hand, in the wound-wait approach, an older process never waits for a younger process. In both schemes needless rollbacks may take place.

Deadlock Detection

IfP' , , efflP* dcfiDeS reS « Urce aJ l«a tio » s|ate Ic
I Wait- esource o f « Ch ,yPC '* exist then <We in Wai,
[' »»» denotes deadlock. Each site maintains it.
1 S' ait-for graph n<XleS « w «« fM graph
i X P"**®*' ,0Ca S «« o» Which are ronning
indi... on other »» * o n «l rCqucs,,n * or holding the
resource local to that that sue.

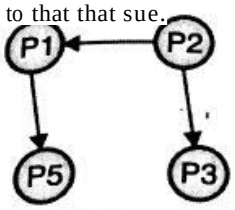
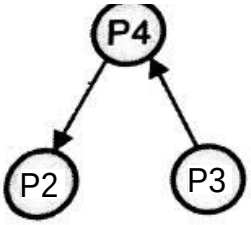


fig.6-141 : Wait-for 8ra Ph at site 8,



F jg,6.14*2: Wait-for graph at site S_2

I ifpioc® ess pi js running on site S_t and it request for the t| jen pj sen(js message to site S_2 and resource at site S_2,

1 edge Pi to Pj gets addcd in wait , for graph of site Sz

tfJoca j wait-for graph contains cycle then deadlock has I urred If there is no cyc1eS in local wait , for graphs I of all sites does no(mean dcadlock not o ccurrcd ,

w can be a deadlock. The union of the all local rat-far graph is taken on some site to obtain global wait-for graph.

If this global wait-for graph contains cycle then system isin deadlocked state,

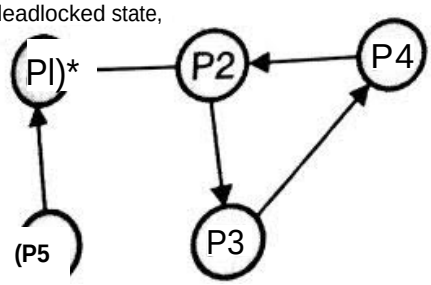


Fig. 043 : Global wa ,t-for graph after union of wait-for graphs at site Si and Si

Wai i-forgraphs can be organized m followin g ways.

6.14.2(A) Centralized Approach

Q. Explain centralized algorithm for deadlock detection in distributed system.

- In this approach, there is a deadlock detection t rdinator that maintains a global wail-for graph wi ich is union of local wait-for graphs from different siteR. 1 he two types of graph are generated- One is real graph which is no illustrates the real but unidentified state of the system at any point in time. The constructed graph is an approximation generated by the coordinator during the execution of its algorithm. The reported result by generated graph should be correct. On existence of deadlock, it should report properly and if reported system is certainly in deadlocked state.
- The updating of global wait-for graph is carried out if following events take place in system.
  - On addition or deletion of edge in local wait-for graphs.
  - Periodically, after number of changes.
  - On invocation of deadlock detection algorithm.
- On invocation of dcadlock detection algorithm, if cycle exist in global wait-for graph then process responsible is rolled back and local sites are also informed by coordinator about the victim. The local sites also roll back the roll out process. On addition or deletion of edge in local wait-for graph, a local site should immediately convey information to coordinator. Coordinator then updates these modifications in global wait-for graph.
- Periodically when local sites sends messages to coordinator about any changes in local wait-for grapns then false cycle may exists in global wait-for graph.

  This happens due to delay incurred in reaching the messages from different local sites to coordmator. Hence, although there is no deadlock, recovery wi initiated. To overcome this problem, messages should be assigned with identifiers or timestamp.

6.14.2(B) Fully Distributed Approach

Q. Explain distributed algorithm for deadlock d system.

- In ,h,S «P Each sitt builds a wait-for graph indicating a de,eCtl o n ;h; total graph, tending on the dynamic part C f the system. The thought is that, if a be baviof ot

deadlock exists, a cycle will appear in at least one of the partial graphs.

- In «, approach, each she — its local graph. In each local wait-for graph, we „ for P Edge Pi to P„ in graph indicates Pi J* wai i g dX hern in another site being held by any process. In the same way, edge P„ >□Pj in S«ph indicates thai. process at another site is waiting to acquire a resource presently being held by *Pj* in this iocal sue.
- These modified wail-for graphs arc then used for detection of deadlock. If local wait-for graph nas cycle which does not contains P_M then deadlock that involves local processes of that site is occurred. This deadlock is handled locally at that site only.
- If local wait-for graph has cycle which contains P„ then there is possibility of distributed deadlock in which processes from different sites are involved. In this case distributed deadlock detection algorithm is invoked by site whose wait-for graph contains P„.

6.15 Exam Pack (Review Questions)

<sr Syllabus Topic : Distributed Operating System

Q. Define distributed system. *(Refer section 6.1)*

Q. What are its characteristics? Explain. *(Refer section 6.1.1)*

Q. What are objectives behind building the distributed system? *(Refer section 6.1.2)*

Q. Explain different types of distributed operating systems. *(Refer section 6.2)*

☞ Syllabus Topic : Network Based OS

Q. What are the reasons behind process migration? *(Refer section 6.2.2(C))*

☞ Syllabus Topic : Network Structure

Q. Explain LAN type of network. *(Refer section 6.3.1)*

Q. Explain WAN type of network. *(Refer section 6.3.2)*

☞ Syllabus Topic : Network Topology

Q. Explain different topology for network with its advantagesand disadvantages. *(Refer section 6.4)*

☞ Syllabus Topi Tcommumcation Structure .

Q. What is naming? How names are resolved? Explain. *(Refer section 6.5.1)*

O. Explain different routing strategies. *(Refer section 6.5.2)*

Q. What are the different connection strategies used in netwoi'k communication? *(Refer section 6.5.4)*

☞ Syllabus Topic : Communication Protocols

Q. Explain different communication protocols. *(Refer section 6.6)*

☞ Syllabus Topic : Distributed File Systems

Q. Explain working of distributed file system (DFS). *(Refer section 6.7)*

☞ Syllabus Topic : Naming and Transparency

Q. Explain various naming schemes. *(Refer section 6.8.2)*

☞ Syllabus Topic : Remote File Access

q. What is difference between remote service and caching ? *(Refer section 6.9.5)*

<r Syllabus Topic : Distributed Synchronization

Q. Explain centralized algorithm for mutual exclusion in distributed system. *(Refer section 6.12.1(A))*

Q. Explain distributed algorithm for mutual exclusion in distributed system. (Refer section *6.12.1(B))*

☞ Syllabus Topic : Concurrency Control

Q. Describe majority protocol in distributed system. *(Refer section 6.13.1 (C))*

☞ Syllabus Topic : Deadlock Handling

Q. Explain deadlock prevention and avoidance in distributed system. *(Refer section 6.14.1)*

Q. Explain centralized algorithm for deadlock detection in distributed system. *(Refer section 6.14.2(B))*

Q. Explain distributed algorithm for deadlock detection in distributed system. *(Refer section 6.14.2(C))*

□□□