DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(JUNE 2016-NOV 2016)

OPERATING SYSTEMS

V SEMESTER

SYLLABUS

CS T51 OPERATING SYSTEMS

UNIT – I

Introduction: Mainframe Systems – Desktop Systems – Multiprocessor Systems – Distributed Systems – Clustered Systems - Real Time Systems – Hardware Protection – System Components – Handheld Systems - Operating System Services – System Calls – System Programs – Process Concept – Process Scheduling – Operations on Processes – Cooperating Processes – Inter-process Communication.

UNIT – II

Threads: Overview – Threading issues - CPU Scheduling – Basic Concepts – Scheduling Criteria – Scheduling Algorithms – Multiple-Processor Scheduling – Real Time Scheduling - The Critical- Section Problem – Synchronization Hardware – Semaphores – Classic problems of Synchronization – Critical regions – Monitors.

UNIT – III

System Model – Deadlock Characterization – Methods for handling Deadlocks -Deadlock Prevention – Deadlock avoidance – Deadlock detection – Recovery from Deadlocks - Storage Management – Swapping – Contiguous Memory allocation – Paging – Segmentation – Segmentation with Paging.

UNIT – IV

Virtual Memory – Demand Paging – Process creation – Page Replacement – Allocation of frames – Thrashing - File Concept – Access Methods – Directory Structure – File System Mounting – File Sharing – Protection.

UNIT – V

File System Structure – File System Implementation – Directory Implementation – Allocation Methods – Free-space Management. Kernel I/O Subsystems - Disk Structure – Disk Scheduling – Disk Management – Swap-Space Management.
Case Study: The Linux System, Windows.

Text Books:

1. Abraham Silberschatz, Peter Baer Galvin and Greg Gagne, “Operating System Concepts”, John Wiley & Sons (ASIA) Pvt. Ltd, Seventh edition, 2005.
2. Harvey M. Deitel, Paul J. Deitel, and David R. Choffnes, “Operating Systems”, Prentice Hall, Third edition, 2003.

Reference Books:

1. William Stallings, Operating Systems: Internals and Design Principles**,** Prentice -Hall of India, Sixth edition, 2009.
2. Gary J. Nutt, “Operating Systems: A Modern Perspective”, Addison-Wesley, Second edition, 2001.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SUBJECT: OPERATING SYSTEMS **SUBJECT CODE: CS T51**

UNIT I

Introduction: Mainframe Systems – Desktop Systems – Multiprocessor Systems – Distributed Systems – Clustered Systems - Real Time Systems – Hardware Protection – System Components – Handheld Systems - Operating System Services – System Calls – System Programs – Process Concept – Process Scheduling – Operations on Processes – Cooperating Processes – Inter-process Communication.

2 Marks

1. What is an Operating system? (APR'15)

An operating system is a program that manages the computer hardware. It also provides a basis for application programs and act as an intermediary between a user of a computer and the computer hardware. It controls and coordinates the use of the hardware among the various application programs for the various users.

2. Why is the Operating System viewed as a resource allocator & control program?

A computer system has many resources - hardware & software that may be required to solve a problem, like CPU time, memory space, file-storage space, I/O devices & soon. The OS acts as a manager for these resources so it is viewed as a resource allocator. The OS is viewed as a control program because it manages the execution of user programs to prevent errors & improper use of the computer.

3. What is the Kernel?

A more common definition is that the OS is the one program running at all times on the computer, usually called the kernel, with all else being application programs.

4. What are Batch systems?

Batch systems are quite appropriate for executing large jobs that need little interaction. The user can submit jobs and return later for the results. It is not necessary to wait while the job is processed. Operators batched together jobs with similar needs and ran them through the computer as a group.

5. What is the advantage of Multiprogramming?(APR '12)

Multiprogramming increases CPU utilization by organizing jobs so that the CPU always has one to execute. Several jobs are placed in the main memory and the processor is switched from job to job as needed to keep several jobs advancing while keeping the peripheral devices in use. Multiprogramming is the first instance where the Operating system must make decisions for the users. Therefore they are fairly sophisticated.

6. What is an Interactive computer system?

Interactive computer system provides direct communication between the user and the system. The user gives instructions to the operating system or to a program directly, using a keyboard or mouse, and waits for immediate results.

7. What do you mean by Time-sharing systems?

Time-sharing or multitasking is a logical extension of multiprogramming. It allows many users to share the computer simultaneously. The CPU executes multiple jobs by switching among them, but the switches occur so frequently that the users can interact with each program while it is running.

8. What are multiprocessor systems & give their advantages?(APR '14)(APR '12)(APR'15)

Multiprocessor systems also known as parallel systems or tightly coupled systems are systems that have more than one processor in close communication, sharing the computer bus, the clock and sometimes memory & peripheral devices. Their main advantages are

• Increased throughput

• Economy of scale

• Increased reliability

9. What are the different types of multiprocessing?

Symmetric multiprocessing (SMP): In SMP each processor runs an identical copy of the Os & these copies communicate with one another as needed. All processors are peers. Examples are Windows NT, Solaris, Digital UNIX, OS/2 & Linux.

Asymmetric multiprocessing: Each processor is assigned a specific task. A master processor controls the system; the other processors look to the master for instructions or predefined tasks. It defines a master-slave relationship. Example SunOS Version 4.

10. What is Distributed system? (APR '12, NOV '15)

A distributed system is a collection of independent Computers that appears to its users as a single coherent system. Examples of Distributed Systems The Internet: net of nets global access to "everybody" (data, service, other actor; open ended),enormous size (open ended).

11. What are the advantages of distributed processing? (APR '13)

1. Higher performance: Applications can execute in parallel and distribute the load across multiple servers.
2. Collaboration: Multiple applications can be connected through standard distributed computing mechanisms.
3. Higher reliability & availability: Applications or servers can be clustered in multiple machines.
4. Scalability: By deploying reusable distributed components on powerful servers.
5. Extensibility: Dynamic (re)configuration of applications distributed across network.
6. Higher productivity & lower development cycle time: Breaking up large problems into smaller ones, these individual components can be developed by smaller development teams in isolation.

7. Reuse: Services that can potentially be used by multiple client applications.

8. Reduced cost: Due to the reuse of once developed components that are accessible over the network

12. What are the disadvantages of distributed processing?

Software: Complexity of programming distributed systems

Networking: The network can saturate or cause other problems

Security: Easy access also applies to secret data

13. What is graceful degradation?

In multiprocessor systems, failure of one processor will not halt the system, but only slow it down. If there are ten processors & if one fails the remaining nine processors pick up the work of the failed processor. This ability to continue providing service is proportional to the surviving hardware is called graceful degradation.

14What is Dual-Mode Operation?

The dual mode operation provides us with the means for protecting the operating system from wrong users and wrong users from one another. User mode and monitor mode are the two modes. Monitor mode is also called supervisor mode, system mode or privileged mode. Mode bit is attached to the hardware of the computer to indicate the current mode. Mode bit is '0' for monitor mode and '1' for user mode.

15. What are privileged instructions?

Some of the machine instructions that may cause harm to a system are designated as privileged instructions. The hardware allows the privileged instructions to be executed only in monitor mode.

16. How can a user program disrupt the normal operations of a system?

A user program may disrupt the normal operation of a system by

- Issuing illegal I/O operations
- By accessing memory locations within the OS itself
- Refusing to relinquish the CPU

17. How is the protection for memory provided?

The protection against illegal memory access is done by using two registers. The base register and the limit register. The base register holds the smallest legal physical address; the limit register contains the size of the range. The base and limit registers can be loaded only by the OS using special privileged instructions.

18. What are the various OS components?

The various system components are

- Process management
- Main-memory management
- File management
- I/O-system management
- Secondary-storage management
- Networking
- Protection system
- Command-interpreter system

19. What is a process?

A process is a program in execution. It is the unit of work in a modern operating system. A process is an active entity with a program counter specifying the next instructions to execute and a set of associated resources. It also includes the process stack, containing temporary data and a data section containing global variables.

20. What is a process state and mention the various states of a process?(APR '12)

As a process executes, it changes state. The state of a process is defined in part by the current activity of that process. Each process may be in one of the following states:

- New
- Running
- Waiting
- Ready
- Terminated

21. What is process control block?

Each process is represented in the operating system by a process control block also called a task control block. It contains many pieces of information associated with a specific process. It simply acts as a repository for any information that may vary from process to process. It contains the following information:

- Process state
- Program counter
- CPU registers
- CPU-scheduling information
- Memory-management information
- Accounting information
- I/O status information

22. What are the use of job queues, ready queues & device queues?

As a process enters a system, they are put into a job queue. This queue consists of all jobs in the system. The processes that are residing in main memory and are ready & waiting to execute are kept on a list called ready queue. The list of processes waiting for a particular I/O device is kept in the device queue.

23. What is meant by context switch? (NOV'14)

Switching the CPU to another process requires saving the state of the old process and loading the saved state for the new process. This task is known as context switch. The context of a process is represented in the PCB of a process.

24. What are the factors that need to be considered to determine the degree of multiprogramming in a system?

The two factors that need to be considered are:

1. The overheads in context switching may become excessive.
2. With excessive multiprogramming the response times may become unacceptable.

25. What is the difference between the idle and blocked state of a process?

In idle state, the process is waiting for the processor to become free so that it can execute. In blocked state, the process has been put out from the running state by the processor due to some I/O.

26. When a process requests for I/O, how many process switches take place?

Two. In the first switch, the process to be switched is taken out and the scheduler starts executing. Then the next process is brought to execution. So there are two process switches.

27. Differentiate tightly coupled systems and loosely coupled systems?

S.No	Loosely coupled systems	Tightly coupled systems
1	Each processor has its own local memory	Common memory is shared by many processors
2	Each processor can communicate with other all through communication lines	No need of any special communication lines

28. What is real time system? (APR 2011) (NOV 2011)

A real time system has well defined, fixed time constraints. Processing must be done within the defined constraints, or the system will fail. It is often used as a control device in a dedicated application.

29. What do you mean by system calls? (APR '12)

System calls provide the interface between a process and the operating system. When a system call is executed, it is treated as by the hardware as software interrupt.

30. What is scheduler?

A process migrates between the various scheduling queues throughout its life time. The OS must select processes from these queues in some fashion. This selection process is carried out by a scheduler.

31. What is independent process?

A process is independent it cannot affect Or be affected by the other processes executing in the system. Any process does not share data with other process is a independent process.

32. What is co-operative process?

A process is co-operating if it can affect or be affected by the other processes executing in the system. Any process that share data with other process is a co-operating process.

33. What is the benefits of OS co-operating process?

- Information sharing.
- Computation speed up.
- Modularity.
- Convenience.

34. How can a user program disturb the normal operation of the system?

- Issuing illegal I/O operation.
- By accessing memory locations within the OS itself.
- Refusing to relinquish the CPU.

35. What is the use of inter process communication.

Inter process communication provides a mechanism to allow the co-operating process to communicate with each other and synchronies their actions without sharing the same address space. It is provided a message passing system.

36. What are the five major activities of an operating system with regard to process management? (APR '14)

The five major activities are:

- The creation and deletion of both user and system processes
- The suspension and resumption of processes
- The provision of mechanisms for process synchronization
- The provision of mechanisms for process communication
- The provision of mechanisms for deadlock handling

37. Describe the differences between symmetric and asymmetric multiprocessing?(APR 2011)

Symmetric processing treats all processors as equals; I/O can be processed on any of them.

Asymmetric processing designates one CPU as the master, which is the only one capable of performing I/O; the master distributes computational work among the other CPUs.

- advantages: Multiprocessor systems can save money, by sharing power supplies, housings, and peripherals. Can execute programs more quickly and can have increased reliability.
- disadvantages: Multiprocessor systems are more complex in both hardware and software. Additional CPU cycles are required to manage the cooperation, so per-CPU efficiency goes down.

38. Define Information Sharing?(NOV '12)

Information sharing describes the exchange of data between various organizations, people and technologies.

There are several types of information sharing: Information shared by individuals .Information shared by organizations. Information shared between firmware/software. The advent of wide distributed networks, intranets, cross-platform compatibility, application porting, and standardization of IP protocols have all facilitated the huge growth in global information sharing.

39. List the types of server systems?(NOV '13)

1. File Server
2. Database server
3. Transaction server.

40. What is the use of fork and exec system calls?

Fork is a system call by which a new process is created. Exec is also a system call, which is used after a fork by one of the two processes to replace the process memory space with a new program.

41. What's the difference between multitasking, multiprogramming & multiprocessing?

Multiprogramming- Jobs to be executed are loaded into a pool. Some number of those jobs are loaded into main memory, and one is selected from the pool for execution by the CPU. If at some point the program in progress terminates or requires the services of a peripheral device, the control of the CPU is given to the next job in the pool. As programs terminate, more jobs are loaded into memory for execution, and CPU control is switched to another job in memory. In this way the CPU is always executing some program or some portion thereof, instead of waiting for a printer, tape drive, or console input .

Multiprocessing - the simultaneous execution of two or more programs or instruction sequences by separate CPUs under integrated control

multitasking System - the concurrent or interleaved execution of two or more jobs by a single CPU.

Multiuser System - a computer system in which multiple terminals connect to a host computer that handles processing tasks.

42. Define PCB and write its contents?(APRIL 2011)

Process Control block is used for storing the collection of information about the Processes and this is also called as the Data Structure which Stores the information about the process. The information of the Process is used by the CPU at the Run time. The various information which is stored into the PCB as follows:

1) Name of the Process.
2) State of the Process. Means Ready, Active, Wait.
3) Resources allocated to the Process
4) Memory which is provided to the Process.
5) Scheduling information.
6) Input and Output Devices used by the Process.

7) Process ID or a Identification Number which is given by the CPU when a Process Request for a Service.

43. What is meant by independent process and cooperating process? (APR'12)

Independent process cannot affect or be affected by the execution of another process. Cooperating process can affect or be affected by the execution of another process.

44. Name any three components of the system. (Nov '15)

1. Process Management
2. Main Memory Management
3. File Management
4. I/O System Management
5. Secondary Management
6. Networking
7. Protection System
8. Command-Interpreter System

11 Marks

1. Explain computer systems which are categorized according to the number of processors used? (APR '12)

An Operating system is a program that controls the execution of application programs and acts as an interface between the user of a computer and the computer hardware.

It provides the basis for applications programs and acts as an intermediary between user of a computer and the computer hardware.

A more common definition is that the operating system is the one program running at all times on the computer (usually called the kernel), with all else being applications programs.

An Operating system is concerned with the allocation of resources and services, such as memory, processors, devices and information. The Operating System correspondingly includes programs to manage these resources, such as a traffic controller, a scheduler, memory management module, I/O programs, and a file system.

COMPONENTS OF A COMPUTER SYSTEM

(Explain briefly about components of computer system.)

Every general purpose computer consists of the hardware, operating system, users, application programs.

The hardware consists of memory, CPU, ALU, I/O devices, peripheral device and storage device. System program consists of compilers, loaders, editors, OS etc. The application program consists of business program, database program.

Hardware – Central processing Unit(CPU), Memory, Input/Output (I/O) Devices.

Operating Systems – Controls and co-ordinate the hardware among application and user.

The fig. 1.1 shows the conceptual view of a computer system

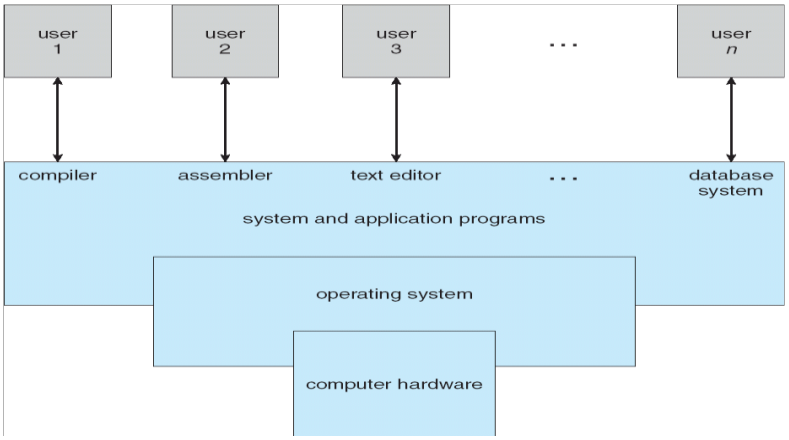


Fig 1.1 Conceptual view of a computer system

Application programs – Word processors, Spread sheets, Computers , Web browsers.

Operating system can be replaced from two view points:

➔ the user view

➔ the system view

User view:

User view of the computer varies by the interface being used.

Ease of use → Users sit at the front of a PC. Designed for one user to maximize work,

Resource utilization → Users at the terminal. Assures all available CPU time, memory , I/O are used efficiently.

Work stations →users at workstations connected to network users share resources(files, compile, print servers).

System view:

From computers point of view the operating system is a program that is most intimate with the hardware.

Resource allocator → operating system acts as manager of resources, decides how to allocate to programs so that it can operate efficiently.

Control program→control program manages the execution of user programs to prevent errors and improper use of the computer.

Every computer must have an operating system to run other programs. The operating system and coordinates the use of the hardware among the various system programs and application program for a various users. It simply provides an environment within which other programs can do useful work.

The operating system is a set of special programs that run on a computer system that allow it to work properly. It performs basic tasks such as recognizing input from the keyboard, keeping track of files and directories on the disk, sending output to the display screen and controlling a peripheral device.

OS is designed to serve two basic purposes:

1. It controls the allocation and use of the computing system‘s resources among the various user and tasks.
2. It provides an interface between the computer hardware and the programmer that simplifies and makes feasible for coding, creation, debugging of application programs.

The operating system must support the following tasks. The tasks are :

1. Provides the facilities to create, modification of program and data files using and editor.
2. Access to the compiler for translating the user program from high level language to machine language.
3. Provide a loader program to move the compiled program code to the computer‘s memory for execution.
4. Provide routines that handle the details of I/O programming .

I/O System Management

I/O System Management

The module that keeps track of the status of devices is called the I/O traffic controller. Each I/O device has a device handler that resides in a separate process associated with that device.

The I/O subsystem consists of

1. A memory management component that includes buffering, caching and spooling.

2. A general device driver interface.

Drivers for specific hardware devices.

The computer processor also called CPU (Central Processing Unit) is one of the major components in computer systems. Usually, it is referred as the brain of computer, because it is the place where all the computing process (calculation, comparison and logical decisions) is performed. This dictates your computer performance is mainly depend on the type of processor installed in your computer. The more the computing power of a processor, the higher and faster is the processing capacity of that computer.

Currently, wide varieties of computer processor types are available on the market. Both **Intel** and **AMD**, the largest microprocessor manufacturers in the world, have introduced several types of computer processors.

Each processor type is different from the other in performance and technology. Unless you know the features behind each type of processor, choosing the right type could be a challenging task when purchasing, upgrading or building your own computer.

Groups of computer processor types

Based on the overall performance and the type of work designed for, computer processors are grouped into three main categories. There could be other applicable methods to classify computer processors. However, this grouping will help to us ease the complication when selecting a processor both for desktop and mobile computers. Computer processors are grouped into the following three divisions. This grouping doesn't include processors for Server and Workstation PCs.

1. **High-End Processors**

Processors in this group are designed for intensive applications, since the programs require high processing power. What makes high-end is the advanced microprocessor technology incorporated with these types of processors. Other than the normal usage of computer applications, if you are into Statistical analysis, intensive graphics, creating and editing professional videos, extreme 3D gaming, multitasking and multi-threading application then you should choose a computer installed with a processor categorized in high-end group.

Both Intel and AMD introduced processors categorized as high-end. The latest **Intel Core I Series processors**: **i3, i5, i7, i9** and **AMD Phenom** family processors are among high-end CPU's. The prices of high-end processors are a bit higher comparing with the rest CPU's.

2. **Mid-End computer processor types**

Mi-range processors are meant for middle intensive tasks. Beyond the standard work you do with mid-range processors, you can do tasks such as basic 3D Gaming, casual photo editing, home video creating, and multimedia applications.

Some of the common mid-range Intel processors include **Intel Core 2 Quad**, **Intel Core 2 Extreme**, **Intel Core 2 Duo, Intel Pentium Dual Core and Intel Core Duo/Solo**. **AMD Phenom 1 X3**, **AMD Turion family** and **AMD Athlon** family processors are categorized into mid-range CPUs.

3. **Basic or economy computer processor types**

As the name implies, processors in this group are low performing CPUs with cheap price. If you are into non-intensive tasks such as simple gaming, office applications, internet browsing, email and common graphics, then your choice of processor grouped as budget CPUs. AMD Sempron, AMD Athlon Neo and Intel Atom, Intel Centrino, Centrino Duo and Celeron are grouped into economy processors.

2. Write in detail about Mainframe systems (8) (APR '14)

Mainframe systems

Mainframe computer systems were the first computers used to tackle many commercial and scientific applications. It growths from batch systems to time shared systems.

-Batch Systems

-Multiprogrammed Systems

-Time sharing Systems

1.Batch Systems

Input devices – Card readers, tape drivers.

Output devices – Line printers, tape drivers, card punches.

User does not interact directly with the system.

Job is prepared (program, data, control information)

Job is usually in the form of punch cards.

Control transferred automatically from one job to another.

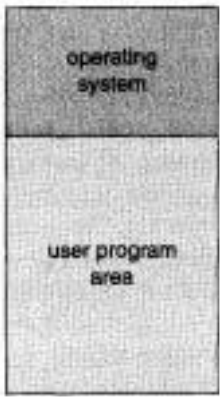
- Batches jobs with similar needs to speedup processing
- After completion of one job then only the next job will be executed
- Task of os

- It automatically transfers control from one job to another
- Demerit -It executes one job at a time

CPU idle during I/O

I/O devices idle during CPU busy

Memory Layout for a Simple Batch System



Memory layout for Multiprogramming batch system

To speed up processing; Jobs with similar needs are batched together.

2. The multiprogrammed systems

- Keeps more than one job in memory simultaneously.
- Jobs are organized so that CPU always has one to execute.
- OS keys several jobs simultaneously in memory.
- OS picks and begins execute one job.
- When a job performs I/O, OS switches to another job.
- It increases CPU scheduling and utilization.
- All jobs enter the system kept in the job pool on a disk scheduler brings jobs from pool into memory.
- Selecting the job from job pool is known as **Job scheduling.**
- Once the job loaded into memory ,it is ready to execute ,if several jobs are ready to run at the same time ,the system must choose among them ,making this decision is **CPU scheduling**.

Merit :

- ✓ CPU is never idle

DEMERITS :

CPU is often idle.

Speeds of i/o devices are slower.

Executes only one job at a time.

3.Time-Sharing Systems–Interactive Computing

- Also called as multi tasking system.
- Extension of multiprogramming.
- Multi user ,single processor OS
- Time sharing or multi tasking allows more than one program to run concurrently.
- Multitasking is the ability to execute more than one task at the same time.a task is a program.
- In multitasking only one CPU is involved ,the CPU switches from one program to another so quickly that it gives the appearance of executing all of the program runs at the same time.
- Response time should be short.

Time sharing systems should provide the following mechanisms:

→File system.

→Disk management

→Concurrent Execution

→Job synchronization and communication

- Two types of multitasking

1. Pre emptive

   Time slice given to CPU by OS
2. Cooperative or non preemptive

   In this each program can control the CPU for as long as it needs CPU.

2. Write sort notes on Desktop Systems (4)

- Personal computers – computer system dedicated to a single user.
- I/O devices – keyboards, mice, display screens, small printers.
- User convenience and responsiveness.
- Can adopt technology developed for larger operating system.

- Often individuals have sole use of computer and do not need advanced CPU utilization or protection features.
- May run several different types of operating systems (Windows, MacOS, UNIX, Linux)

3. Explain Multiprocessor systems or Parallel Systems (6) (APR 2011)

- Also known as Parrallel systems or tightly coupled systems.
- Multiprocessor systems with more than one CPU in close communication.
- Tightly coupled system – processors share memory and a clock; communication usually takes place through the shared memory.

Advantages of parallel system:

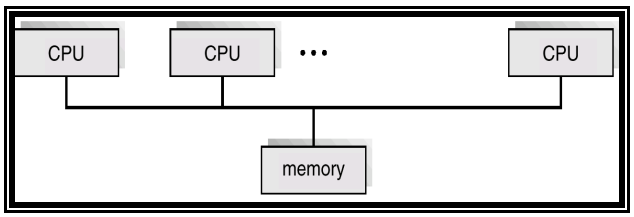
- Increased throughput- By increasing the no of processors work is done in less time.
- Economical – multiprocessor system can save more money because they can share periperal, mass storage and power supply.
- Increased reliability- If one processors fields then the remaining processors will share the work of the failed processors. This is known as **graceful degradation or fault tolerant**

TYPES:

- Symmetric multiprocessing (SMP)
  - Each processor runs an identical copy of the operating system.
  - Copies communicate with one another.
  - All processors are peers; no master – slave relationship.
  - Allow processors and resources to be shared.
  - Many processes can run at once without performance deterioration.
  - Most modern operating systems support SMP

Eg: All modern OS – Window NT; UNIX, LINUX, Solris.

Symmetric Multiprocessing Architecture



- Asymmetric multiprocessing
  - Each processor is assigned a specific task; master processor schedules and allocates work to slave processors.
  - More common in extremely large systems

- Follows master – slave relationship.
- Master processor schedules and allocates work to the slave processors.
- Common in extremely large systems.
- Eg: Sun OS(Version4).

4. Write about Distributed Systems (6) (APR 2011)(APR '12)

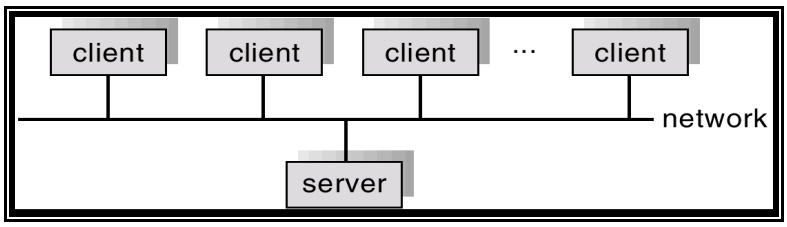
- Depends on networking for their functionality.
- Distribute the computation among several physical processors.
- Loosely coupled system – each processor has its own local memory; processors communicate with one another through various communications lines, such as high-speed buses or telephone lines.
- Requires networking infrastructure.
- LAN(LOCAL AREA NETWORK) – within a room/building.
- WAN(WIDE AREA NETWORK) – between buildings , cities.
- MAN(METROPOLITAN AREA NETWORK) – link buildings within a city.
- Bluetooth – short distance communication.
- Advantages of distributed systems.
    - Resources Sharing
    - Computation speed up – load sharing
    - Reliability
    - Communications

1. Client server systems

In centralized system acts as a server system to satisfy request generate by client systems it is the server system is categorized as follows

1. **Compute server system-** client can send request to which they execute the actions and send back result to the client.
2. **File server system** – It provides a file system interface where clients can create, update, read and delete files.

General Structure of Client-Server



3. Peer to Peer System

The growth of the computer networks lead to the internet and WWW. Virtually all modem PCs and workstations are capable of running a web browser for accessing hypertext documents on the web. Several operating systems now include the web browsers, electronic mail, remote login and file transfer clients and servers.

- Network connectivity is an essential component of a computer system.
- These processors do not share memory as a clock.

5. Write short notes on Clustered Systems and real time &hand held systems (8) (NOV 13)

Clustered systems

Clustered systems gather together multiple CPUs to accomplish computational work and runs on cluster nodes.

- Clustering allows two or more systems to share storage.
- Provides high reliability.
- Each node monitors one or more of others.
- If the monitored machine fails monitoring machine can take ownership of its storage and restart the application running on failed machine.
- The failed machine can remain down, but user's and clients experience interrupter.
- **Asymmetric clustering**: one server standby while the other runs the application. The standby server monitors the active machine.
  - One machine is in hot standby mode
  - Other machine runs the application.
  - The standby machine monitors the active machine.
  - If server fails the standby host becomes active server.
- **Symmetric clustering**: all N hosts are running the application and they monitor each other.
  - Two or more hosts runs application and monitors each other.
  - More efficient.
- **Parrallel clusters:**

  Allow multiple hosts to access the same data on shared storage.

  Eg: Oracle Parallel Server.

Real-Time Systems

- Often used as a control device in a dedicated application such as controlling scientific experiments, medical imaging systems, industrial control systems, and some display systems.
- Well-defined fixed-time constraints.
- Processing must be done within the defined constraints or the system will fail.
- Real-Time systems may be either hard or soft real-time.
- Hard real-time:
  - Secondary storage limited or absent, data stored in short term memory, or read-only memory (ROM)
  - System generates that critical tasks be completed on time.
  - Conflicts with time-sharing systems, not supported by general-purpose operating systems.
- Soft real-time
  - Limited utility in industrial control of robotics
  - Restrictive type of system.
  - Requires advanced operating system features.
  - Useful in applications (multimedia, virtual reality) requiring advanced operating-system features.

Handheld Systems

- Hand held systems includes Personal Digital Assistants (PDAs) such as palm, pocket PCs/ cellular telephone.
- The varies issues of the them are as follows:
  - Limited space with is a challenging to make the system.
  - Limited memory space and so in turn limited size of the system.
  - It has Slow processors.
  - Web clipping is used to display the content of the web pages because of small display screens.

6. Write short notes on Hardware Protection (6)

Need for hardware protection:

Early os were called resident monitors; the OS began to perform many functions especially I/O .

To improve system utilization ,OS began to share system resources. Protection is needed for any shared resource.

- Dual-Mode Operation
- I/O Protection
- Memory Protection
- CPU Protection

Dual-Mode Operation

- Sharing system resources requires operating system to ensure that an incorrect program cannot cause other programs to execute incorrectly.

- Provide hardware support to differentiate between at least two modes of operations.

1. User mode – execution done on behalf of a user.

2.Monitor mode (also kernel mode or system mode) – execution done on behalf of operating system.

Mode bit added to computer hardware to indicate the current mode: monitor (0) or user (1).

- If bit is set to 1 it is in user mode. If bit is set to 0 it is monitor mode.
- Operating system automatically change the bit value according to the program execution.

When an interrupt or fault occurs hardware switches to monitor mode.

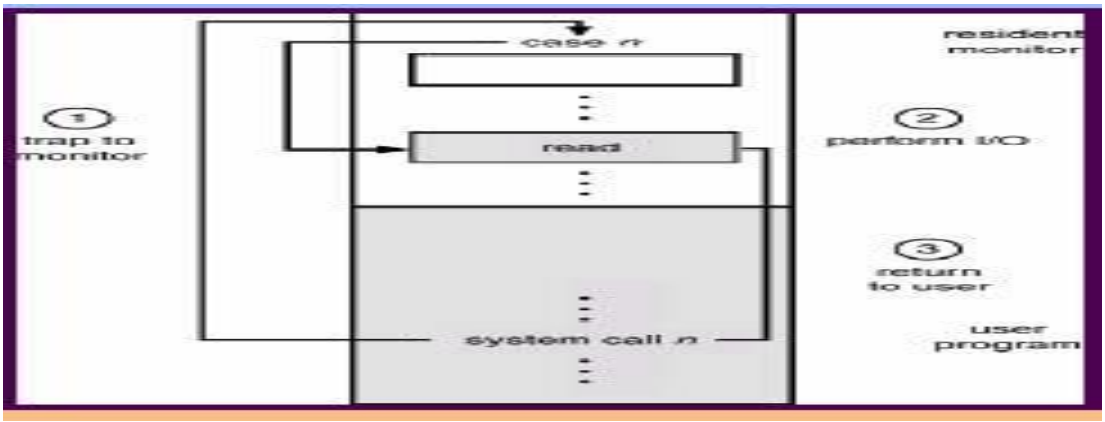
- Initially the hardware starts in monitor mode at system boot time.
- Whenever trap or interrupt occurs, the hardware switches from user mode to monitor mode.

Merits:

- The dual mode operation providers with means of protecting os from current users.
- This protection is accomplished by privileged instructions that can be executed only in monitor mode.

I/O Protection

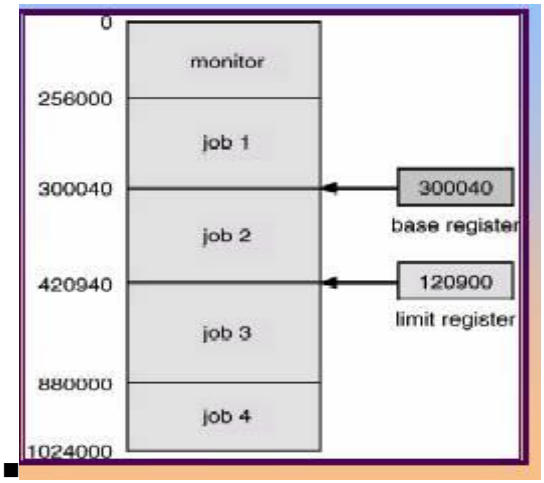
- All I/O instructions are privileged instructions.
- Must ensure that a user program could never gain control of the computer in monitor mode (I.e., a user program that, as part of its execution, stores a new address in the interrupt vector).



Use of A System Call to Perform I/O

Memory Protection

Use of A Base and Limit Register



- Must provide memory protection at least for the interrupt vector and the interrupt service routines.
- Also protect interrupt service routines in os from modification.
- In order to have memory protection, add two registers that determine the range of legal addresses a program may access:
  - Base register – holds the smallest legal physical memory address.
  - Limit register – contains the size of the range

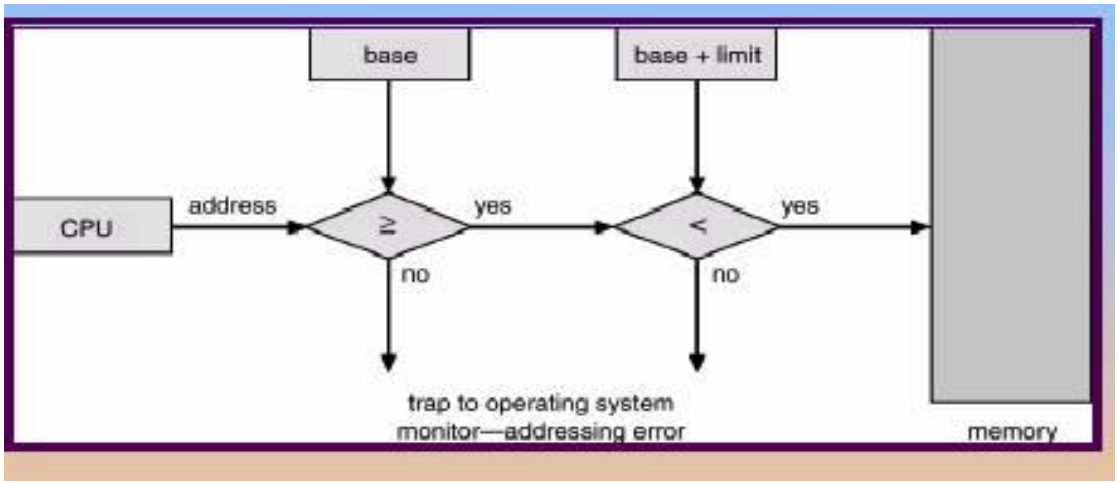
Eg: If base register holds 300040 and limit register is 120900, then program can legally access all addresses from 300040 through 420940. This scheme prevents user program from modifying the code of either os or other users.

FIG: Base and a limit register (logical address space)

Any attempt by a program executing in user mode to access monitor memory or other users memory results in a trap to the monitor which is a fatal error.

- Memory outside the defined range is protected.

Hardware Address Protection



Hardware Protection

- When executing in monitor mode, the operating system has unrestricted access to both monitor and user's memory.
- The load instructions for the base and limit registers are privileged instructions.

CPU Protection

- Ensure that the os maintains control.
- Timer can be net to interrupt the computer after aspecified period.
- Periods may be fixed or variable.
- Timer – interrupts computer after specified period to ensure operating system maintains control.
  - Timer is decremented every clock tick.
  - When timer reaches the value 0, an interrupt occurs.
- Timer commonly used to implement time sharing.
- Timer also used to compute the current time.
- Load-timer is a privileged instruction.

7. List out and discuss operating System Components? (NOV '12)(APR'15)

9. Process Management
10. Main Memory Management
11. File Management
12. I/O System Management
13. Secondary Management
14. Networking
15. Protection System
16. Command-Interpreter System

Process Management

- A process is a program in execution. A process needs certain resources, including CPU time, memory, files, and I/O devices, to accomplish its task.
- The operating system is responsible for the following activities in connection with process management.
  - Process creation and deletion.
  - Process suspension and resumption.
  - Provision of mechanisms for:
    - ✓ process synchronization
    - ✓ process communication

Main-Memory Management

- Memory is a large array of words or bytes, each with its own address. It is a repository of quickly accessible data shared by the CPU and I/O devices.
- Main memory is a volatile storage device. It loses its contents in the case of system failure.
- The operating system is responsible for the following activities in connections with memory management:
  - Keep track of which parts of memory are currently being used and by whom.
  - Decide which processes to load when memory space becomes available.
  - Allocate and deallocate memory space as needed.

File Management

- A file is a collection of related information defined by its creator. Commonly, files represent programs (both source and object forms) and data.
- The operating system is responsible for the following activities in connections with file management:
  - File creation and deletion.
  - Directory creation and deletion.
  - Support of primitives for manipulating files and directories.
  - Mapping files onto secondary storage.
  - File backup on stable (nonvolatile) storage media.

I/O System Management

- Manage and control I/O operations and I/O devices.
- The I/O system consists of:
  - A buffer-caching system
  - A general device-driver interface
  - Drivers for specific hardware devices

Secondary-Storage Management

- Since main memory (primary storage) is volatile and too small to accommodate all data and programs permanently, the computer system must provide secondary storage to back up main memory.
- Most modern computer systems use disks as the principle on-line storage medium, for both programs and data.
- The operating system is responsible for the following activities in connection with disk management:
  - Free space management
  - Storage allocation
  - Disk scheduling

Networking (Distributed Systems)

- A distributed system is a collection processors that do not share memory or a clock. Each processor has its own local memory.
- The processors in the system are connected through a communication network.
- Communication takes place using a protocol.
- A distributed system provides user access to various system resources.
- Access to a shared resource allows:
  - Computation speed-up
  - Increased data availability
  - Enhanced reliability

Protection System

- Protection refers to a mechanism for controlling access by programs, processes, or users to both system and user resources.
- The protection mechanism must:
  - distinguish between authorized and unauthorized usage.
  - specify the controls to be imposed.
  - provide a means of enforcement.

Command-Interpreter System

- Many commands are given to the operating system by control statements which deal with:
  - process creation and management
  - I/O handling
  - secondary-storage management
  - main-memory management
  - file-system access
  - protection

- networking

- The program that reads and interprets control statements is called variously:
  - command-line interpreter
  - shell (in UNIX)

  Its function is to get and execute the next command statement.

8. Write notes on Operating System Services (5) (APR '12, NOV '15)

OS provides services to programs and to users.

Operating system providers services to programs and to users. These services are as follows.

Services to user:

- Program execution – system capability to load a program into memory and to run it.
- I/O operations – since user programs cannot execute I/O operations directly, the operating system must provide some means to perform I/O.
- File-system manipulation – program capability to read, write, create, and delete files.
- Communications – exchange of information between processes executing either on the same computer or on different systems tied together by a network. Implemented via shared memory or message passing.
- Error detection – ensure correct computing by detecting errors in the CPU and memory hardware, in I/O devices, or in user programs.

Services to programs:

- Resource allocation – allocating resources to multiple users or multiple jobs running at the same time.
- Accounting – keep track of and record which users use how much and what kinds of computer resources for account billing or for accumulating usage statistics.
- Protection – ensuring that all access to system resources is controlled.

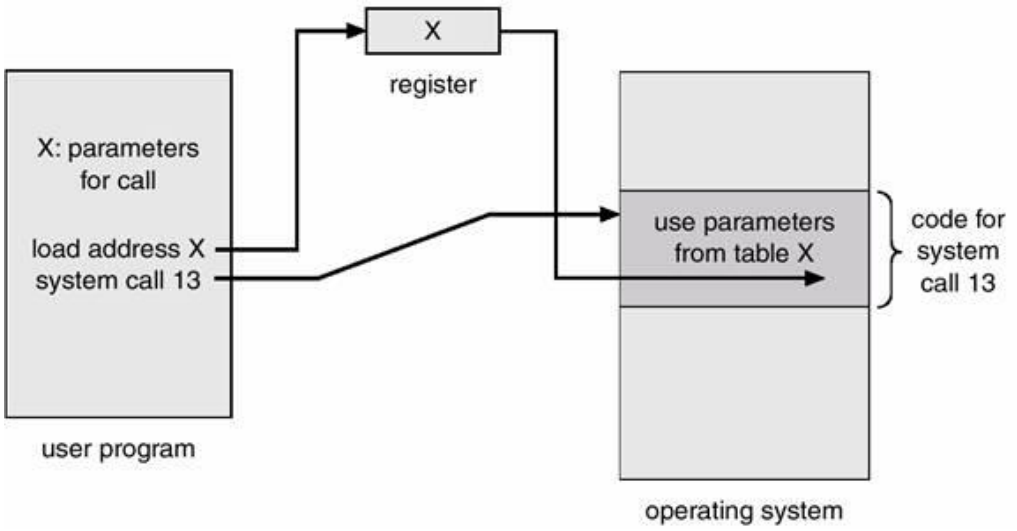
9. Discuss and detail about System Calls and system programs (5) (APR '12)(NOV '12)(NOV'14)

- Any request made by the user to OS is called **system call** for the resources.
- System calls provide the interface between a running program and the operating system.
  - Generally available as assembly-language instructions.
  - Languages defined to replace assembly language for systems programming allow system calls to be made directly (e.g., C, C++)
- Three general methods are used to pass parameters between a running program and the operating system.
  - Pass parameters in registers.
  - Store the parameters in a table in memory, and the table address is passed as a parameter in a register.
  - Push (store) the parameters onto the stack by the program, and pop off the stack by operating system.

Passing of parameters as a table.

System calls provide an interface between the process an the operating system. System calls allow user-level processes to request some services from the operating system which process itself is not allowed to do. In handling the trap, the operating system will enter in the kernel mode, where it has access to privileged instructions, and can perform the desired service on the behalf of user-level process. It is because of the critical nature of operations that the operating system itself does them every time they are needed. For example, for I/O a process involves a system call telling the operating system to read or write particular area and this request is satisfied by the operating system.

system programs provide basic functioning to users so that they do not need to write their own environment for program development (editors, compilers) and program execution (shells). In some sense, they are bundles of useful system calls.



Types of System Calls

- Process control

  Load and execute, create process, terminate process, end, abort, get/set process attributes, wait, get process attribute, set process attribute, wait for time, wait event, signal event, allocate and free memory etc.

- File management

  Open, close, create process, delete, read, write, reposition, get/set file attribute.

- Device management

  Request, release, read, write, reposition

  Read, write, reposition

  Get device attributes, set device attributes

  Logically attach or detach devices.

- Information maintenance

  Get/set time, date

  Get system data, set system data

  Get process, file or device attribute

  Set process, file or device attributes

- Communications

  Open/close connection, send/receive messages.

  Transfer status information.

  Attach or detach remote devices.

10. Explain Different type of system calls? (NOV 12)(APR'15)

Process Control

A running program needs to be able to halt its execution either normally (end) or abnormally **(abort).** If a system call is made to terminate the currently running program abnormally, or if the program runs into a problem and causes an error trap, a dump of memory is sometimes taken and an error message generated. The dump is written to disk and may be examined by a debugger to determine the cause of the problem. Under either normal or abnormal circumstances, the operating system must transfer control to the command interpreter. The command interpreter then reads the next command. In an interactive system, the command interpreter simply continues with the next command; it is assumed that the user will issue an appropriate command to respond to any error. In a batch system, the command interpreter usually terminates the entire job and continues with the next job. Some systems allow control cards to indicate special recovery actions in case an error occurs. If the program discovers an error in its input and wants to terminate abnormally, it may also want to define an error level. More severe errors can be indicated by a higher-level error parameter. It is then possible to combine normal and abnormal termination by defining a normal termination as error at level 0. The command interpreter or a following program can use this error level to determine the next action automatically.

Types of system calls- Process Control

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

File management

- create file, delete file
- open, close
- read, write, reposition
- get file attributes, set file attributes

Device management

- request device, release device
- read, write, reposition
- get device attributes, set device attributes
- logically attach or detach devices

Information maintenance

- get time or date, set time or date
- get system data, set system data
- get process, file, or device attributes
- set process, file, or device attributes

Communications

- create, delete communication connection
- send, receive messages
- transfer status information
- attach or detach remote devices

File Management

System call requires the name of the file and perhaps some of the file's attributes. Once the file is created, we need to open it and to use it. We may also read, write, or reposition (rewind or skip to the end of the file, for example). Finally, we need to close the file, indicating that we are no longer using it. We may need these same sets of operations for directories if we have a directory structure for organizing files in the file system. In addition, for either files or directories, we need to be able to determine the values of various attributes, and perhaps to reset them if necessary. File attributes include the file name, a file type, protection codes, accounting information, and so on. At least two system calls, get file attribute and set file attribute, are required for this function. Some operating systems provide many more calls.

Device Management

A program, as it is running, may need additional resources to proceed. Additional resources may be more memory, tape drives, access to files, and so on. If the resources are available, they can be granted, and control can be returned to the user program; otherwise, the program will have to wait until sufficient resources are available.

Files can be thought of as abstract or virtual devices. Thus, many of the system calls for files are also needed for devices. If the system has multiple users, however, we must first request the device, to ensure exclusive use of it. After we are finished with the device, we must release it. These functions are similar to the open and close system calls for files.

Once the device has been requested (and allocated to us), we can read, write, and (possibly) reposition the device, just as we can with ordinary files. In fact, the similarity between I/O devices and files is so great that many operating systems, including UNIX and MS-DOS, merge the two into a combined **file-device structure.** In this case, 1/0 devices are identified by special file names.

Information Maintenance

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

Communication

There are two common models of communication. In the message-passing model, information is exchanged through an interprocess-communication facility provided by the operating system. Before communication can take place, a connection must be opened. The name of the other communicator must be known, be it another process on the same CPU, or a process on another computer connected by a communications network. Each computer in a network has a host name, such as an IP name, by which it is commonly known. Similarly, each process has a process name, which is translated into an equivalent identifier by which the operating system can refer to it. The **get hostid** and **get processid** system calls do this translation. These identifiers are then passed to the general-purpose **open** and **close** calls provided by the file system, or to specific **open connect ion** and **close connect ion** system calls, depending on the system's model of communications. The recipient process usually must give its permission for communication to take place with an **accept connect ion** call. Most processes that will be receiving connections are special purpose daemons-systems programs provided for that purpose. They execute a **wait for connection** call and are awakened when a connection is made. The source of the communication, known as the *client*, and the

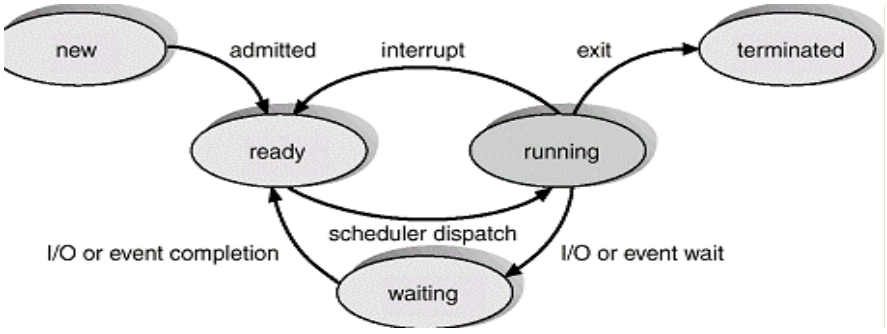
receiving daemon, known as a *server*, then exchange messages by **read message** and **write message** system calls. The **close connect ion** call terminates the
Communication.

11.Write short notes on Process (7) (APR 12)(NOV 13)

- Process is nothing but a program in execution. Process can create sub process called child .the process of creating as many child processes from a parent process is called spawning. Every process has process states. Current activity of the process is called process state.
- A process includes:
  - program counter
  - stack
  - data section

Process State

- As a process executes, it changes state
  - new: The process is being created.
  - running: Instructions are being executed.
  - waiting: The process is waiting for some event to occur.
  - ready: The process is waiting to be assigned to a process.
  - terminated: The process has finished execution.



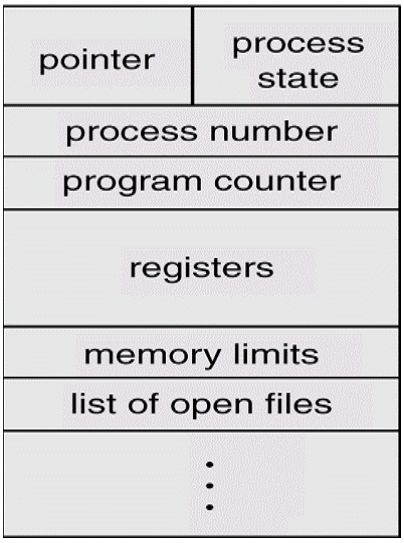
Process control block:

- Each process is represented in the operating system by a processes control block(PCB).
- It is also called as task control block(TCB).

It is a data structure maintained by OS for every process control. Information associated with each process.

- Process state
- Program counter
- CPU registers

- CPU scheduling information
- Memory-management information
- Accounting information
- I/O status information



12.Write about process scheduling (6) (NOV'14)

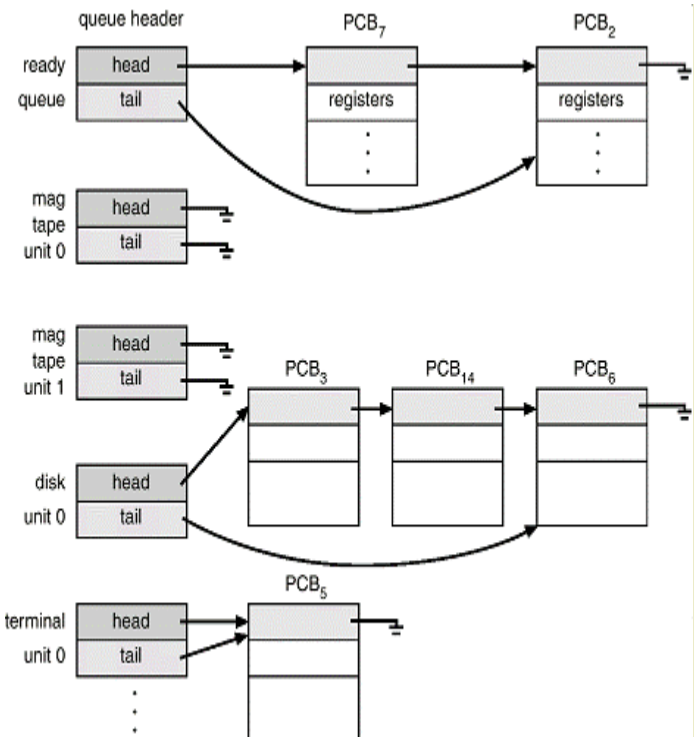
Process Schelduling:

- A microprocessor system can have only one runnig process.
- If more processors exist the rest must wait until the CPU is free and can be rescheduled.

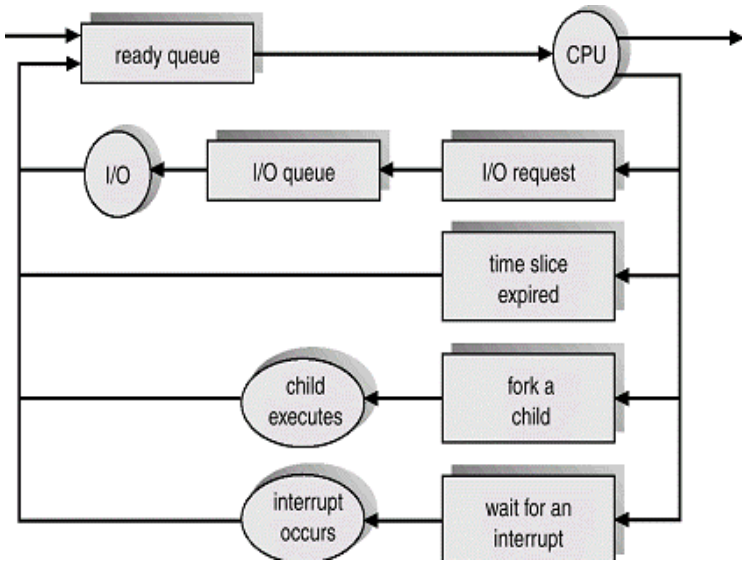
Process Scheduling Queues

- Policy which decides the CPU to switch from one process to other is called **scheduling**.this process method os called process scheduling .it is represented by queuing diagram.
- Process migrates between various queues.
- Job queue – set of all processes in the system.
- Ready queue – set of all processes residing in main memory, ready and waiting to execute.
- Device queues – set of processes waiting for an I/O device.
- Process migration between the various queues.

Ready queue and various I/O queues



Representation of process scheduling



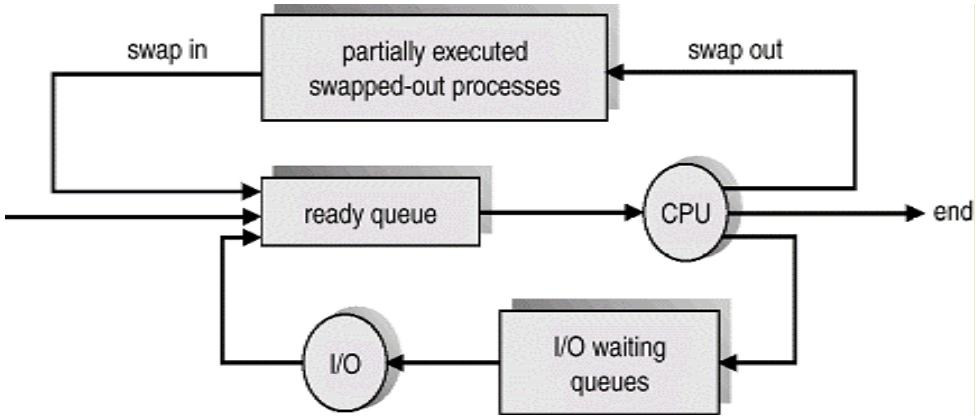
- A process continues the cycle until it terminates at which time it is removed from all queues and has its PCB and resources deallocated.

SCHEDULER:

- A process migrates between the various scheduling queues throughout its lifetime.
- The operating system must select, for scheduling purpose processors from these queues in a certain order.
- The selection process is carried out by the scheduler.

- Long-term scheduler (or job scheduler) – selects which processes should be brought into the ready queue. It is also calles as job scheduler.
- Short-term scheduler (or CPU scheduler) – selects which process should be executed next and allocates CPU. It is also calles as CPU scheduler.
- Medium term scheduler-acts as an intermediate between long term and short term .it uses swapping technique to balance the utilization of CPU.it removes process from memory and later bring it into memory(swapping)
- Short-term scheduler is invoked very frequently (milliseconds) ⇒ (must be fast).
- Long-term scheduler is invoked very infrequently (seconds, minutes) ⇒ (may be slow).
- The long-term scheduler controls the degree of multiprogramming.
- Processes can be described as either:
  - I/O-bound process – spends more time doing I/O than computations, many short CPU bursts.
  - CPU-bound process – spends more time doing computations; few very long CPU bursts.

Addition of medium term scheduling



Context switch

- When CPU switches to another process, the system must save the state of the old process and load the saved state for the new process.
- Context-switch time is overhead; the system does no useful work while switching.
- Time dependent on hardware support.

13.Write about operation on Processes (7) (NOV 13) (NOV'14, NOV '15)

Process Creation

- Fork()-system call used to create child process identical to parent.
- Parent process create children processes, which, in turn create other processes, forming a tree of processes.

- Resource sharing
  - Parent and children share all resources.
  - Children share subset of parent's resources.
  - Parent and child share no resources.
- Execution
  - Parent and children execute concurrently.
  - Parent waits until children terminate.
- Address space
  - Child duplicate of parent.
  - Child has a program loaded into it.
- UNIX examples
  - fork system call creates new process
  - exec system call used after a fork to replace the process' memory space with a new program.

Process Termination

- Process executes last statement and asks the operating system to decide it (exit).
  - Output data from child to parent (via wait).
  - Process' resources are deallocated by operating system.
- Parent may terminate execution of children processes (abort).
  - Child has exceeded allocated resources.
  - Task assigned to child is no longer required.
  - Parent is exiting.

Operating system does not allow child to continue if its parent terminates.

Cascading termination.

Cooperating Processes

- Independent process cannot affect or be affected by the execution of another process.
- Cooperating process can affect or be affected by the execution of another process
- Advantages of process cooperation
  - Information sharing
  - Computation speed-up
  - Modularity
  - Convenience

14. **Explain how cooperating process can communication in direct and indirect communication? (APR '13)**

The processes executing in the operating system may be either **independent processes** or **cooperating processes**. Cooperating processes require an inter-process communication mechanism to communicate with each other. Principally, communication is achieved through two schemes: shared memory and message passing. The shared-memory method requires communicating processes through

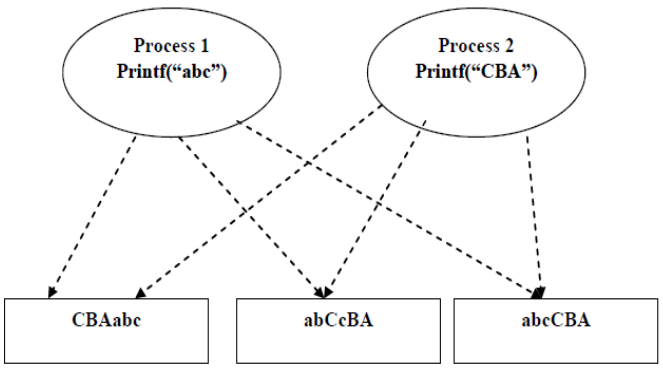
The use of these shared variables. In a shared-memory system, the responsibility for providing communication rests with the application programmers: the operating system needs to provide only the shared memory. The responsibility for providing communication may rest with the operating system itself. These two schemes are not mutually exclusive and can be used simultaneously within a single operating system.

Co-operating process is a process that can affect or be affected by the other processes while executing. If suppose any process is sharing data with other processes, then it is called co-operating process. Benefit of the co-operating processes are :

1. Sharing of information
2. Increases computation speed
3. Modularity
4. Convenience

Information sharing in co – operating processors:

Co-operating processes share the information: Such as a file, memory etc. System must provide an environment to allow concurrent access to these types of resources. Computation speed will increase if the computer has multiple processing elements are connected together. System is constructed in a modular fashion. System function is divided into number of modules.



Behavior of co-operating processes is nondeterministic i.e. it depends on relative execution sequence and cannot be predicted a priori. Co-operating processes are also Reproducible

. For example, suppose one process writes"ABC", another writes"CBA" can get different outputs, cannot tell what comes from which. Which process output first "C" in "ABCCBA". The subtle state sharing that occurs here via the terminal. Not just anything can happen, though. For example,"AABBCC" cannot occur.

15. Explain Producer-Consumer Problem (8)

- A common paradigm for cooperating processes, A producer process produces information that is consumed by a consumer process.
- Have a buffer of items filled by the producer and emptied by the consumer
  - unbounded-buffer no limit on the size of the buffer.
  - bounded-buffer assumes that there is a fixed buffer size.
- Buffer is a shared memory .Producer and consumer run concurrently and must be synchronized. In **bounded buffer**, the consumer must wait if the buffer is empty and the producer must wait if the buffer is full.
- In **Unbounded Buffer**, the consumer may have to wait for new items, but the producer can always produce new items.

Bounded-Buffer – Shared-Memory Solution

- Shared data

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

- Solution is correct, but can only use BUFFER_SIZE-1 elements

Bounded-Buffer – Producer Process

```
ItemnextProduced;
while (1)
 {
while (((in + 1) % BUFFER_SIZE) == out)       ;
/* do nothing */
buffer[in] = next Produced;
in = (in + 1) % BUFFER_SIZE;
```

```
}
Bounded-Buffer – Consumer Process
ItemnextConsumed;
while (1) {
while (in == out);
 /* do nothing */
next Consumed = buffer[out];
out = (out + 1) % BUFFER_SIZE;
}
```

16. Write about Inter process Communication (8) (APR '14, NOV '15)

Interprocess communication (IPC)

IPC provides a mechanism to allow processors to communicate and to synchronize their actions without sharing the same address space.

IPC is useful in a distributed environment.

- Cooperating process can communicate through facility provided
  i. By application programmer
  ii. By os [IPC]
- Mechanism for processes to communicate and to synchronize their actions.
- Message system – processes communicate with each other without resorting to shared variables.
- IPC facility provides two operations:
  - send(message) – message size fixed or variable
  - receive(message)
- If P and Q wish to communicate, they need to:
  - establish a communication link between them
  - exchange messages via send/receive
- Implementation of communication link
  - physical (e.g., shared memory, hardware bus)
  - logical (e.g., logical properties)

A communication link must exist between them. This link can be implemented in a variety of ways.Physical(shared memory, bus) logical(logical properties).

Naming:

- Processers that want to communicate must have a way to refer to each other.
- They can use either direct or indirect communication.

Direct Communication

- Processes must name each other explicitly:
  - send (P, message) – send a message to process P
  - receive(Q, message) – receive a message from process Q
- Properties of communication link
  - Links are established automatically.
  - A link is associated with exactly one pair of communicating processes.
  - Between each pair there exists exactly one link.
  - The link may be unidirectional, but is usually bi-directional.

Indirect Communication

- Messages are directed and received from mailboxes (also referred to as ports).
  - Each mailbox has a unique id.
  - Processes can communicate only if they share a mailbox.
- Properties of communication link
  - Link established only if processes share a common mailbox
  - A link may be associated with many processes.
  - Each pair of processes may share several communication links.
  - Link may be unidirectional or bi-directional.
- Operations
  - create a new mailbox
  - send and receive messages through mailbox
  - destroy a mailbox
- Primitives are defined as:

  send(A, message) – send a message to mailbox A

  receive(A, message) – receive a message from mailbox A
- Solutions
  - Allow a link to be associated with at most two processes.
  - Allow only one process at a time to execute a receive operation.
  - Allow the system to select arbitrarily the receiver. Sender is notified who the receiver was.

Synchronization

- Message passing may be either blocking or non-blocking.
- Blocking is considered synchronous
- Non-blocking is considered asynchronous
- Send and receive primitives may be either blocking or non-blocking.

Blocking send: sending is blocked until message is received.

Non –blocking send: sends message and resumes operation.

Blocking receive: receiver blocks until message is available.

Non blocking receive: receiver retrievals either valid message or null.

Buffering

- Queue of messages attached to the link; implemented in one of three ways.

1. Zero capacity – 0 messages

Sender must wait for receiver (rendezvous).

2. Bounded capacity – finite length of n messages

Sender must wait if link full.

3. Unbounded capacity – infinite length , rename

Sender never waits.

WRITE short notes on system programs.

System programs:

- Systems programs provide a convenient environment for program development and execution.
- They are used for user interface to system calls.
- Categories are as follows:

File management:

These programs create, delete, copy, rename, print, dump, list and generally manipulate files and directories.

- In addition operating system keeps information about all its processers and there are system calls to acccess this information.

S**tatus information:**

- Programs simply ack for date, time, amount of available memory, disk space, number of users or staus information.
- The information is printed to the terminal.

File modification :

- Text editors are available to create and modify the content of files stored on disk/type.

Programming and execution:

- **Once** program is assembled or compiled, it must be loaded into memory to be executed.
- Provides absolute loaders, relocatable loaders, linkage editors, overlay loaders.

Programming language support:

- Common programming languages: c,c++,java,VB
- Compilers, assemblers and interpreters are provided.

Communications:

- provide mechanism for virtual connections among processors, users and computer systems.
- Allow to send messages, because web pages ,send pages, send emails, login remotely or to transfer files.

Pondicherry University Questions

2 Marks

1. What is the advantage of Multiprogramming?(UQ APR'12) (Ref.Pg.No.04Qn.No.5)
2. List the 2 type of server systems?(UQ NOV '13) (Ref.Pg.No.11 Qn.No.39)
3. What is Distributed system? (UQ APR'12, NOV '15) (Ref.Pg.No.05 Qn.No.10)
4. What are the advantages of distributed processing?(UQ APR'13) (Ref.Pg.No.05 Qn.No.11)
5. What is a process state and mention the various states of a process?(UQ APR'12) (Ref.Pg.No.07 Qn.No.20)
6. What is real time system?(UQ APR'11,NOV '11) (Ref.Pg.No.09 Qn.No.28)
7. What do you mean by system calls?(UQ APR'12) (Ref.Pg.No.09 Qn.No.29)
8. What are the five major activities of an operating system with regard to process management?(UQ APR'14) (Ref.Pg.No.10 Qn.No.36)
9. Describe the differences between symmetric and asymmetric multiprocessing?(UQ APR'11) (Ref.Pg.No.10 Qn.No.37)
10. Define Information Sharing?(UQ NOV '12) (Ref.Pg.No.11 Qn.No.38)
11. Define PCB and write its contents?(APR'11) (Ref.Pg.No.11 Qn.No.42)
12. What is meant by independent process and cooperating process? (APR'12) (Ref.Pg.No.12 Qn.No.43)
13. What is meant by context switch? (NOV'14)) (Ref.Pg.No.8 Qn.No.23)
14. What is an Operating system? (APR'15) (Ref.Pg.No.4 Qn.No.1)
15. What are multiprocessor systems & give their advantages?(UQ APR'12,APR'14,APR'15) (Ref.Pg.No.05 Qn.No.8)
16. Name any three components of the system (UQ Nov '15) (Ref.Pg.No. 12 Qn.No.44)

11 MARKS

1. Explain computer systems which are categorized according to the number of processors used?(UQ APR'12) (Ref.Pg.No.13 Qn.No.1)
2. Write in detail about Mainframe systems? (UQ APR'14) (Ref.Pg.No.16Qn.No.2)
3. Explain Multiprocessor systems or Parallel Systems?(UQ APR'11) (Ref.Pg.No.19Qn.No.3)
4. Write about Distributed Systems ? (UQ APR'11 & APR2012) (Ref.Pg.No.20Qn.No.4)
5. Write short notes on Clustered Systems and real time &hand held systems?(UQ NOV'13) (Ref.Pg.No.21 Qn.No.5)
6. Explain how cooperating process can communication in direct and indirect communication?(UQ APR' 13) (Ref.Pg.No.39 Qn.No.14)
7. Write notes on Operating System Services?(UQ APR'12, NOV '15) (Ref.Pg.No.28 Qn.No.8)

8. Write about Inter process Communication?(UQ APR' 14, NOV '15) (Ref.Pg.No.41 Qn.No.16)

9. Write short notes on Process?(UQ APR'12NOV '13) (Ref.Pg.No.33 Qn.No.11)

10. Write about process scheduling (NOV'14) (Ref.Pg.No.35 Qn.No.12)

11. Write about operation on Processes?(UQ NOV '13,NOV'14, NOV '15) (Ref.Pg.No.37 Qn.No.13)

12. Discuss and detail about System Calls and system programs ?(UQ APR'12 ,NOV '12 , NOV'14) (Ref.Pg.No.29 Qn.No.9)

13. Explain Different type of system calls?(UQ NOV '12,APR'15) (Ref.Pg.No.31 Qn.No.10)

14. List out and discuss operating System Components?(UQ NOV'12,APR'15) (Ref.Pg.No.25 Qn.No.7)

UNIT II

Threads: Overview – Threading issues - CPU Scheduling – Basic Concepts – Scheduling Criteria – Scheduling Algorithms – Multiple-Processor Scheduling – Real Time Scheduling - The Critical- Section Problem – Synchronization Hardware – Semaphores – Classic problems of Synchronization – Critical regions – Monitors.

2 Marks

1. What is a thread? (APR'15, NOV '15)

A thread otherwise called a lightweight process (LWP) is a basic unit of CPU utilization, it comprises of a thread id, a program counter, a register set and a stack. It shares with other threads belonging to the same process its code section, data section, and operating system resources such as open files and signals.

2. What are the benefits of multithreaded programming? (NOV'14)

The benefits of multithreaded programming can be broken down into four major categories:

- Responsiveness
- Resource sharing
- Economy
- Utilization of multiprocessor architectures

3. Write the types of Thread?

- Kernel-supported threads (e.g. Mach and OS/2) - kernel of O/S sees threads and manages switching between threads i.e. in terms of analogy boss (OS) tells person (CPU) which thread in process to do next.
- User-level threads - supported above the kernel, via a set of library calls at the user level. Kernel only sees process as whole and is completely unaware of any threads i.e. in terms of analogy manual of procedures (user code) tells person (CPU) to stop current thread and start another (using library call to switch threads)

4. **Compare user threads and kernel threads?**

User threads

User threads are supported above the kernel and are implemented by a thread library at the user level. Thread creation & scheduling are done in the user space, without kernel intervention. Therefore they are fast to create and manage blocking system call will cause the entire process to block.

Kernel threads

Kernel threads are supported directly by the operating system .Thread creation; scheduling and management are done by the operating system. Therefore they are slower to create & manage compared to user threads. If the thread performs a blocking system call, the kernel can schedule another thread in the application for execution

5. Define thread cancellation & target thread.

The thread cancellation is the task of terminating a thread before it has completed. A thread that is to be cancelled is often referred to as the target thread.

For example, if multiple threads are concurrently searching through a database and one thread returns the result, the remaining threads might be cancelled.

6. What are the different ways in which a thread can be cancelled?

Cancellation of a target thread may occur in two different scenarios:

• Asynchronous cancellation: One thread immediately terminates the target thread is called asynchronous cancellation.

• Deferred cancellation: The target thread can periodically check if it should terminate, allowing the target thread an opportunity to terminate itself in an orderly fashion.

7. Define CPU scheduling (APR'15)

CPU scheduling is the process of switching the CPU among various processes. CPU scheduling is the basis of multiprogrammed operating systems. By switching the CPU among processes, the operating system can make the computer more productive.

8. What is preemptive and non preemptive scheduling?

Under nonpreemptive scheduling once the CPU has been allocated to a process, the process keeps the CPU until it releases the CPU either by terminating or switching to the waiting state. Preemptive scheduling can preempt a process which is utilizing the CPU in between its execution and give the CPU to another process.

9. What is a Dispatcher? (NOV '11)

The dispatcher is the module that gives control of the CPU to the process selected by the short-term scheduler. This function involves:

• Switching context

- Switching to user mode
- Jumping to the proper location in the user program to restart that program.

10. What is dispatch latency?

The time taken by the dispatcher to stop one process and start another running is known as dispatch latency.

11. What are the various scheduling criteria for CPU scheduling?

The various scheduling criteria are

- CPU utilization
- Throughput
- Turnaround time
- Waiting time
- Response time

12. Define throughput? (NOV '11)(ARPIL '14)

Throughput in CPU scheduling is the number of processes that are completed per unit time. For long processes, this rate may be one process per hour; for short transactions, throughput might be 10 processes per second.

13. What is turnaround time?

Turnaround time is the interval from the time of submission to the time of completion of a process. It is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O.

14. Define race condition? (ARPIL '14)

When several process access and manipulate same data concurrently, then the outcome of the execution depends on particular order in which the access takes place is called race condition. To avoid race condition, only one process at a time can manipulate the shared variable.

15. What is critical section problem? (APR '11)

Consider a system consists of 'n' processes. Each process has segment of code called a critical section, in which the process may be changing common variables, updating a table, writing a file. When

one process is executing in its critical section, no other process can allowed to execute in its critical section.

16. What are the requirements that a solution to the critical section problem must satisfy? (NOV'14)

The three requirements are

- Mutual exclusion
- Progress
- Bounded waiting

17. Define entry section and exit section.

The critical section problem is to design a protocol that the processes can use to cooperate. Each process must request permission to enter its critical section. The section of the code implementing this request is the entry section. The critical section is followed by an exit section. The remaining code is the remainder section

18.Give two hardware instructions and their definitions which can be used for implementing mutual exclusion.

- TestAndSet

```
boolean TestAndSet (boolean &target)
{
boolean rv = target;
target = true;
return rv;
}
```

- Swap

```
void Swap (boolean &a, boolean &b)
{
boolean temp = a;
a = b;
b = temp;
}
```

19.What is semaphores?

A semaphore 'S' is a synchronization tool which is an integer value that, apart from initialization, is accessed only through two standard atomic operations; wait and signal. Semaphores can be used to deal with the n-process critical section problem. It can be also used to solve various synchronization problems. The classic definition of 'wait'

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

The classic definition of 'signal'

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

20. Define busy waiting and spinlock?

When a process is in its critical section, any other process that tries to enter its critical section must loop continuously in the entry code. This is called as busy waiting and this type of semaphore is also called a spinlock, because the process while waiting for the lock.

21.What happens if the time allocated in a Round Robin Scheduling is very large? And what happens if the time allocated is very low?

It results in a FCFS scheduling. If time is too low, the processor through put is reduced. More time is spent on context switching

22.What is reader writers problem in shared file type of interprocess communication?

In this problem if reader is faster than writer or writer is faster than reader than problem is bound to creep in .if reader is faster than writer then it would try to read the memory location which is not written by writer process. if writer is faster than reader then it would just keep filling the buffers unboundedly. this problem is reader. Writer problem in shared file IPC. So there is need for synchronization between them.

23. Write down Scheduling Criteria?

- **CPU utilization** i.e. CPU usage - to maximize
- **Throughput** = number of processes that complete their execution per time unit - to maximize
- **Turnaround time** = amount of time to execute a particular process - to minimize
- **Waiting time** = amount of time a process has been waiting in the ready queue - to minimize
- **Response time** = amount of time it takes from when a job was submitted until it initiates its first response (output), **not** to time it completes output of its first response - to minimize

24. What is the difference between process and thread?

1. Threads are easier to create than processes since they don't require a separate address space.
2. Multithreading requires careful programming since threads share data structures that should only be modified by one thread at a time. Unlike threads, processes don't share the same address space.
3. Threads are considered lightweight because they use far less resources than processes.
4. Processes are independent of each other. Threads, since they share the same address space are interdependent, so caution must be taken so that different threads don't step on each other.
This is really another way of stating #2 above.
5. A process can consist of multiple threads.

25. What is Spooling?

Acronym for simultaneous peripheral operations on line. Spooling refers to putting jobs in a buffer, a special area in memory or on a disk where a device can access them when it is ready.
Spooling is useful because device access data that different rates. The buffer provides a waiting station where data can rest while the slower device catches up the spooling.

26. List the advantage of Spooling?

1. The spooling operation uses a disk as a very large buffer.
2. Spooling is however capable of overlapping I/O operation for one job with processor operations for another job.

27. What is meant by CPU–I/O Burst Cycle, CPU burst, I/O burst?

- **CPU–I/O Burst Cycle** – Process execution consists of a *cycle* of CPU execution and I/O wait.

- **CPU burst** is length of time process needs to use CPU before it next makes a system call (normally request for I/O).
- **I/O burst** is the length of time process spends waiting for I/O to complete.

28. Define Aging and starvation? (APR' 14)

Starvation: Starvation is a resource management problem where a process does not get the resources it needs for a long time because the resources are being allocated to other processes.

Aging: Aging is a technique to avoid starvation in a scheduling system. It works by adding an aging factor to the priority of each request. The aging factor must increase the requests priority as time passes and must ensure that a request will eventually be the highest priority request (after it has waited long enough).

29. What is context switch? (APR '11)

In a multitasking operating system stops running one process and starts running another. Many operating systems implement concurrency by maintaining separate environments or "contexts" for each process. The amount of separation between processes, and the amount of information in a context, depends on the operating system but generally the OS should prevent processes interfering with each other, e.g. by modifying each other's memory.

A context switch can be as simple as changing the value of the program counter and stack pointer or it might involve resetting the MMU to make a different set of memory pages available.

30. What is a monitor? (NOV '15)

A **monitor** is a synchronization construct that allows threads to have both mutual exclusion and the ability to wait (block) for a certain condition to become true. **Monitors** also have a mechanism for signaling other threads that their condition has been met.

11 Marks

CPU Scheduling:

Basic concepts:

DEFINITION:

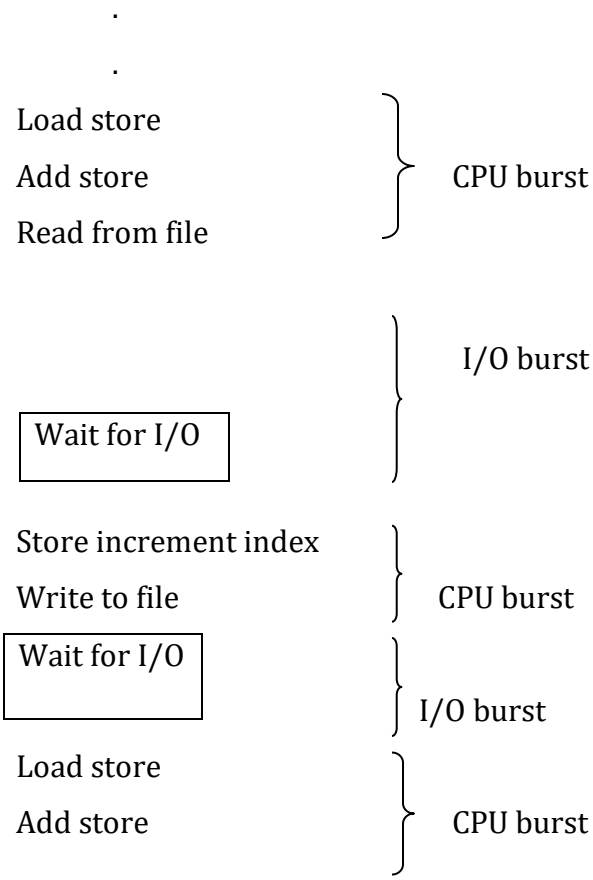
- CPU scheduling is the process of switching the CPU among various processes.
- CPU scheduling is the basic of multiprogrammed operating systems.
- By switching the CPU among processes, the operating system can make the computer more productive.

CPU – I/O BURST CYCLE:

The success of CPU scheduling depends on the property of processes:

- The process execution consists of a cycle of CPU execution and I/O wait.
- Processes alternate between these two states.
- Process execution begin with a CPU burst; followed by an I/O burst, then another CPU burst then another I/O burst and so on.
- Last CPU burst will end with a system request to terminate execution.
- An I/O bound program would have many, very short CPU bursts.
- A CPU bound program might have a few very long CPU bursts.

Fig: Alternating sequence of CPU and I/O bursts



.

1. Write short notes on CPU scheduler (8) (NOV 13) (APR '11) (NOV '15)

CPU Scheduler

- CPU scheduler selects one of the processes in the ready queue to be executed.
- There are two types of scheduling .they are
  1. Preemptive scheduling
  2. Non preemptive scheduling
- Preemptive scheduling-during process with the ,if is possible to remove the CPU from the process then it is called preemptive scheduling.
- Non preemptive scheduling-during processing with the CPU from the process then it is not possible to remove the CPU from the process then it is called non-preemptive scheduling.
- CPU scheduling decisions may take place when a process:

1. Switches from running to waiting state.
2. Switches from running to ready state.
3. Switches from waiting to ready.
4. Terminates.

- Scheduling under 1 and 4 is non-preemptive.
- All other scheduling is *preemptive.*

Dispatcher

- Dispatcher module gives control of the CPU to the process selected by the short-term scheduler; this involves:
  - switching context
  - switching to user mode
  - jumping to the proper location in the user program to restart that program
- *Dispatch latency* – time it takes for the dispatcher to stop one process and start another running.

Scheduling Criteria

- CPU utilization – keep the CPU as busy as possible
- Throughput – # of processes that complete their execution per time unit
- Turnaround time – amount of time to execute a particular process
- Waiting time – amount of time a process has been waiting in the ready queue

- Response time – amount of time it takes from when a request was submitted until the first response is produced, not output (for time-sharing environment)

Optimization Criteria

- Max CPU utilization
- Max throughput
- Min turnaround time
- Min waiting time
- Min response time

Scheduling algorithms:

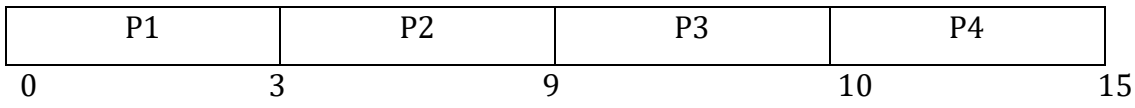
- First –come first served scheduling(FCFS)
- Shortest-job-first scheduling(SJF)
- Priority scheduling
- Round-robin scheduling(RR)
- Multilevel queue scheduling
- Multilevel feedback queue scheduling

2. Explain FCFS (6) (NOV 13)

- It is a non pre emptive algorithm
- The process which requests the CPU first us allocated to the CPU first.
- Demerit –A long CPU bound job may take the CPU and force short job to wait for a long tern called convoy effect
- Gantt chart -It represents the order in which the process is executed
- Example

Process	Burst time
P1	3
P2	6
P3	4
P4	2

Gantt chart:



Waiting time

Process	waiting time
P1	0
P2	3
P3	9
P4	13

Average waiting time=(0+3+9+13)/4 = 6.25 ms

Turn around time (TAT):

TAT=waiting time + burst time

Process	TAT
P1	3
P2	9
P3	13
P4	15

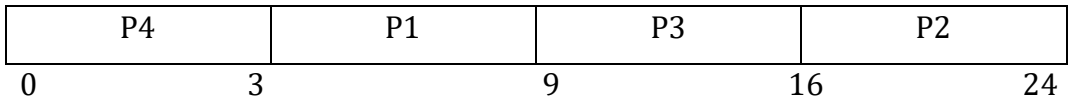
Average TAT=(3+9+13+15)/4=10 ms

3.Write short notes on shortest job scheduling (6)

- CPU is allocated to a job with smaller CPU burst that is the shortest CPU burst job will have the higher priority over other jobs
- Two schemes:
  - nonpreemptive – once CPU given to the process it cannot be preempted until completes its CPU burst.
  - preemptive – if a new process arrives with CPU burst length less than remaining time of current executing process, preempt. This scheme is known as the Shortest-Remaining-Time-First (SRTF).
- SJF is optimal – gives minimum average waiting time for a given set of processes.
- Example

Process	Burst time
P1	6
P2	8
P3	7
P4	3

Gantt chart:



Waiting time

Process	waiting time
P1	3
P2	16
P3	9
P4	0

Average waiting time=(3+16+9+0)/4 = 7 ms

Turn around time (TAT):

TAT=waiting time + burst time

Process	TAT
P1	9
P2	24
P3	16
P4	3

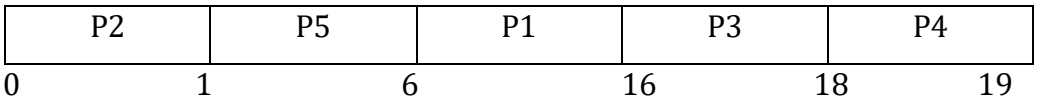
Average TAT=(9+24+16+3)/4=13 ms

4.Write short notes on priority scheduling (6)

- A priority number (integer) is associated with each process
- The CPU is allocated to the process with the highest priority (smallest integer ≡ highest priority).
  - Preemptive
  - nonpreemptive
- preemptive-preempt the CPU if the priority of the newly arrived process is higher than the priority of the currently running process
- non preemptive - allows the currently running process to complete its CPU burst.
- SJF is a priority scheduling where priority is the predicted next CPU burst time.
- Problem in priority scheduling is Starvation –i.e low priority processes may never execute.
- Solution -Aging – as time progresses increase the priority of the process.
- Example

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

Gantt chart:



Waiting time

Process	waiting time
P1	6
P2	0
P3	16
P4	18
P5	1

Average waiting time=(6+0+16+18+1)/5 = 8.2ms

Turn around time (TAT):

TAT=waiting time + burst time

Process	TAT
P1	16
P2	1
P3	18
P4	19
P5	6

Average TAT=(16+1+18+19+6)/5=12 ms

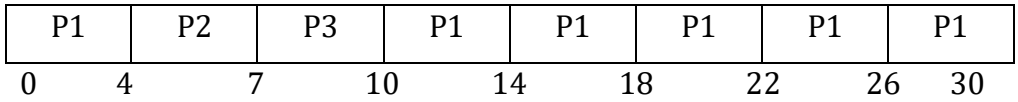
5. Explain Round Robin (RR) (NOV '15) (6)

- It is a pre-emptive scheduling
- Each process gets a small unit of CPU time (*time quantum*), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue.
- If there are n processes in the ready queue and the time quantum is q, then each process gets $1/n$ of the CPU time in chunks of at most q time units at once. No process waits more than $(n-1)q$ time units.
- Performance
  - q large $\Rightarrow$ FIFO
  - q small $\Rightarrow$ q must be large with respect to context switch, otherwise overhead is too high.

Example given time quantum=4ms

Process	Burst time
P1	24
P2	3
P3	3

Gantt chart:



Waiting time

Process	waiting time
P1	10-4=6
P2	4
P3	7

Average waiting time=(6+4+7)/3 = 5.6 ms

Turn around time (TAT):

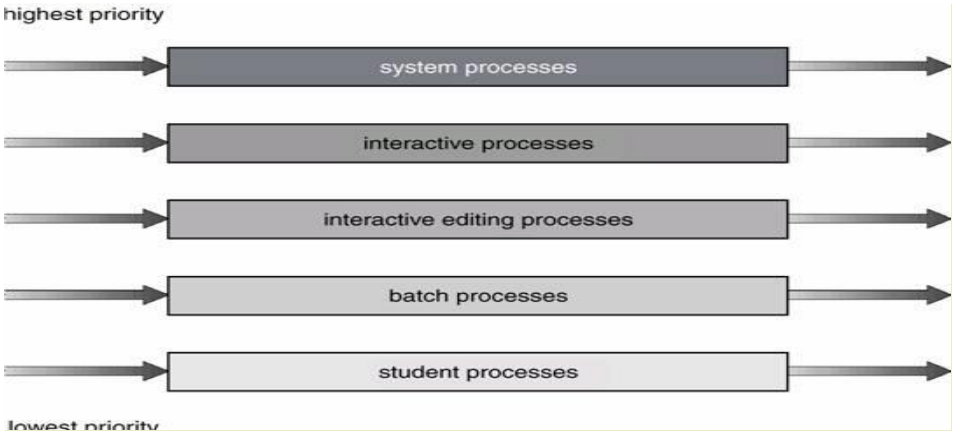
TAT=waiting time + burst time

Process	TAT
P1	30
P2	7
P3	10

Average TAT=(30+7+10)/3=15.6 ms

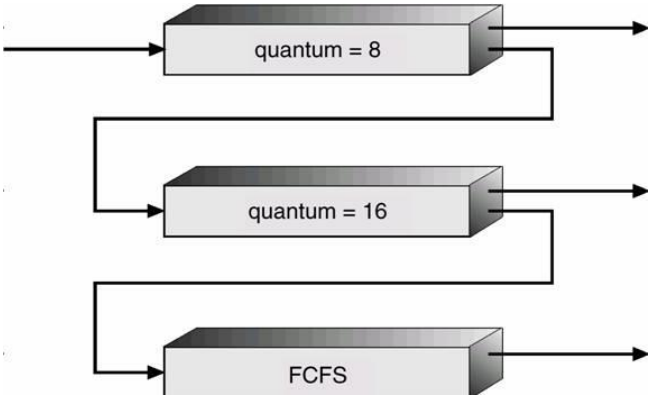
6.Explain Multi level queue scheduling (6)(NOV '11)

- Ready queue is partitioned into separate queues:
  foreground (interactive)
  background (batch)
- Each process is permanently assigned to one queue based on some property eg: process type
- Each queue has its own scheduling algorithm,
  foreground – RR
  background – FCFS
- Scheduling must be done between the queues.
  - Fixed priority scheduling; (i.e., serve all from foreground then from background). Possibility of starvation.
  - Time slice – each queue gets a certain amount of CPU time which it can schedule amongst its processes; i.e., 80% to foreground in RR
  - 20% to background in FCFS



7.Explain Multilevel Feedback Queue (7)

- A process can move between the various queues; aging can be implemented this way.
- Multilevel-feedback-queue scheduler defined by the following parameters:
  - number of queues
  - scheduling algorithms for each queue
  - method used to determine when to upgrade a process
  - method used to determine when to demote a process
  - method used to determine which queue a process will enter when that process needs service



Example of Multilevel Feedback Queue

- Three queues:
  - Q_0 – time quantum 8 milliseconds
  - Q_1 – time quantum 16 milliseconds
  - Q_2 – FCFS
- Scheduling
  - A new job enters queue Q_0 which is served FCFS. When it gains CPU, job receives 8 milliseconds. If it does not finish in 8 milliseconds, job is moved to queue Q_1.
  - At Q_1 job is again served FCFS and receives 16 additional milliseconds. If it still does not complete, it is preempted and moved to queue Q_2.

8. Write short notes on Multiple-Processor Scheduling (4)(NOV '11)(APR'12)

- Here each queue can have separate processor
- Suppose if any of the queue is empty then that processor connected to it become idle. To avoid this problem we have to use common ready queue
- All the processes are sent to common ready queue and the processor has to take the process from that queue therefore here the processor is self scheduling
- Here also problem arise if two processor selects same process to avoid this one of the processor has to act as master for other processors that is master slave relationship
- The master has to select job from common ready queue and allocate that process to any one of slave processor.

Real-Time Scheduling

- *Hard real-time* systems – required to complete a critical task within a guaranteed amount of time.
- *Soft real-time* computing – requires that critical processes receive priority over less fortunate ones.

9. Write in detail about Critical section problem OR what is critical section problem and explain two process solution and multiple process solutions? (11) (NOV 11) (APR'15)

The Critical-Section Problem

- *n* processes all competing to use some shared data
- Each process has a code segment, called *critical section*, in which the shared data is accessed.
- Problem – ensure that when one process is executing in its critical section, no other process is allowed to execute in its critical section.
  - The critical section problem is to design a protocol that the processes can use to co-operate.
  - Each process must request permission to enter its critical section.
  - The section of code implementing this request is the entry section,
  - The critical section is followed by an exit section.
  - The remaining code is the remainder section.

GENERAL STRUCTURE OF TYPICAL PROCESS P1,

Do

{

Entry section

Critical section

Exit section

Remainder section

} while (TRUE);

Solution to Critical-Section Problem

- Mutual Exclusion. If process P_i is executing in its critical section, then no other processes can be executing in their critical sections.
- Progress. If no process is executing in its critical section and there exist some processes that wish to enter their critical section, then the selection of the processes that will enter the critical section next cannot be postponed indefinitely.
- Bounded Waiting. A bound must exist on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted.Assume that each process executes at a nonzero speed .No assumption concerning relative speed of the *n* processes.

TWO PROCESS SOLUTION:

Algorithm 1

- Process p0 and p1 both are cooperating through a shared variable turn.
- Turn may be 0 or 1 , if it is 0 p0 will be executed that is p0 ‘s critical section executes if it is 1,p1 will be executes that is p1 ‘s critical section executes.
- Shared variables:
  - int turn;
    initially turn = 0
  - turn = i $\Rightarrow$ P_i can enter its critical section
- Process P_i

```
do {
        while (turn != i) ;
                critical section
        turn = j;
                remainder section
} while (1);
```

- Satisfies mutual exclusion, but not progress
  (because strict alternation is enforced)

Algorithm 2

- Shared variables flag[0] and flag[1] are used to synchronize two computing processes.
- flag[0] and flag[1] are initially set to false.
- Whenever a process p0 intends to enter its CS ,it indicates by setting flag[0] which is accessible by other coo.perating process p1.now p0 checks whether flag[1] is set,if not then p0 enters its CS.
- Shared variables
  - boolean flag[2];
    initially flag [0] = flag [1] = false.
  - flag [i] = true $\Rightarrow$ P_i ready to enter its critical section

Process P_i

```
do {
flag[i] := true;
        while (flag[j]) ;
```

```
                critical section
            flag [i] = false;
                remainder section
        } while (1);
```

- Satisfies mutual exclusion, but not progress requirement

  (flag[0]:=true; flag[1]:=true; ➔ both P_i looping in while())

Algorithm 3

- If flag[0] is set and turn equals to zero then p0 enters CS.
- If flag[1] ==1 and turn==1 thenp1 enters to CS.
- Combined shared variables of algorithms 1 and 2.
- Process P_i

```
        do {
            flag [i]:= true;
        turn = j;
        while (flag [j] and turn = j) ;
                critical section
            flag [i] = false;
                remainder section
        } while (1);
```

- Meets all three requirements (mutual exclusion, progress, bounded waiting); solves the critical-section problem for two processes.

10. Explain Bakery Algorithm OR Multiple process solution (5) (APR'15)

- Before entering its critical section, process receives a number. Holder of the smallest number enters the critical section.
  - Holder of the smallest number enters the critical section.
- If processes P_i and P_j receive the same number, if $i < j$, then P_i is served first; else P_j is served first.
- Shared data

  boolean choosing[n];

  int number[n];

  Data structures are initialized to false and 0 respectively

  Algorithm

```
do {
```

```
choosing[i] = true;
number[i] = max(number[0], number[1], …, number [n – 1])+1;
choosing[i] = false;
for (j = 0; j < n; j++) {
        while (choosing[j]) ;
        while ((number[j] != 0) && (number[j], j) < (number[i],i)) ;
}
    critical section
number[i] = 0;
    remainder section
} while (1);
```

11. Write short notes on Synchronization Hardware (5) (NOV 11)(NOV 13)

- Test and modify the content of a word atomically.

```
boolean TestAndSet(boolean &target) {
        boolean rv = target;
        target = true;

        return rv;
    }
```

Mutual Exclusion with Test-and-Set

- Shared data:

```
boolean lock = false;
```

- Process P_i

```
do {
while (TestAndSet(lock)) ;
critical section
lock = false;
remainder section
```

```
        }
while(1);
```

Synchronization Hardware (Swap)

- Atomically swap two variables.

```
void Swap(boolean &a, boolean &b) {
        boolean temp = a;
        a = b;
        b = temp;
}
```

Mutual Exclusion with Swap

- Shared data (initialized to false):

```
boolean lock;
```

- Local data:

```
boolean key;
```

- Process P_i

```
do {
        key = true;
        while (key == true)
                Swap(lock, key);
            critical section
        lock = false;
            remainder section
} while(1);
```

Bounded Mutual Exclusion with T&S

- Shared data (initialized to false):

```
boolean lock;
boolean waiting[n];
```

- Process P_i

```
do {
waiting[i] = true;
key = true;
```

```
        while (waiting[i] && key) key = TestAndSet(lock);
        waiting[i] = false;
                critical section
        j=(i+1) % n;
        while ((j!=i) && !waiting[j])
                j = (j+1) % n;
        if (j == i)         lock = false;
        else     waiting[j] = false;
                remainder section
} while(1);
```

12.Write short notes on Semaphores(NOV '12, NOV '15)

- Synchronization tool that does not require busy waiting.
- A Semaphore *S* is an integer variable can only be accessed via two indivisible (atomic) operations wait and signal.

```
wait (S):
        while S≤ 0 do no-op;
        S--;

signal (S):
        S++;
```

  - There are 2 processes p1 and p2 .p1 checks the valur of S,initially S=1 therefore s!=0. Since S- - makes s=0 so p1 enter to its CS .At the same time p2 also want to execute CS ,it checks S value which is already 0 so that it understands some process is in CS.so that p2 has to wait.
  - After p1 finishes,it increments S by 1 therefore now S=1.now p2 checks S value,which is not equal to 0 ,so that p2 will decrement S by 1 therefore S=0 no p2 enter to its CS.

Critical Section of *n* Processes

- N processes share a semaphore mutex initialized to 1.
- Shared data:

```
semaphore mutex;    // initially mutex = 1
```

- Process P_i:

```
do {
```

```
        wait(mutex);
           critical section
             signal(mutex);
      remainder section
} while (1);
```

Semaphore Implementation

- Define a semaphore as a record

```
typedef struct {
  int value;
  struct process *L;
} semaphore;
```

- Assume two simple operations:
  - block() suspends the process that invokes it.
  - wakeup(*P*) resumes the execution of a blocked process P.

Implementation

- Semaphore operations now defined as

```
wait(S):
S.value--;
if (S.value < 0)
{
add this process to S.L;
block();
}
signal(S):
S.value++;
if (S.value <= 0) {
remove a process P from S.L;
wakeup(P);
}
```

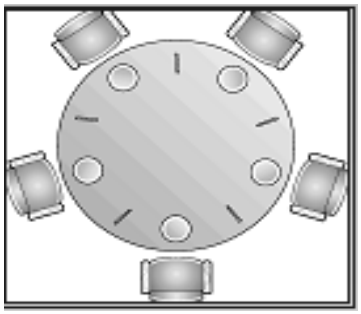
Semaphore as a General Synchronization Tool

- Execute *B* in P_j only after *A* has executed in P_i
- Use semaphore *flag* initialized to 0

- Code:

P_i	P_j
⋮	⋮
A	*wait*(*flag*)
signal(*flag*)	*B*

13.Write short notes on Dining-Philosophers Problem (6) (NOV'14)



- Five philosophers are seated in a circular table
- A philosopher needs two forks to eat.
- When a philosopher gets hungry, she tries to pick up the left and right chopsticks.
  - ✓ A philosopher may pick up only one chopstick at a time.
- When a philosopher finished eating she puts down both chopsticks.
- The life of a philosopher consists of alternate periods of eating and thinking.
- Philosopher *i*:

```
do {
        wait(chopstick[i])
        wait(chopstick[(i+1) % 5])
                …
                eat
                …
        signal(chopstick[i]);
        signal(chopstick[(i+1) % 5]);
                …
                think
                …                    } while (1);
```

- The solution guarantees no two neighbors are eating simultaneously but may create a deadlock.
- Possible remedies:

1. Allow at most 4 philosophers to be sitting at the table.

2. Allow a philosopher to pickup his chopsticks only if both chopsticks are available.
3. An odd philosopher picks up first her left chopstick and then her right chopstick, an even philosopher picks up first her right chopstick and then her left chopstick.

- A deadlock free solution may not be starvation free.

14.Write short notes on Readers-Writers Problem (6)(APR '14,NOV'14)

- A data object is shared among several processes.
- Readers-processes that only want to read the shared objects.
- Writers-processes that want to update that is read and write the shared data object.
- More than one readers are allowed to access the shared object simultaneously.
- Writers must have exclusive access to the shared object.
- No reader will be kept waiting unless a writer has already obtained permissions to use the shared object
  - ✓ Writers may starve.
- Shared data

semaphore mutex, wrt;

Initially

mutex = 1, wrt = 1, readcount = 0

Readers-Writers Problem Writer Process

```
wait(wrt);
    …
writing is performed
 …
signal(wrt);
```

Readers-Writers Problem Reader Process

```
wait(mutex);
readcount++;
if (readcount == 1)
    wait(wrt);
signal(mutex);
    …


reading is performed
```

…

wait(mutex);

readcount--;

if (readcount == 0)

signal(wrt);

signal(mutex):

- The first reader to get access to database. subsequent readers increment a counter read count. As reader leaves ,they decrement the counter and the last one does an signal to the blocked writer.

15.Explain producer-consumer problem (7)(APR '14)

- A common paradigm for cooperating processes, A producer process produces information that is consumed by a consumer process.
- Have a buffer of items filled by the producer and emptied by the consumer
  - unbounded-buffer no limit on the size of the buffer.
  - bounded-buffer assumes that there is a fixed buffer size.
- Buffer is a shared memory .Producer and consumer run concurrently and must be synchronized. In bounded buffer, the consumer must wait if the buffer is empty and the producer must wait if the buffer is full.

In Unbounded Buffer,the consumer may have to wait for new items,but the producer can always produce new items

Bounded-Buffer Problem

- Shared data

  semaphore full, empty, mutex;

  Initially:

  full = 0, empty = n, mutex = 1

  Bounded-Buffer Problem Producer Process

  do {

  …

  produce an item in nextp

  …

  wait(empty);

  wait(mutex);

```
                …
            add nextp to buffer
                …
            signal(mutex);
            signal(full);
        } while (1);
```

- Count =number of buffers
- The producer produces an item and access buffer by decrementing count by 1 and sets mutex=0 and then adds item to buffer. after that producer leaves CS by setting mutex=1 ,increments full buffers by 1.

Bounded-Buffer Problem Consumer Process

```
        do {
            wait(full)
            wait(mutex);
                …
            remove an item from buffer to nextc
                …
            signal(mutex);
            signal(empty);
                …
            consume the item in nextc
                …
        } while (1);
```

- The consumer decrementing full buffer by 1 and then enters CS by setting mutex=0 after that consumes an item ,while leaving CS sets mutex=1 and increments empty buffers by 1.

16.Write short notes on Critical region (8)

- High-level synchronization construct
- A shared variable *v* of type *T*, is declared as:

```
v: shared T
```

- Variable *v* accessed only inside statement

region v when B do S

where *B* is a boolean expression.

While statement *S* is being executed, no other process can access variable *v*.

- Regions referring to the same shared variable exclude each other in time.
- When a process tries to execute the region statement, the Boolean expression *B* is evaluated. If *B* is true, statement *S* is executed. If it is false, the process is delayed until *B* becomes true and no other process is in the region associated with *v*.

Example – Bounded Buffer

- Shared data:

```
struct buffer {
        int pool[n];
        int count, in, out;
}
```

Bounded Buffer Producer Process

- Producer process inserts nextp into the shared buffer

```
region buffer when( count < n) {
pool[in] = nextp;
in:= (in+1) % n;
count++;
}
```

Bounded Buffer Consumer Process

- Consumer process removes an item from the shared buffer and puts it in nextc

```
region buffer when (count > 0) {
        nextc = pool[out];
                out = (out+1) % n;
                count--;
        }
```

Implementation region *x* when *B* do *S*

- Associate with the shared variable *x*, the following variables:

semaphore mutex, first-delay, second-delay;
int first-count, second-count;

- Mutually exclusive access to the critical section is provided by mutex.

- If a process cannot enter the critical section because the Boolean expression B is false, it initially waits on the first-delay semaphore; moved to the second-delay semaphore before it is allowed to reevaluate *B*.
- Keep track of the number of processes waiting on first-delay and second-delay, with first-count and second-count respectively.

- The algorithm assumes a FIFO ordering in the queuing of processes for a semaphore.

- For an arbitrary queuing discipline, a more complicated implementation is required.

17.Write short notes on Monitors (11)

- High-level synchronization construct that allows the safe sharing of an abstract data type among concurrent processes.

```
monitor monitor-name
        {
shared variable declarations
        procedure body P1 (...) {
                . . .
        }
procedure body P2 (...) {
                . . .
        }
                procedure body Pn (...)
        {               . . .
        }
        {
                initialization code
        }
```

}

- To allow a process to wait within the monitor, a condition variable must be declared, as

  condition x, y;

- Condition variable can only be used with the operations wait and signal.
  - The operation

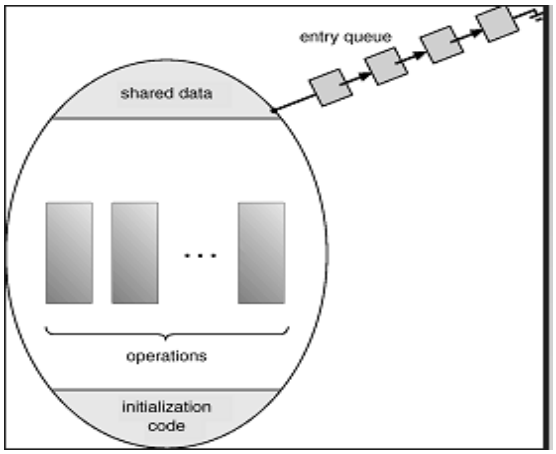
    x.wait();

means that the process invoking this operation is suspended until another process invokes

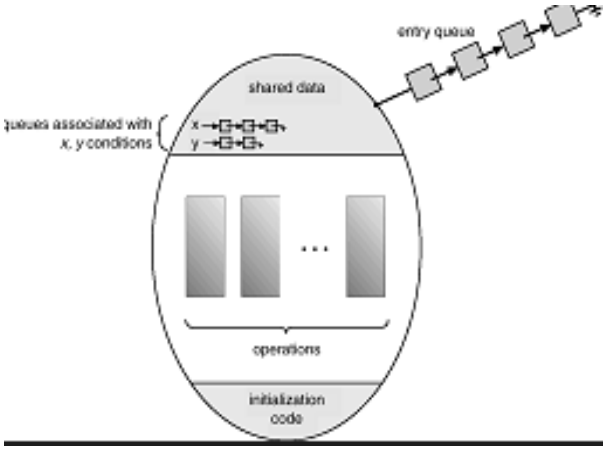
    x.signal();

  - The x.signal operation resumes exactly one suspended process. If no process is suspended, then the signal operation has no effect.

Schematic view of a monitor:



Monitor with condition variables



Dining Philosophers Example

```
monitor dp
enum {thinking, hungry, eating} state[5];
condition self[5];
void pickup(int i)              // following slides
```

```
void putdown(int i)  // following slides
void test(int i)                    // following slides
void init() {
        for (int i = 0; i < 5; i++)
state[i] = thinking;
        }
        }
        void pickup(int i) {
        state[i] = hungry;
                test[i];
                if (state[i] != eating)
                        self[i].wait();
        }
        void putdown(int i) {
                state[i] = thinking;
                // test left and right neighbors
                test((i+4) % 5);
                test((i+1) % 5);
        }
        void test(int i) {
                if ( (state[(i + 4) % 5] != eating) &&
                  (state[i] == hungry) &&
                  (state[(i + 1) % 5] != eating)) {
                        state[i] = eating;
                        self[i].signal();
                }
        }
```

Monitor Implementation Using Semaphores

- Variables

{

```
semaphore mutex;  // (initially  = 1)
semaphore next;     // (initially  = 0)
int next-count = 0;
```

- Each external procedure *F* will be replaced by

```
wait(mutex);
   …
  body of F;
   …
if (next-count > 0)
        signal(next);
else
        signal(mutex);
```

- Mutual exclusion within a monitor is ensured.

Monitor Implementation

- For each condition variable *x*, we have:

```
semaphore x-sem; // (initially  = 0)
int x-count = 0;
```

- The operation x.wait can be implemented as:

```
x-count++;
if (next-count > 0)
        signal(next);
else
        signal(mutex);
wait(x-sem);
x-count--;
```

- The operation x.signal can be implemented as:

```
if (x-count > 0) {
        next-count++;
        signal(x-sem);
        wait(next);
        next-count--;
}
```

- *Conditional-wait* construct: x.wait(c);
  - c – integer expression evaluated when the wait operation is executed.
  - value of c (a *priority number*) stored with the name of the process that is suspended.
  - when x.signal is executed, process with smallest associated priority number is resumed next.
- Check two conditions to establish correctness of system:
  - User processes must always make their calls on the monitor in a correct sequence.
  - Must ensure that an uncooperative process does not ignore the mutual-exclusion gateway provided by the monitor, and try to access the shared resource directly, without using the access protocols.

18. Explain real time scheduling?

Real-Time Scheduling

Real-time computing is divided into two types:

- HARD REAL-TIME SYSTEMS
- SOFT REAL-TIME COMPUTING

HARD REAL-TIME SYSTEMS:

- **Hard real-time** systems are required to complete a critical task within a guaranteed amount of time.
- Generally, a process is submitted along with a statement of the amount of time in which it needs to complete or perform I/O.
- The scheduler then either admits the process, guaranteeing that the process will complete on time, or rejects the request as impossible.
- This is known as **resource reservation.**
- Such a guarantee requires that the scheduler know exactly how long each type of operating-system function takes to perform, and therefore each operation must be guaranteed to take a maximum amount of time.

- Such a guarantee is impossible in a system with secondary storage or virtual memory, as we shall show in the next few chapters, because these subsystems cause unavoidable and unforeseeable variation in the amount of time to execute a particular process.
- Therefore, hard real-time systems are composed of special-purpose software running on hardware dedicated to their critical process, and lack the full functionality of modern computers and operating systems.

SOFT REAL TIME COMPUTING:

- **Soft real-time** computing is less restrictive. It requires that critical processes receive priority over less fortunate ones.
- Although adding soft real-time functionality to a time-sharing system may cause an unfair allocation of resources and may result in longer delays, or even starvation, for some processes, it is at least possible to achieve.
- The result is a general-purpose system that can also support multimedia, high-speed interactive graphics, and a variety of tasks that would not function acceptably in an environment that does not support soft real-time computing. Implementing soft real-time functionality requires careful design of the scheduler and related aspects of the operating system.
- First, the system must have priority scheduling, and real-time processes must have the highest priority.
- The priority of real-time processes must not degrade over time, even though the priority of non-real-time processes may. Second, the dispatch latency must be small. The smaller the latency, the faster a real-time process can start executing once it is runnable. The high-priority process would be waiting for a lower-priority one to finish. This situation is known as **priority inversion.**
- In fact, a chain of processes could all be accessing resources that the high-priority process needs. This problem can be solved via the **priority-inheritance protocol,** in which all these processes (the ones accessing resources that the high-priority process needs) inherit the high priority until they are done with the resource in question. When they are finished, their priority reverts to its original value.

The **conflict phase** of dispatch latency has two components:

1. Preemption of any process running in the kernel
2. Release by low-priority processes resources needed by the high-priority process

As an example, in Solaris 2, the dispatch latency with preemption disabled is over 100 milliseconds. However, the dispatch latency with preemption enabled is usually reduced to 2 milliseconds.

THREADS: OVERVIEW

DEFINITION

- A thread called as a light weight process (LWP) is a basic unit of CPU utilization.
- It comprises a thread ID, a program counter, a register set and a stack.
- It shares with other threads belonging to the same process its code section, data section, and other operating system resources such as open files and signals.

Benefits of multi threaded programming

1. Responsiveness
2. Resource sharing
3. Economy
4. Utilization of multiprocessor architecture.

User and Kernel Threads

Threads maybe provided at either the user level, for user threads or by the kernel for kernel threads.

USER THREADS:

- Supported above the kernel and are implemented by a thread library at the user level.
- Library provides support for thread execution, scheduling and management with no support from kernel.
- User threads are fast to create and manage.
- Blocking system call will cause the entire process to block.

KERNEL THREADS:

- Supported directly by the operating system.
- Kernel performs thread creation, scheduling and management in kernel space.
- They are slower to create and manage than the user thread.
- If thread performs blocking system call, the kernel can schedule another thread in the application for execution.

19. Explain in detail about the threading issues (APR'15)

Threading Issues:

FORK AND EXEC SYSTEM CALLS:

Fork system call – create a separate, duplicate process.

Exec system call – runs an executable file.

In multithreaded program ; semantics of fork and exec change

Fork

- duplicate all threads

- duplicates only the thread that invoked fork ()

Exec

- Program specified in the parameter to exec will replace the entire process.

Thread cancellation

It is a task of terminating a thread before it has completed.

Example:

When a user presses a button on a web browser that stops a web page from loading any further. Often a web is loaded in a separate thread when a user pressed the stop button, the thread loading the page is cancelled.

Target thread:

A thread that is to be cancelled is often referred to as the target thread.

Cancellations of a target thread may occur in two situations:

Two general approaches:

1. **Asynchronous cancellation** terminates the target thread immediately
2. **Deferred cancellation** allows the target thread to periodically check if it should be cancelled
   - ✓ Allow cancellation at safe points
   - ✓ Pthreads refer to safe point as cancellation points

Thread pools

Motivating example:

A web server creates anew thread to service each request.

Two concerns:

1. The amount of time required to create the thread prior to servicing the request ,compounded with the fact that this thread will be discarded once it has completed its work that is overhead to create thread

2. No limit on the number of thread created, may exhaust system resources, such as CPU time or memory .

To overcome the above said problem we need thread pools.

General idea:

- Create a pool of threads at process startup.
- If request comes in, then wakeup a thread from pool, assign the request to it,if no thread available ,server waits until one is free
- After completing the service, thread returns to pool.

Advantages:

- ✓ Usually slightly faster to service a request with an existing thread than create a new thread
- ✓ Allows the number of threads in the application(s) to be bound to the size of the pool

20. Describe the difference between wait (A) where A is a semaphore and B Wait() ,where B is a condition variable in a monitor. (NOV'14)

Monitor

- A monitor is a **set of multiple** routines which are protected by a mutual exclusion lock.
- None of the routines in the monitor can be executed by a thread until that thread acquires the lock.
- This means that only ONE thread can execute within the monitor at a time.
- Any other threads must wait for the thread that's currently executing to give up control of the lock.

However, a thread can actually suspend itself inside a monitor and then wait for an event to occur. If this happens, then another thread is given the opportunity to enter the monitor. The thread that was suspended will eventually be notified that the event it was waiting for has now occurred, which means it can wake up and reacquire the lock.

Semaphore

A semaphore is a simpler construct than a monitor because it's just a lock that protects a shared resource – and not a set of routines like a monitor. The application must acquire the lock before using that shared resource protected by a semaphore.

Example of a Semaphore – a Mutex

A mutex is the most basic type of semaphore, and mutex is short for mutual exclusion. In a mutex, only one thread can use the shared resource at a time. If another thread wants to use the shared resource, it must wait for the owning thread to release the lock.

Differences between Monitors and Semaphores

Both Monitors and Semaphores are used for the same purpose – thread synchronization. But, monitors are simpler to use than semaphores because they handle all of the details of lock acquisition and release. An application using semaphores has to release any locks a thread has acquired when the application terminates – this must be done by the application itself. If the application does not do this, then any other thread that needs the shared resource will not be able to proceed.

Another difference when using semaphores is that every routine accessing a shared resource has to explicitly acquire a a lock before using the resource. This can be easily forgotten when coding the routines dealing with multithreading . Monitors, unlike semaphores, automatically acquire the necessary locks.

Condition variable in Monitor:

A **condition variable** is basically a container of threads that are waiting on a certain **condition**. **Monitors** provide a mechanism for threads to temporarily give up exclusive access in order to wait for some **condition** to be met, before regaining exclusive access and resuming their task.

Pondicherry University Questions

2 Marks

1. What is a Dispatcher? (UQ NOV '11) (Ref.Pg.No.2 Qn.No.9)
2. Define throughput? (UQ NOV '11 &APR'14) (Ref.Pg.No.3 Qn.No.12)
3. Define race condition? (UQ APR'14) (Ref.Pg.No.3 Qn.No.14)
4. What is critical section problem? (UQ APR'11) (Ref.Pg.No.3 Qn.No.15)
5. Define Aging and starvation? (APR' 14) (Ref.Pg.No.7 Qn.No.28)
6. What is context switch?(APR '11) (Ref.Pg.No.7 Qn.No.29)
7. What are the benefits of multithreaded programming? (NOV'14)) (Ref.Pg.No1 Qn.No.2)
8. What are the requirements that a solution to the critical section problem must satisfy? (NOV'14)) (Ref.Pg.No.4 Qn.No.16)
9. What is a thread? (APR'15, NOV '15) (Ref.Pg.No.1 Qn.No.1)
10. Define CPU scheduling (APR'15)) (Ref.Pg.No.2 Qn.No.7)
11. What is a monitor? (NOV '15) (Ref.Pg.No.7 Qn.No.30)

11 MARKS

1. Write short notes on CPU scheduler? (UQ NOV '13, APR'11, NOV '15) (Ref.Pg.No.9 Qn.No.1)
2. Explain FCFS? (UQ NOV '13) (Ref.Pg.No.10 Qn.No.2)
3. Explain Multi level queue scheduling?(UQ NOV '11) (Ref.Pg.No.15 Qn.No.6)
5. Write in detail about Critical section problem? (UQ NOV '11) (Ref.Pg.No.17 Qn.No.9)
6. Write short notes on Multiple-Processor Scheduling and real time scheduling?(UQ NOV '11 & APR'12) (Ref.Pg.No.16 Qn.No.8)
7. Write short notes on Synchronization Hardware? (UQ NOV '11 &NOV '13) (Ref.Pg.No.20 Qn.No.11)
8. Write short notes on Semaphores?(UQ NOV '12 , NOV '13, NOV '15) (Ref.Pg.No.22 Qn.No.12)
9. What is critical section problem and explain two process solution and multiple process solutions? (APR'15) (Ref.Pg.No.15 Qn.No.9)
10. Explain producer-consumer problem? (UQ APR'14) (Ref.Pg.No.26 Qn.No.15)
11. Explain in detail about the threading issues (APR'15)) (Ref.Pg.No.35 Qn.No.19)
12. Write short notes on Dining-Philosophers Problem (NOV'14) (Ref.Pg.No.24 Qn.No.13)
13. Write short notes on Readers-Writers Problem? (UQ APR'14,NOV'14) (Ref.Pg.No.25 Qn.No.14)
14. Explain Bakery Algorithm OR Multiple process solution (APR'15)) (Ref.Pg.No.19 Qn.No.10)
15. Describe the difference between wait(A) where A is a semaphore and B Wait() ,where B is a condition variable in a monitor. (APR'15) (Ref.Pg.No.37 Qn.No.21)
16. Explain briefly about round robin scheduling with diagram. (NOV '15) (Ref.Pg.No.14 Qn.No.5)

UNIT III

System Model: Deadlock Characterization – Methods for handling Deadlocks -Deadlock Prevention – Deadlock avoidance – Deadlock detection – Recovery from Deadlocks - Storage Management – Swapping – Contiguous Memory allocation – Paging – Segmentation – Segmentation with Paging -

2 Marks

1. Define deadlock. (NOV 13, NOV '15)

A process requests resources; if the resources are not available at that time, the process enters a wait state. Waiting processes may never again change state, because the resources they have requested are held by other waiting processes. This situation is called a deadlock.

2. What is the sequence in which resources may be utilized?

Under normal mode of operation, a process may utilize a resource in the following sequence:

- Request: If the request cannot be granted immediately, then the requesting process must wait until it can acquire the resource.
- Use: The process can operate on the resource.
- Release: The process releases the resource.

3. What are conditions under which a deadlock situation may arise? Or Write four general strategies for dealing with deadlocks? (APR'15)(NOV'14)

A deadlock situation can arise if the following four conditions hold simultaneously in a system:

- Mutual exclusion
- Hold and wait
- No pre-emption
- Circular-wait

4. What is a resource-allocation graph?

Deadlocks can be described more precisely in terms of a directed graph called a system resource allocation graph. This graph consists of a set of vertices V and a set of edges E. The set of vertices V is partitioned into two different types of nodes; P the set consisting of all active processes in the system and R the set consisting of all resource types in the system.

5. Define request edge and assignment edge.

A directed edge from process Pi to resource type Rj is denoted by Pi, Rj; it signifies that process Pi requested an instance of resource type Rj and is currently waiting for that resource. A directed edge from resource type Rj to process Pi is denoted by Rj, Pi, it signifies that an instance of resource type has been allocated to a process Pi. A directed edge Pi ,Rj is called a request edge. A directed edge Rj,Pi is called an assignment edge.

6. What are the methods for handling deadlocks?

The deadlock problem can be dealt with in one of the three ways:

- Use a protocol to prevent or avoid deadlocks, ensuring that the system will never enter a deadlock state.
- Allow the system to enter the deadlock state, detect it and then recover.
- Ignore the problem all together, and pretend that deadlocks never occur in the system.

7. Define deadlock prevention.

Deadlock prevention is a set of methods for ensuring that at least one of the four necessary conditions like mutual exclusion, hold and wait, no preemption and circular wait cannot hold. By ensuring that that at least one of these conditions cannot hold, the occurrence of a deadlock can be prevented.

8. Define deadlock avoidance.

An alternative method for avoiding deadlocks is to require additional information about how resources are to be requested. Each request requires the system consider the resources currently available, the resources currently allocated to each process, and the future requests and releases of each process, to decide whether the could be satisfied or must wait to avoid a possible future deadlock.

9. What are a safe state and an unsafe state?

A state is safe if the system can allocate resources to each process in some order and still void a deadlock. A system is in safe state only if there exists a safe sequence. A sequence of processes <P1,P2,....Pn> is a safe sequence for the current allocation state if, for each Pi, the resource that Pi can still request can be satisfied by the current available resource plus the resource held by all the Pj, with j<i. if no such sequence exists, then the system state is said to be unsafe.

10. What is banker's algorithm?

Banker's algorithm is a deadlock avoidance algorithm that is applicable to a resource allocation system with multiple instances of each resource type. The two algorithms used for its implementation are:

- Safety algorithm: The algorithm for finding out whether or not a system is in a safe state.
- Resource-request algorithm: if the resulting resource allocations safe, the transaction is completed and process Pi is allocated its resources. If the new state is unsafe Pi must wait and the old resource-allocation state is restored.

11. Define logical address and physical address.

An address generated by the CPU is referred as logical address. An address seen by the memory unit that is the one loaded into the memory address register of the memory is commonly referred to as physical address.

12. What is logical address space and physical address space?

The set of all logical addresses generated by a program is called a logical address space; the set of all physical addresses corresponding to these logical addresses is a physical address space.

13. Why the page sizes is always power of 2 in memory? (NOV 13)

In operating systems that paging is implemented by breaking up an address into a page and offset number. It is most efficient to break the address into X page bits and Y offset bits, rather than perform arithmetic on the address to calculate the page number and offset. Because each bit position represents a power of 2, splitting an address between bits results in a page size that is a power of 2.

14.What is the main function of the memory-management unit?

The runtime mapping from virtual to physical addresses is done by a hardware device called a memory management unit (MMU).

15. Define dynamic loading.

To obtain better memory-space utilization dynamic loading is used. With dynamic loading, a routine is not loaded until it is called. All routines are kept on disk in a reloadable load format. The main program is loaded into memory and executed. If the routine needs another routine, the calling routine checks whether the routine has been loaded. If not, the reloadable linking loader is called to load the desired program into memory.

16. Define Swapping (APR'15)

A process needs to be in memory to be executed. However a process can be swapped temporarily out of memory to a backing tore and then brought back into memory for continued execution. This process is called swapping.

17. Define dynamic linking.

Dynamic linking is similar to dynamic loading, rather that loading being postponed until execution time, linking is postponed. This feature is usually used with system libraries, such as language subroutine libraries. A stub is included in the image for each library routine reference. The stub is a small piece of code that indicates how to locate the appropriate memory-resident library routine, or how to load the library if the routine is not already present.

18. What is of resource allocation graph? (APR'12)

Deadlocks can be described more precisely in terms of a directed graph called a system resource-allocation graph.

19. What is External Fragmentation? (APR 2014)

External Fragmentation happens when a dynamic memory allocation algorithm allocates some memory and a small piece is left over that cannot be effectively used. If too much external fragmentation occurs, the amount of usable memory is drastically reduced. Total memory space exists to satisfy a request, but it is not contiguous.

20. What are the common strategies to select a free hole from a set of available holes?

The most common strategies are

a. First fit

b. Best fit

c. Worst fit

21. What do you mean by best fit?(APR 2012)

Best fit allocates the smallest hole that is big enough the entire list has to be searched; unless it is sorted by size this strategy produces the smallest leftover hole.

22. What do you mean by first fit?(APR 2012)

First fit allocates the first hole that is big enough. Searching can either start at the beginning of the set of holes or where the previous first-fit search ended. Searching can be stopped as soon as a free hole that is big enough is found.

23. What are the criteria for LRU page replacement algorithms?(APR 2012)

- LRU stands for least recently used
- LRU replacement associates with each page the time of that page's last use.
- When a page must be replaced LRU chooses a page that has not been used for the longest period of time.
- This strategy is apt for looking backward in time.

24. Define effective access time.

Let p be the probability of a page fault (0£p£1). The value of p is expected to be close to 0; that is, there will be only a few page faults. The effective access time is Effective access time = (1-p) * ma + p * page fault time.ma : memory-access time

25. Write the difference between internal and external fragmentation.

Internal Fragmentation is the area in a region or a page that is not used by the job occupying that region or page. This space is unavailable for use by the system until that job is finished and the page or region is released.

26. Explain the difference between paging and segmentation.

	Paging	Segmentation
Length	Fixed length	Variable length
Address space	One dimensional address	Two dimensional address
Protection	Not easy	Good
Sharing	Not easy	Good
Fragment	Internal fragment	External fragment
Linking	Static linking	Dynamic linking
Loading	Dynamic loading	Dynamic loading

27. What are overlays?

To enable a process to be larger than the amount of memory allocated to it, overlays are used. The idea of overlays is to keep in memory only those instructions and data that are needed at a given time. When other instructions are needed, they are loaded into space occupied previously by instructions that are no longer needed.

28. Define paging? (NOV '15)

It is a memory management scheme .This is another possible solution to the external fragmentation. This allow the physical address space of a process to be non-contiguous thus allowing a process to be allocated in physical memory where ever the free space is available.

11 Marks

1. Write short notes on Dead lock and its characteristics?(6 Marks)

A process request for resource if it is not available at the time, a process enters wait state. Waiting process may never again change the state because the resource they requested are held by other process, this situation is called deadlock.

Example:

Suppose a computer has one tape drive and one plotter, the process A and B request tape drive and plotter respectively .Both request are granted.

Now A request the plotter and B request tape drive. Without giving up of its resources, then both the request cannot be granted. This situation is called deadlock.

Example

Semaphores A and B, initialized to 1

P0	P1
wait (A);	wait(B);
wait (B);	wait(A);

SYSTEM MODEL:

Under the normal mode of operation a process may utilize a resource in only the following sequence:

1. Request:

   If the request cannot be granted immediately(eg.the resource is being used by another process), then the requesting process must wait until it can acquire the resource.

2. Use:

   The process can operate on the resource (for eg.if the resource is a printer, the process can print)

3. Release:

   The process releases the resource.

Deadlock Characterization

Necessary condition:

A deadlock situation can arise if the following four conditions hold simultaneously in a system:

- Mutual exclusion: At least one resource must be held in a non-sharable mode; that is,only one process at a time can use a resource. If another process requests that resource, the requesting process must be delayed until the resource has been released.

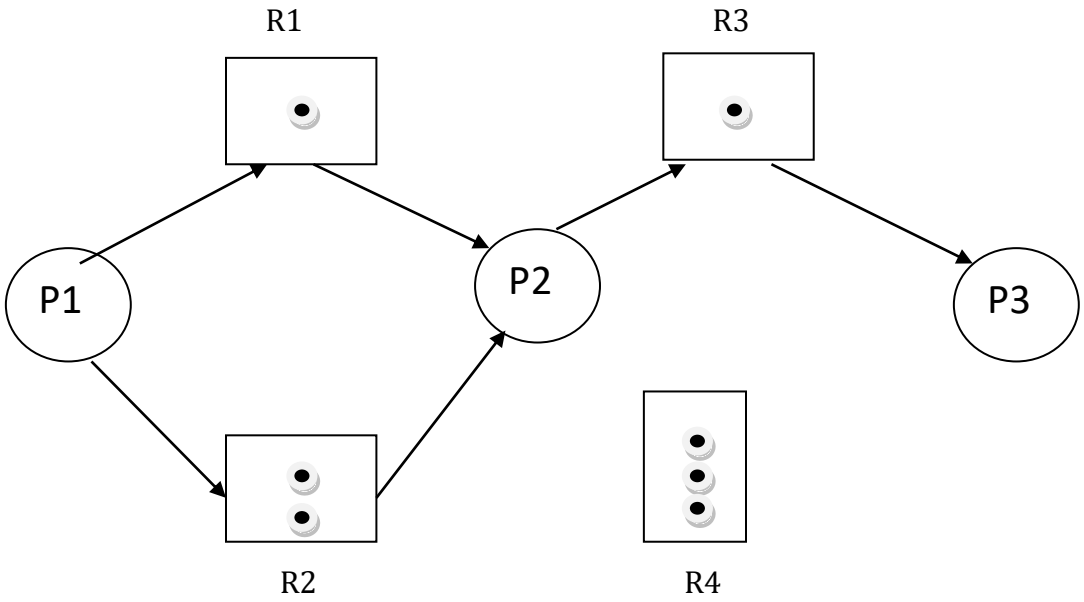
- Hold and wait: a process must be holding at least one resource is waiting to acquire additional resources held by other processes.
- No preemption: a resource can be released only voluntarily by the process holding it, after that process has completed its task.
- Circular wait: there exists a set $\{P_0, P_1, \ldots, P_0\}$ of waiting processes such that P_0 is waiting for a resource that is held by P_1, P_1 is waiting for a resource that is held by P_2, …, Pn-1 is waiting for a resource that is held by Pn, and P_0 is waiting for a resource that is held by P_0.
- Resource allocation graph:deadlocks can be described more precisely in terms of directed graph called a system resource allocation graph.

This graph consists of a set of vertices V and a set of edges E.

-the set of vertices V is partitioned into two different types;

P-set containing all active processes.

R-set consisting of all resource types.



Request edge: Directed edge p_i->r_j is called a request edge.

Assignment edge:A directed edge r_j->p_i is called an assignment edge.

2. Explain Deadlock Prevention in detail? (6 Marks) (APR'15)

Deadlock prevention:

-Deadlock occurs if each of the four necessary conditions hold.

-By ensuring that at least one of these conditions cannot hold; the occurrence of deadlock can be prevented.

- **Mutual Exclusion** – not required for sharable resources; must hold for non sharable resources.

- **Hold and Wait** – must guarantee that whenever a process requests a resource, it does not hold any other resources.

Require process to request and be allocated all its resources before it begins execution, or allow process to request resources only when the process has none. Low resource utilization; starvation possible.

- **No Preemption** – If a process that is holding some resources requests another resource that cannot be immediately allocated to it, then all resources currently being held are released.

Preempted resources are added to the list of resources for which the process is waiting.

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

- **Circular Wait** – impose a total ordering of all resource types, and require that each process requests resources in an increasing order of enumeration.

3. Explain Deadlock Avoidance in details? (11 Marks) (APR 2012)

Simplest and most useful model requires that each process declare the maximum number of resources of each type that it may need.

The deadlock-avoidance algorithm dynamically examines the resource-allocation state to ensure that there can never be a circular-wait condition.

Resource-allocation state is defined by the number of available and allocated resources, and the maximum demands of the processes.

Safe State:

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

System is in safe state if there exists a safe sequence of all processes.

Sequence <P_1, P_2…, Pn> is safe if for each P_i , the resources that P_i can still request can be satisfied by currently available resources + resources held by all the P_j , with j < i.

If Pi resource needs are not immediately available, then P_i can wait until all P_j have finished. When P_j is finished, P_i can obtain needed resources, execute, return allocated resources, and terminate.

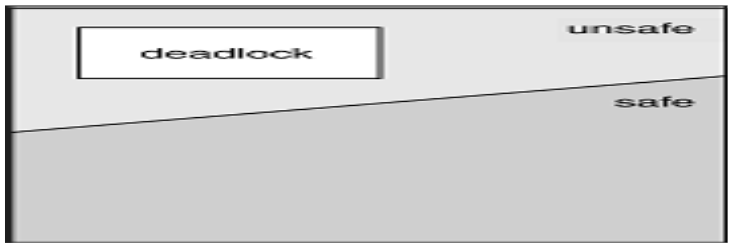
When P_i terminates, P_{i+1} can obtain its needed resources, and so on.

Basic Facts:

If a system is in safe state then no deadlocks. If a system is in unsafe state then possibility of deadlock to happen.

Avoidance an ensure that a system will never enter an unsafe state.

Safe, unsafe, dead lock state



Resource-Allocation Graph Algorithm

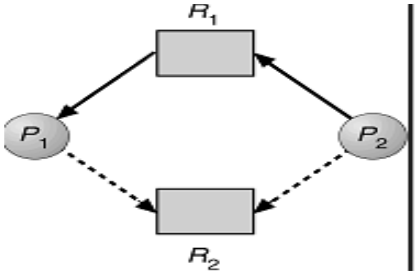
- Claim edge P_i

  P_i->R_j

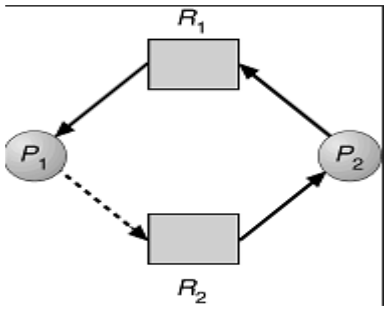
Indicated that process P_j may request resource R_j; represented by a dashed line.

- Claim edge converts to request edge when a process requests a resource.
- When a resource is released by a process, assignment edge reconverts to a claim edge.
- Resources must be claimed a priori in the system.

Resource allocation graph for dead lock avoidance (for one instance of each resource)



Unsafe state in resource allocation graph



4. Explain Banker's Algorithm (6) (APR 2012, NOV '15)

- Applicable to systems with multiple instances of each resource type.
- Each process must declare the maximum number of instance required for each resource type upon entering the system.
- When a process requests a set of resources, the system determines whether the allocation of these resources will have the system in a safe state.
- Yes: allocate the resources

- No: the process must wait

Data Structures for the Banker's Algorithm

- **Available**: Vector of length m. If available [j] = k, there are k instances of resource type R_j available.
- **Max**: n x m matrix. If Max [i,j] = k, then process P_i may request at most k instances of resource type R_j.
- **Allocation**: n x m matrix. If Allocation [i,j] = k, then P_i is currently allocated k instances of R_j.
- **Need**: n x m matrix. If Need [i,j] = k, then P_i may need k more instances of R_j to complete its task.

Need [i,j] = Max[i,j] – Allocation [i,j].

Safety Algorithm

1.Let Work and Finish be vectors of length m and n, respectively.

Initialize Work = Available

Finish [i] = false for i = 1, 2, 3, …, n.

2. Find an i such that both:

(a) Finish [i] = false

(b) $Need_i$<= Work

If no such i exists, go to step 4.

3. Work = Work + Allocation i

Finish[i] = true go to step 2.

4. If Finish [i] == true for all i, then the system is in a safe state.

Resource-Request Algorithm for Process *Pi*

$Request_i$ = request vector for process P_i.

If Request [j] = k then process P_i wants k instances of resource type R_j.

1. If $Request_i$ < =Need i go to step 2. Otherwise, raise error condition, since process has exceeded its maximum claim.

2. If $Request_i$< =Available, go to step 3. Otherwise Pi must wait, since resources are not available.

3. Pretend to allocate requested resources to Pi by modifying the state as follows:

Available = Available – $Request_i$;

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

$Need_i$ = $Need_i$ – $Request_i$;

- If safe the resources are allocated to Pi.
- If unsafe Pi must wait, and the old resource-allocation state is restored

5. Explain Deadlock Detection (8 Marks)? (NOV 2011) (NOV 2012)

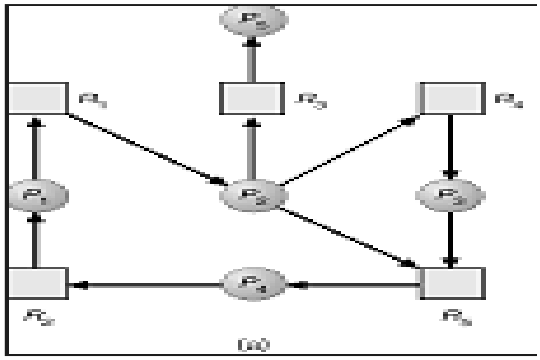
In an Deadlock detection the following methods are followed

- Allow system to enter deadlock state
- Detection algorithm
- Recovery scheme

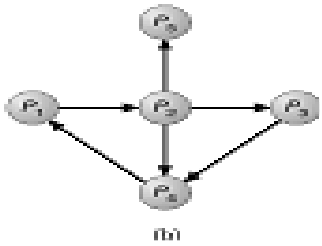
Single Instance of Each Resource Type

- Maintain wait-for graph Nodes are processes.
- P_i<P_j if P_i is waiting for P_j.
- Periodically invoke an algorithm that searches for a cycle in the graph.
- An algorithm to detect a cycle in a graph requires an order of n2 operations, where n is the number of vertices in the graph.

Several Instances of a Resource Type Resource allocation graph



Corresponding wait graph



- Available: A vector of length m indicates the number of available resources of each type.
- Allocation: An n x m matrix defines the number of resources of each type currently allocated to each process.
- Request: An n x m matrix indicates the current request of each process.
- If Request[i,j] = k, then process P_i is requesting k more instances of resource type R_j.

Detection Algorithm

1. Let Work and Finish be vectors of length m and n, respectively. Initialize as follows:

(a) Work = Available

(b) For i = 1,2, …, n,

If Allocation[i]#0, then Finish[i] = false;

Otherwise, Finish[i] = true.

2.Find an index i such that both:

(a) Finish[i] == false

(b) $Request_i$<=Work

If no such i exists, go to step 4.

3. Work = Work + $Allocation_i$

Finish[i] = true

go to step 2.

4. If Finish[i] == false, for some i, 1 <=i <=n, then the system is in deadlock state. Moreover,

If Finish[i] == false, then P_i is deadlocked.

Detection-Algorithm Usage

- When, and how often, to invoke depends on.
- How often a deadlock is likely to occur.
- How many processes will need to be rolled back.

One for each disjoint cycle

- If detection algorithm is invoked arbitrarily, there may be many cycles in the resource graph and so we would not be able to tell which of the many deadlocked processes "caused" the deadlock.

6. Write short notes on Recovery from Deadlock (5 Marks) (NOV 2011) (NOV 2012)

Process Termination

- Abort all deadlocked processes.
- Abort one process at a time until the deadlock cycle is eliminated.
- In which order should we choose to abort.
- Priority of the process.
- How long process has computed, and how much longer to completion.
- Resources the process has used.
- Resources process needs to complete.
- How many processes will need to be terminated.
- Is process interactive or batch system

Recovery from Deadlock:

Resource Preemption

- Selecting a victim – minimize cost.
- Rollback – return to some safe state, restart process for that state.

- Starvation – same process may always be picked as victim, include number of rollback in cost factor.

7. Explain about contiguous memory allocation (5Marks) (APR 2014)

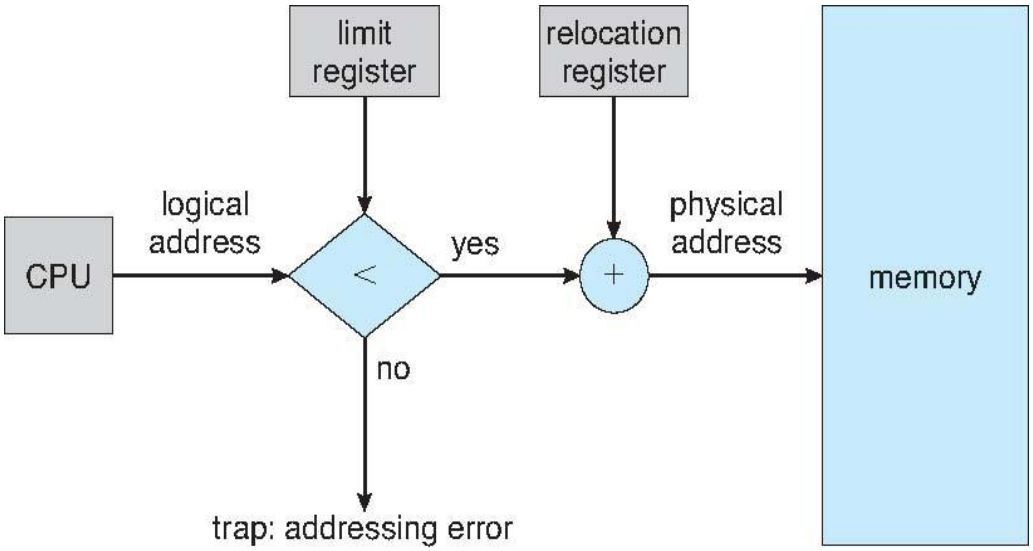
Contiguous Allocation

Main memory usually divided into two partitions: Resident operating system usually held in low memory with interrupt vector. User processes held in high memory.

Single-partition allocation

In the below figure the Relocation register scheme used to protect user processes from each other, and from changing operating system code and data. Relocation register contains value of smallest physical address, limit register contains range of logical addresses each logical address must be less than the limit register.MMU contains logical address dynamically.

Hardware support for relocation and limit register



Multiple partition allocation

In this method we can store several processes in memory at the same time before allocate the memory to variable processes they are waiting in ready queue one of the simplest methods for memory allocation is to divide the memory into a number of fixed size partition.

Each process may contain only one process when a partition is free; a process is selected from the input queue and is loaded into the free partition. When the process terminates the partition becomes available for another process. The os keeps a table indicating which parts of memory are available and which are occupied a free block of memory is called a hole.

When a process needs to be in the memory, a hole large enough to satisfy this request is found and is allocated to that process the rest is kept for future request.

The set of holes is searched first to determine which hole to allocate first fit, best fit and worst fit are the most common strategies used to select the free hole from the set of available holes.

First-fit: Allocate the *first* hole that is big enough.

Best-fit: Allocate the *smallest* hole that is big enough; must search entire list, unless ordered by size. Produces the smallest left over hole.

Worst-fit: Allocate the *largest* hole; must also search entire list. Produces the largest leftover hole.

Fragmentation:

-memory fragmentation can be internal as well as external

External fragmentation:

-External fragmentation exists when enough total memory space exits to satisfy a request; but it is not contiguous, storage is fragmented into large number of small holes.

-This fragmentation problem can be severe. There may be a block of free memory between every two processes.

Internal fragmentation:

-The memory allocated to a process may be slightly larger than the requested memoy.

-The difference between these two numbers is internal fragmentation-memory that is internal to partition but is not being used.

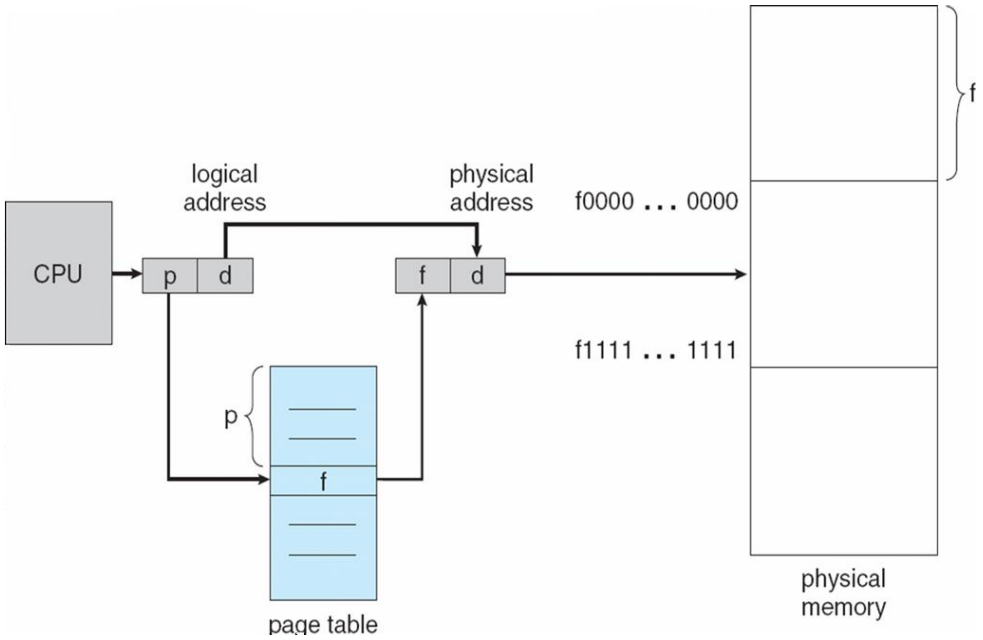
8. Explain Paging in detail? (6) (APR 2014, NOV '15)

It is a memory management scheme .This is another possible solution to the external fragmentation. This allow the physical address space of a process to be non-contiguous thus allowing a process to be allocated in physical memory where ever the free space is available.

Basic method:-

- Physical memory is broken into fixed size blocks called frames.
- Similarly the logical memory is broken into block of same size called pages.
- When a process is to be executed its pages are loaded into available memory frames from the backing store.
- The backing store is divided into fixed size blocks that are of the same size as memory frames.

Paging hardware algorithm:

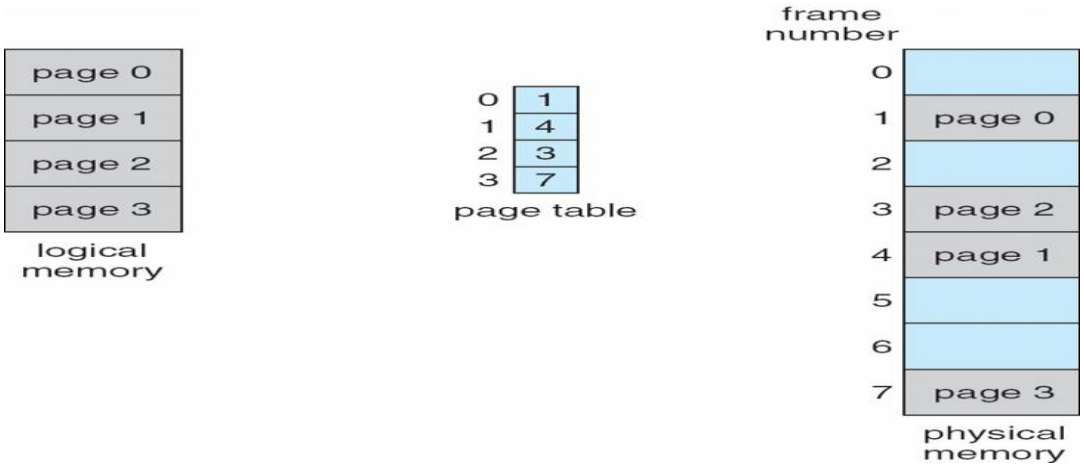


Every address generated by the CPU is divided into two parts: page number and page offset.

- The page number is used to index into a page table.
- The page table contains the base address of each page in physical memory.
- The base address is combined with the page offset to define the physical address that is sent to the memory unit.
- The page size is defined by the hardware and is usually power of 2 varying between 512bytes and 8192bytes per page.

Total logical address space 2m and Page size is 2n ,When address is created and m-n bite=page number (p) then n bits=displacement (d) p acts as an index to page table.

Paging example



Here the page size is 4 bytes and the size of the physical memory is 32 bytes (8 pages)Logical address zero maps to page 0 offset 0, indexing into page table. We find that page 0 is in frame 5,thus the logical address maps to physical address 20.Logical address 3(page 0, offset 3) maps to physical

address 23, logical address 4 is (page 1, offset 0) according to the page table, page 1 is mapped to frame 6.Thus logical address 4 maps to physical address 24.

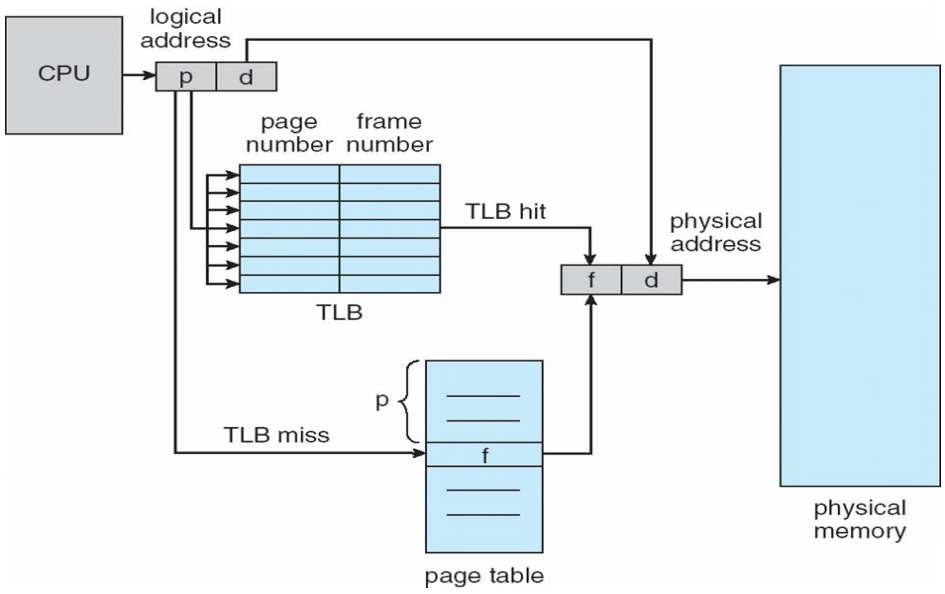
Notice that paging itself is form of dynamic relocation. Every logical address is bound by paging hardware to some physical address. When we use a paging scheme we have no external fragmentation We may have some internal fragmentation.

9. Explain TLB (6Marks)

TLB stands for Translation Look aside Buffer. In a cached system, the base addresses of the last few referenced pages is maintained in registers called the TLB that aids in faster lookup. TLB contains those page-table entries that have been most recently used. Normally, each virtual memory reference causes 2 physical memory accesses one to fetch appropriate page table entry, and one to fetch the desired data.

Using TLB in between, this is reduced to just one physical memory access in cases of TLB hit. Problem with paging is that, extra memory references to access translation tables can slow programs done by a factor of 2 or 3.Too many entries in the translation tables to keep them all loaded in fast processor memory. The standard solution to this problem is to use a special, small fast lookup hardware cache; called TLB.The TLB is associative high speed memory.

Paging hardware with TLB



A translation buffer is used to store a few of the translation table entries. It is very fast but only for a small numbers of entries.

On each memory reference has

1. First ask TLB it knows about the page the page, if so the reference proceeds fast.

2. If TLB has no information for page must go through page and segment table to get information. Reference takes a long time, but gives the information for this page to TLB so it will know it for next reference.

TLB hit: If the page member is found in TLB; TLB hit

TLB miss: If the page number is not found in TLB; TLB miss

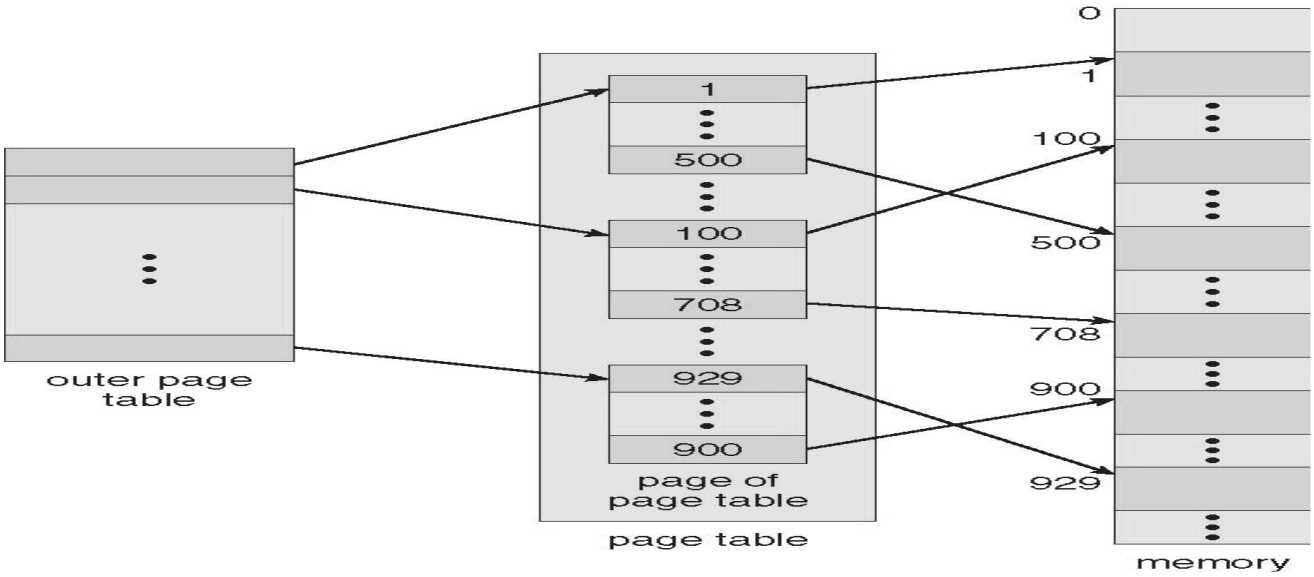
10. Describe the paging memory management scheme in details?(APR 2012)

Structure of the Page Table:

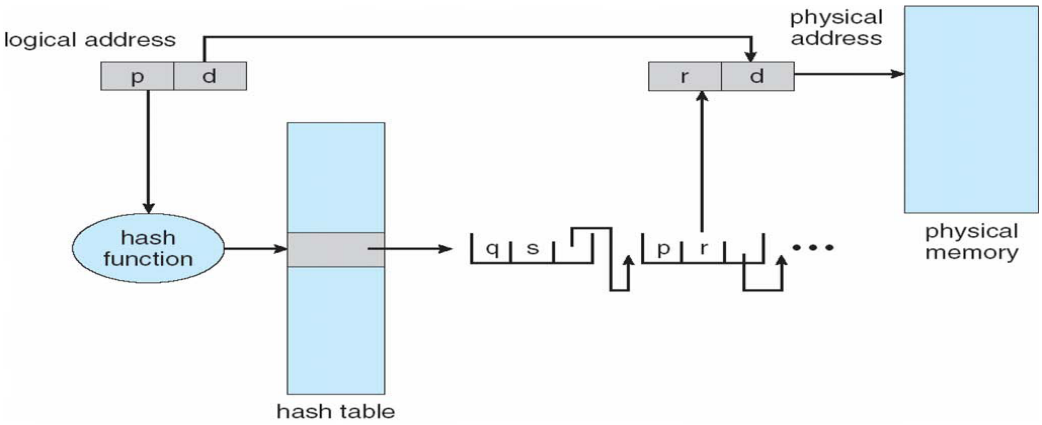
- Hierarchical Paging
- Hashed Page Tables
- Inverted Page Tables

Hierarchical Page Tables:

Recent computer system supports a large logical address space from 232 to 264.In this system the page table becomes large. It is difficult to allocate contiguous main memory for page table. To solves this problem; two level page table scheme is used. The page table is divided into number of smaller pieces.



Hashed Page Tables



Hashed Page Tables handles the address space larger than 32 bits .The virtual page number is used as hashed value. Linked list is used in the hash table. Each entry contains the linked list of element that hash to the same location.

Each element in the bash table contains the following fields

1. Virtual page number
2. Mapped page frame value
3. Pointer to the next element in the linked list

Working methods:

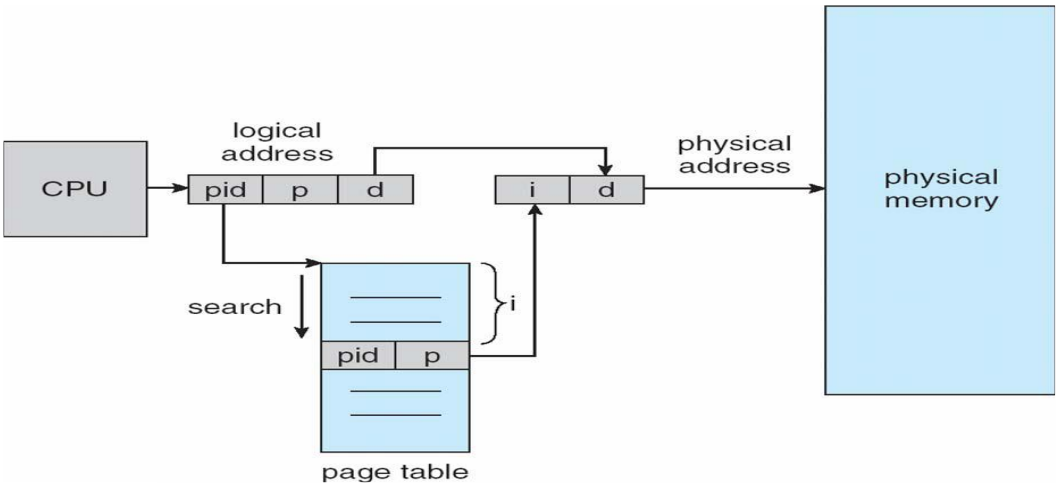
1. Virtual page number is taken from virtual address
2. Virtual page number is hashed with the hash table
3. Virtual page number is compared with the first element of linked list
4. Both the value is matched, that value (i.e. page frame) used for calculating physical address.
5. If both values are not matched the entire linked list is searched for a matching.

Inverted Page Table:

An address space have grown to 64 bits the size of traditional page table becomes a problem even with 2 level page tables the tables themselves can become too large. Inverted Page Table is used to solve this problem. The inverted Page Table has one entry for each real page of memory. A physical page table instead of a logical one. The physical page table is often calls as Inverted Page Table.

This table contains one entry per age frame .An Inverted Page Table is very good at mapping from physical page to logical page number, but not very good at mapping from virtual page number to physical page number.

There is no other hardware or registers dedicated to memory mapping the TLB can be quite larger, so that missing entry faults are rare. With an Inverted Page Table, most address translations are handled by the TLB when there is a miss in the TLB, the OS is notified and TLB miss handler is invoked. Hashing is used to speed up the search.



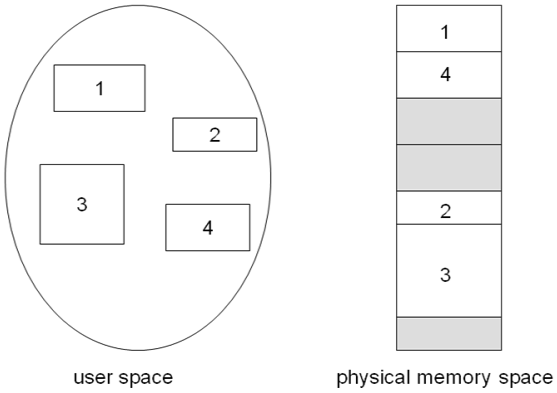
11. Discuss in Segmentation in details? (11) (NOV 13)

Segmentation

Segmentation is a Memory Management technique in which memory is divided into variable sized chunks which can be allocated to processes. Each chunk is called a segment. A table stores the information about all such segments and is called **Global Descriptor Table** (GDT). A GDT entry is called Global Descriptor. It comprise of: Logical Address to Linear Address.

A program is a collection of segments. A segment is a logical unit such as: main program, procedure, function, method, object, local variables, global variables, common block, stack, symbol table, arrays.

Logical View of Segmentation



- Logical address consists of a two tuple. <segment-number, offset>
- **Segment table** – maps two-dimensional physical addresses; each table entry has:

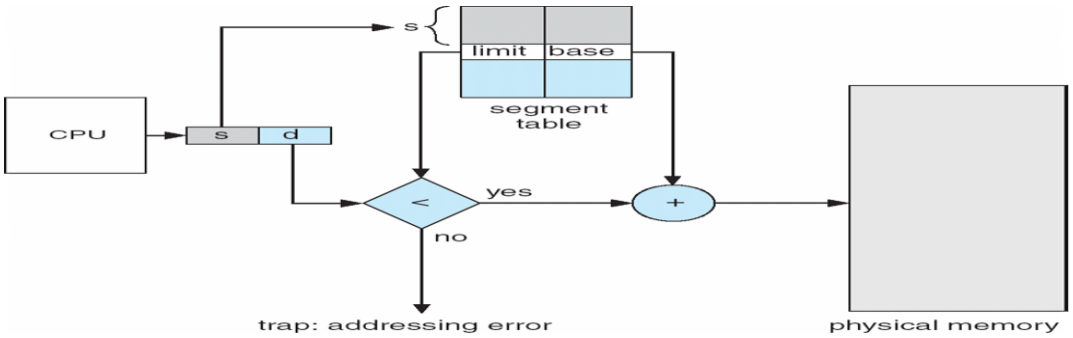
  base – contains the starting physical address where the segments reside in memory.

limit – specifies the length of the segment

- **Segment-table base register (STBR)** points to the segment table's location in memory.
- **Segment-table length register (STLR)** indicates number of segments used by a program.

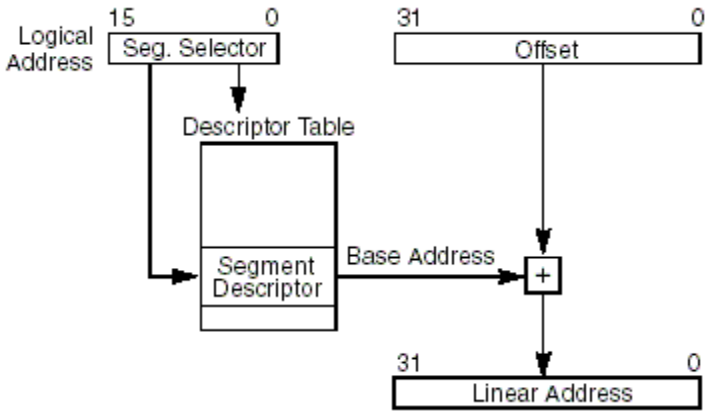
  Segment number ***s*** is legal if ***s*** < **STLR**

Segmentation Hardware



To translate a logical address into a linear address, the processor does the following:

1. Uses the offset in the segment selector to locate the segment descriptor for the segment in the GDT or LDT and reads it into the processor. (This step is needed only when a new segment selector is loaded into a segment register.)
2. Examines the segment descriptor to check the access rights and range of the segment to insure that the segment is accessible and that the offset is within the limits of the segment.
3. Adds the base address of the segment from the segment descriptor to the offset to form a linear address.



If paging is not used, the processor maps the linear address directly to a physical address (that is, the linear address goes out on the processor's address bus). If the linear address space is paged, a second level of address translation is used to translate the linear address into a physical address.

Segment Selector

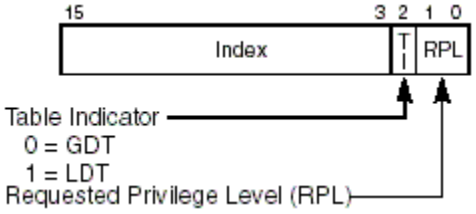
A segment selector is a 16-bit identifier for a segment. It does not point directly to the segment, but instead points to the segment descriptor that defines the segment. A segment selector contains the following items:

Index

Index bits 3 through 15. Selects one of 8192 descriptors in the GDT or LDT. The processor multiplies the index value by 8 the number of bytes in a segment descriptor and adds the result to the base address of the GDT or LDT from the GDTR or LDTR register, respectively.

TI (table indicator) flag

In table indication flag is Bit 2. Specifies the descriptor table to use: clearing this flag selects the GDT and setting this flag selects the current LDT.



Segmented Paging

Benefits: faster process start times, faster process growth, memory sharing between processes.

Costs: somewhat slower context switches, slower address translation.

- Pure paging system => (virtual address space)/ (page size) entries in page table.

12. Consider the following snapshot of a system: (11) (NOV 13)

	Allocation	**Max**	**Available**
	ABCD	ABCD	ABCD
Po	0012	0012	1520
P1	1000	1750	
P2	1354	2356	
P3	0632	0652	
P4	0014	0656	

a. What is the content of the matrix Need?
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?

A. the content of the matrix Need

	A	B	C	D
P0	0	0	0	0
P1	0	7	5	0
P2	0	0	2	0
P3	0	2	0	0

P4	0	6	4	2

B. the system in a safe state

P0 P2 P3 P4 P1

C.request from process P1 arrives for (0,4,2,0), can the request be granted immediately

Yes request from process P1 arrives for (0, 4, 2, 0), can the request be granted immediately .

13. Consider the following snapshot of a system: (11) (NOV 14)

	Allocation			Max		
	A	B	C	A	B	C
Po	0	0	2	2	0	3
P1	1	1	0	2	3	5
P2	0	0	1	1	2	3
P3	1	0	0	2	0	3
P4	0	0	2	0	1	5

Available :

A=0. B=2 c=1

The content of the matrix Need

	A	B	C
P0	2	0	1
P1	1	2	5
P2	1	2	2
P3	1	0	3
P4	0	1	3

The system is in unsafe state because the requirement of all the process are more than the available resources

Pondicherry University Questions

2 Marks

1. Define deadlock (UQ NOV'13) (Ref.Pg.No.1 Qn.No.1)
2. Why the page sizes is always power of 2 in memory? (UQ NOV'13) (Ref.Pg.No.3 Qn.No.13)
3. What is of resource allocation graph?(UQ APR'12) (Ref.Pg.No.4 Qn.No.18)
4. What is External Fragmentation? (APR 2014) (Ref.Pg.No.4 Qn.No.19)
5. What do you mean by best fit?(APR'12) (Ref.Pg.No.4 Qn.No.21)
6. What do you mean by first fit?(APR'12) (Ref.Pg.No.4 Qn.No.22)
7. What are the criteria for LRU page replacement algorithms?(APR 2012) (Ref.Pg.No.5 Qn.No.23)
8. Define Swapping (APR'15) (Ref.Pg.No.3 Qn.No.16)
9. What are conditions under which a deadlock situation may arise? Or Write four general strategies for dealing with deadlocks? (APR'15)(NOV'14)) (Ref.Pg.No.1 Qn.No.3)
10. Define paging. (NOV '15) (Ref.Pg.No.5 Qn.No.28)

11 Marks

1. Explain Deadlock Avoidance in details? (UQ APR'12) (Ref.Pg.No.8 Qn.No.3)
2. Describe bankers algorithms?(UQ APR'12, NOV '15) (Ref.Pg.No.9 Qn.No.4)
3. Explain Deadlock Detection? (UQ NOV'11& NOV'12) (Ref.Pg.No.11 Qn.No.5)
4. Write short notes on Recovery from Deadlock? (UQ NOV '11& NOV 12) (Ref.Pg.No.12 Qn.No.6)
5. Explain about contiguous memory allocation? (UQ APR'14) (Ref.Pg.No.13 Qn.No.7)
6. Explain Paging in detail? (UQ APR'14, NOV '15) (Ref.Pg.No.14 Qn.No.8)
7. Describe the paging memory management scheme in details?(UQ APR'12) (Ref.Pg.No.15 Qn.No.10)
8. Discuss in Segmentation in details?(UQ NOV'13) (Ref.Pg.No.19 Qn.No.11)
9. Consider the following snapshot of a system: (UQ NOV'13) (Ref.Pg.No.21 Qn.No.12)

	Allocation	Max	Available
	A B C D	A B C D	A B C D
Po	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	

a. What is the content of the matrix Need?

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?

10. Explain storage management .A system has 2 A resources 3 B and 6 C resources .5 processes their current allocation and their maximum allocation are shown below. Is the system in a safe state? If so ,show one sequence of processes which allow the system to complete .If not, explain why. (NOV'14) (Ref.Pg.No.22 Qn.No.13)

	Allocation			Max		
	A	B	C	A	B	C
Po	0	0	2	2	0	3
P1	1	1	0	2	3	5
P2	0	0	1	1	2	3
P3	1	0	0	2	0	3
P4	0	0	2	0	1	5

11. Explain Deadlock Prevention in detail? (APR'15) (Ref.Pg.No.7 Qn.No.2)

UNIT IV

Virtual Memory – Demand Paging – Process creation – Page Replacement – Allocation of frames – Thrashing. File Concept: Access Methods – Directory Structure – File System Mounting – File Sharing – Protection

1. What is virtual memory?(APR '11)(APR '12)

Virtual memory is a technique that allows the execution of processes that may not be completely in memory. It is the separation of user logical memory from physical memory. This separation provides an extremely large virtual memory, when only a smaller physical memory is available.

2. What is Demand paging?

Virtual memory is commonly implemented by demand paging. In demand paging, the pager brings only those necessary pages into memory instead of swapping in a whole process. Thus it avoids reading into memory pages that will not be used anyway, decreasing the swap time and the amount of physical memory needed.

3. Define lazy swapper.

Rather than swapping the entire process into main memory, a lazy swapper is used. A lazy swapper never swaps a page into memory unless that page will be needed.

4. Compare Swapper with Lazy Swapper?(NOV '13)

Rather than swapping the entire process into main memory, a lazy swapper is used. A lazy swapper never swaps a page into memory unless that page will be needed.

5. What is a pure demand paging?

When starting execution of a process with no pages in memory, the operating system sets the instruction pointer to the first instruction of the process, which is on a non-memory resident page, the process immediately faults for the page. After this page is brought into memory, the process continues to execute, faulting as necessary until every page that it needs is in memory. At that point, it can execute with no more faults. This schema is pure demand paging.

6. Define secondary memory.

This memory holds those pages that are not present in main memory. The secondary memory is usually a high speed disk. It is known as the swap device, and the section of the disk used for this purpose is known as swap space.

7. Consider a logical address space of eight pages of 1024 words each, mapped onto a physical memory of 32 frames. A. How many bits are there in the logical address? B. How many bits are there in the physical address?

Answer:

A. Logical address: 13 bits

B. Physical address: 15 bits

8. What is the basic approach of page replacement? If no frame is free is available, find one that is not currently being used and free it.

A frame can be freed by writing its contents to swap space, and changing the page table to indicate that the page is no longer in memory. Now the freed frame can be used to hold the page for which the process faulted.

9. What is the various page replacement algorithms used for page replacement?

- FIFO page replacement
- Optimal page replacement
- LRU page replacement
- LRU approximation page replacement
- Counting based page replacement
- Page buffering algorithm.

10. What are the major problems to implement demand paging?

The two major problems to implement demand paging is developing

a. Frame allocation algorithm

b. Page replacement algorithm

11. What is a reference string?

An algorithm is evaluated by running it on a particular string of memory references and Computing the number of page faults. The string of memory reference is called a reference string.

12. What is thrashing? (APR 11, NOV '15)

Thrashing occurs when a computer's virtual memory subsystem is in a constant state of paging, rapidly exchanging data in memory for data on disk, to the exclusion of most application-level processing. This causes the performance of the computer to degrade or collapse. The situation may continue indefinitely until the underlying cause is addressed.

13. Write any two causes of Thrashing?

In virtual memory systems, thrashing may be caused by programs or workloads that present insufficient locality of reference: if the working set of a program or a workload cannot be effectively held within physical memory, then constant data swapping, *i.e.,* thrashing, may occur. The term was first used during the tape operating system days to describe the sound the tapes made when data was being rapidly written to and read from them. Many older low-end computers have insufficient RAM (memory) for modern usage patterns and increasing the amount of memory can often cause the computer to run noticeably faster. This speed increase is due to the reduced amount of paging necessary.

14. State the functions of working set strategy model (NOV'14)

To prevent thrashing, *we* must provide a process with as many frames as it needs. To find how many frames a process needs for that a technique called working set strategy used . The working-set strategy starts by looking at how frames a process is actually using. This approach defines the locality of process execution. The locality model states that, as a process executes, it moves from locality to locality. *A* locality is a set of pages that are actively used together . Allocate enough frames to a process to accommodate its current locality. It will fault for the pages in its locality until all these pages are in memory; then, it will not fault again until it changes localities.

15. Enumerate Search Path? (NOV '13)

When you enter a command, the operating system has to search for that program. The path defines the directories which the system will search. The current directory is NOT automatically included in the search path.

To add the directory /usr/local/bin to the existing path, use the command: PATH=$PATH:/usr/local/bin

16. What is a file?

A file is a named collection of related information that is recorded on secondary storage. A file contains either programs or data. A file has certain "structure" based on its type.

- File attributes: Name, identifier, type, size, location, protection, time, date
- File operations: creation, reading, writing, repositioning, deleting, truncating, appending, renaming
- File types: executable, object, library, source code etc.

17. What is Access method?

An access method defines the technique that is used to store and retrieve data. Access methods have their own data set structures to organize data, system-provided programs (or **macros**) to define data sets, and utility programs to process data sets.

18.What are the possible methods to be used to access file?

There are two possibilities: sequential access and random access.

Sequential access: Each access to a given file starts where the previous access to that file finished (the first access to the file starts at the beginning of the file). Sequential access is the most common and gives the highest performance. For some devices (e.g. magnetic or paper tape) access must be sequential.

R**andom access**: The bytes are accessed in any order. Thus each access must specify which bytes are desired. This is done either by having each read/write specify the starting location or by defining another system call (often named seek) that specifies the starting location for the next read/write.

19. List the various File Attributes? (APR '12)(NOV '12)

- **Name** – only information kept in human-readable form
- **Identifier** – unique tag (number) identifies file within file system
- **Type** – needed for systems that support different types
- **Location** – pointer to file location on device
- **Size** – current file size
- **Protection** – controls who can do reading, writing, executing
- **Time, date, and user identification** – data for protection, security, and usage monitoring

20. What are the various file operations?

- **Create:** The effect of create is essential if a system is to add files. However, it need not be a separate system call. (For example, it can be merged with open).
- **Delete:** Essential, if a system is to delete files.
  - **Open**: Not essential. An optimization in which a process translates a file name to the corresponding disk locations only once per execution rather than once per access. We shall see that for the Unix inode-based file systems, this translation can be quite expensive.
  - **Close:** Not essential. Frees resources without waiting for the process to terminate.
  - **Read**: Essential Must specify filename, file location, number of bytes, and a buffer into which the data is to be placed. Several of these parameters can be set by other system calls and in many operating systems they are.
  - **Write:** Essential, if updates are to be supported. See read for parameters.
  - **Seek:** Not essential (could be in read/write). Specify the offset of the next (read or write) access to this file.
  - **Get attributes**: Essential if attributes are to be used.
  - **Set attributes**: Essential if attributes are to be user settable.
  - **Rename:** Copy and delete is not an acceptable substitute for big files. Moreover, copy-delete is not atomic.

21. Write some file type and its extension?

File Type	Usual extension	Function
executable	Exe,com,bin or none	Read to run machine language program
object	Obj,O	Compiled ,machine language, not linked
Source code	c,cc,java,pas,asm,a	Source code in various language
batch	Bat,sh	Commands to the command interpreter
text	txt,doc	Textual data,documents
Word processor	Wp,tex,rrf,doc	Various word processor formats
library	Lib,a,mpeg,mov,rm	Libraries of routines for programmers.

22.What are the information associated with an open file?

Several pieces of information are associated with an open file which may be:

- File pointer
- File open count
- Disk location of the file
- Access rights

23. What are the different accessing methods of a file? (APR'15)

The different types of accessing a file are:

• Sequential access: Information in the file is accessed sequentially

• Direct access: Information in the file can be accessed without any particular order.

• Other access methods: Creating index for the file, indexed sequential access method (ISAM) etc.

24.What is Access control?

The most common approach to the protection problem is to make access dependent on the identity of the user. Different users may need different types of access to a file or directory. The most general scheme to implement dependent access is to associate with each file and directory specifying user names and the types of access allowed for each user. When a user requests access to a particular file, the operating system checks the access list associated with that file. If that user is listed for the requested access, the access is allowed. Otherwise, a protection violation occurs, and the user job is denied access to the file.

25.What are the advantages and disadvantages of access control? (NOV'14)

The access control approach has the advantage of enabling complex access methodologies. The main problem with access lists is their length. If we want to allow everyone to read a file, we must list all users with read access. This technique has two undesirable consequences:

1. Constructing such a list may be a tedious and unrewarding task, especially if we do not know in advance the list of users in the system.
2. The directory entry, previously of fixed size, now must be of variable size, resulting in more complicated space management.

26. What is Directory?

The device directory or simply known as directory records information-such as name, location, size, and type for all files on that particular partition. The directory can be viewed as a symbol table that translates file names into their directory entries. OR Directory is collection of files. **Directories** maintain information about files: For a large number of files, may want a directory structure is directories under directories.

27. What are the operations that can be performed on a directory?

The operations that can be performed on a directory are

• Search for a file

- Create a file
- Delete a file
- Rename a file
- List directory
- Traverse the file system

28. What are the most common schemes for defining the logical structure of a directory?

The most common schemes for defining the logical structure of a directory

- Single-Level Directory
- Two-level Directory
- Tree-Structured Directories
- Acyclic-Graph Directories
- General Graph Directory

29. Define UFD and MFD.

In the two-level directory structure, each user has her own user file directory (UFD). Each UFD has a similar structure, but lists only the files of a single user. When a job starts the system's master file directory (MFD) is searched. The MFD is indexed by the user name or account number, and each entry points to the UFD for that user.

30. What is a path name?

A pathname is the path from the root through all subdirectories to a specified file. In a two-level directory structure a user name and a file name define a path name.

31. What is System file checker in Windows?(NOV '11)

System File Checker (sfc)is a utility in Windows that allows users to scan for corruptions in Windows system files and restore corrupted files. Scans and verifies the versions of all protected system files after you restart your computer.

32. Give any two criteria to chose a file organization? (APR '12)

1. Fast access to single record or collection of related records.
2. Easy record adding,updating,removal without disrupting.
3. Storage efficiency.

33. How information is maintained in is Directory?

Information maintained in a directory:

Name	The user visible name.
Type	The file is a directory, a program image, a user file, a link, etc.
Location	Device and location on the device where the file header is located.
Size	Number of bytes/words/blocks in the file.
Position	Current next-read/next-write pointers.
Protection	Access control on read/write/ execute/delete
Usage	time of creation/access, etc.

34. What is Directory Structure?

In an Partitions (or Volumes) can be viewed as the abstraction of virtual disks. Disks can be partitioned into separate areas such that each partition is treated as a separate storage device. The other way a partition may be defined to be more than one disk device than one. Partitions can store multiple operating systems such that a system can boot more than one OS.Each partition contains information about files in a device directory (or) a VTOC(Volume Table of Contents)is a device directory Each directory records file attribute information.

35. What are the operations that can be performed on a directory?

Search for a file – need to find a particular entry or be able to find file names based on a pattern match.

Create a file -and add its entry to the directory.

Delete a file – and remove remove it from the directory.

List a directory –list both the files in the directory and the directory contents for each file.

Rename a file –renaming may imply changing the position of the file entry in the directory structure.

Traverse the file system –the directory needs a logical structure such that every directory and every file within each directory can be accessing efficiently.

36. Mention the Directory design goals?

- Efficiency to locating a file quickly.
- Naming is convenient to users.
  - ✦Two users can have same name for different files.
  - ✦The same file can have several different names.
- Grouping in a logical grouping of files by properties, (e.g., all Java programs, all games, …)

37. What is File Lock? (NOV '11) (APR '11)

File locking is a mechanism that restricts access to a computer file by allowing only one user or process access at any specific time. Systems implement locking to prevent the classic interceding update scenario (race condition).It may also refer to additional security applied by a computer user either by using Windows security, NTFS permissions or by installing a third party file locking software.

38. What is File Control Block? (APR '14)

A File Control Block (FCB) is a file system structure in which the state of an open file is maintained. An FCB is managed by the operating system, but it resides in the memory of the program that uses the file, not in operating system memory. This allows a process to have as many files open at one time as it wants to, provided it can spare enough memory for an FCB per file.

39. Define Single-Level Directory?

A single-level directory with file entries for all users contained in the same directory.

Advantages:

Easy to support and understand.

Disadvantages:

Requires unique file names unique file names unique file names {the naming problem}.

No natural system for keeping track of file names {the grouping problem}

40. Define Two Level Directory?

Two Level Directory is a separate directory for each user. The system's Master File Directory (MFD) Master File Directory (MFD) Master File Directory (MFD) has pointers to individual User File Directories (UFD's).

File names default to localized UFD for all operations.

Advantages:

- Solves the name-collision problem.
- Isolates users from one another a form of protection.
- Efficient searching.

Disadvantages:

- Restricts user cooperation.

- No logical grouping capability (other than by user).

41. Define Tree-Structured Directories?

This generalization to a directory tree structure of arbitrary height allows users to create their own subdirectories and organize their files accordingly.

Advantages:

- Efficient searching
- Grouping Capability
- Each user has a current directory(working directory)

  cd /spell/mail/prog

  Type list

42. Define Acyclic-Graph Directories?

A tree structure prohibits the sharing of files or directories. Acyclic graphs allow directories to have shared subdirectories and files.

43. What is File System Mounting?

A file system must be mounted before it can be available to processes on the system. The mount procedure the OS is given the device name and the location within the file structure name location within the file structure location within the file structure at which to attach the the file system. {the mount point}.A mount point is typically an empty directory where the mounted file system will be attached.The OS verifies that device has valid file system by asking device driver to read the device directory and verify that directory has the proper format.

44. What is File Authentication?(NOV '11)(ARPIL '11)

An important aspect of security on a computer system is the granting or denying of permissions (sometimes called access rights). Permission is the ability to perform a specific operation such as to gain access to data or to execute code. Permissions can be granted at the level of directories, subdirectories, files or applications, or specific data within files or functions within **applications.**

45. Define File Sharing?

- Sharing of files on multi-user systems is desirable.
- Sharing may be done through a protection.
- On distributed systems, files may be shared across a network .

- Network File System (NFS) is a common distributed file sharing method.

46. How will you protect the file system?

Protecting the File system the File owner/creator should be able to control: what can be done and by whom

The following access methods are to protecting file

1. Read
2. Write
3. Execute
4. Append
5. Delete
6. List

47. What is the principle of optimality? (NOV '15)

Optimality is the best solution that can be achieved. In page replacement algorithm, the best replacement technique can be achieved by using optimal page replacement algorithm. It is simply replaces the page that will not be used for the longest period of time.

11 MARKS

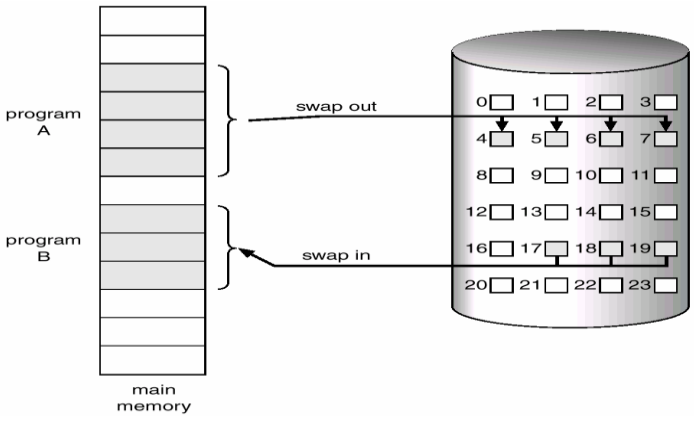
1. Explain demand paging in detail? (11)

Demand paging:

As there is much less physical memory than virtual memory the operating system must be careful that it does not use the physical memory inefficiently. One way to save physical memory is to only load virtual pages that are currently being used by the executing program. For example, a database program may be run to query a database. In this case not the entire database needs to be loaded into memory, just those data records that are being examined. Also, if the database query is a search query then the it does not make sense to load the code from the database program that deals with adding new records. This technique of only loading virtual pages into memory as they are accessed is known as demand paging.

Transfer of a paged memory to contiguous disk space

- It is similar in paging system with swapping.
- Processes resides on secondary memory, when we want to execute a process, we swap it into memory.
- Rather than swapping the entire process into memory, we can use a **lazy swapper**.
- A lazy swapper never swaps a page into memory unless that page will be needed. Swapper that deals with pages is a pager.
- When a process is to be swapped in, instead of swapping a whole process the page brings in only those necessary pages into memory.



- Thus it avoids reading into memory pages that will not be used anyway, decreasing the swap time and the amount of physical memory needed.
- With this scheme we need some hardware support to distinguish between those pages that are in memory and those pages that are in disk.

- The valid, invalid bit scheme can be used for this purpose.
- When the bit is set it valid it indicates that the associated is both legal and in memory.
- If the bit set to invalid is not valid or is valid but not in memory.

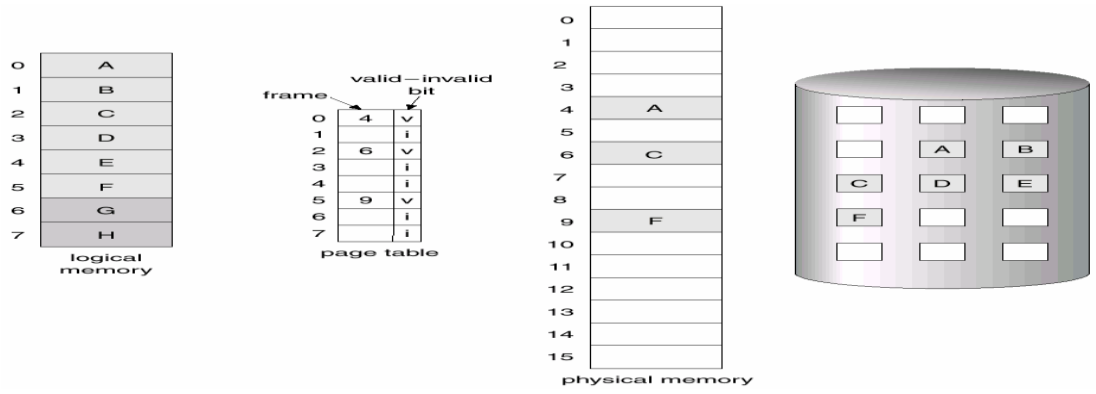
Page table when some pages are not in main memory:

- With each page table entry a valid–invalid bit is associated (v ⇒ in-memory, i ⇒ not-in-memory)
- Initially valid–invalid bit is set to i on all entries.
- During address translation, if valid–invalid bit in page table entry is **I** ⇒ page fault

Example of a page table snapshot:

If there is a reference to a page, first reference to that page will trap to operating system:page fault

1. Operating system looks at another table to decide:
   - Invalid reference ⇒ abort
   - Just not in memory
2. Get empty frame
3. Swap page into frame
4. Reset tables
5. Set validation bit = **v**
6. Restart the instruction that caused the page fault



Page fault

- Memory access for a legal page not in memory causes a page fault trap.
- Needs to bring the page into memory.

Steps of bringing a page into memory

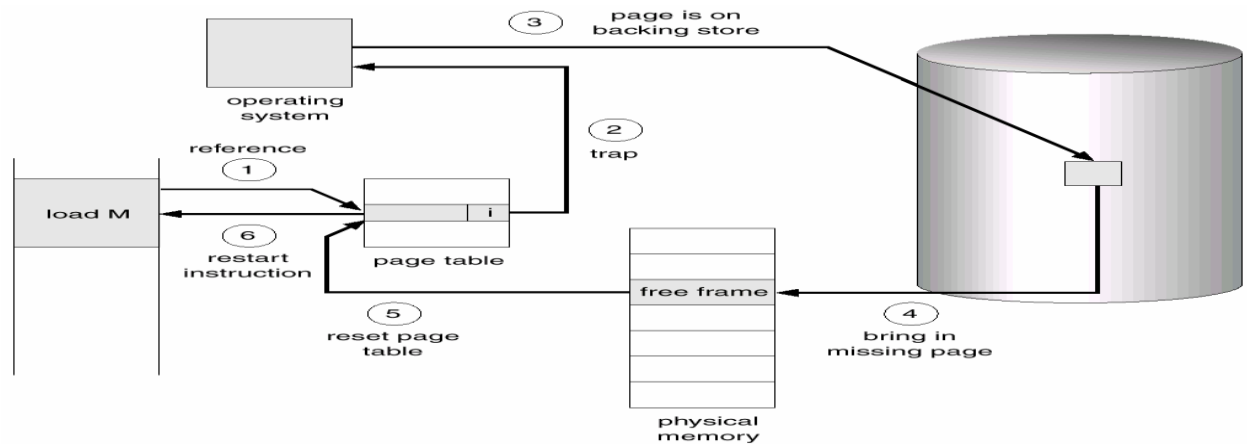
1. Find a free frame
2. Read the desired page from disk into the free frame
3. Update page table valid bit set to v.
4. Restart the interrupted instruction

Pure demand paging

Start executing a process with no pages in memory. Bring a page into memory only when needed.

Steps in handling a page fault:

- We check an internal table for this process to determine whether the reference was valid or invalid memory access.
- If the reference was invalid, we terminate the process. If the reference was valid but we have not yet bought in that page
- Now we find a free frame from free frame list.
- We Schedule a disk operation to read the desired page into newly allocated frame.
- When the disk read is complete, we need to modify the internal table which is kept with the process and page table to indicate the page form the memory.
- We restart the instruction that was interrupted by the illegal address trap.
- The process now can access the page as though it had always been in the memory.
- Restart instruction
- block move
- auto increment/decrement location



In the above figure Page Fault Rate $0 \le p \le 1.0$

- if $p = 0$ no page faults
- if $p = 1$, every reference is a fault

Effective Access Time (EAT)

EAT = (1–p)x memory access+ p(page fault overhead + swap page out+ swap page in+ restart overhead)

2. Explain page replacement in detail? (11) (APR'15, NOV '15)

Page replacement

When a fault occurs, the OS loads the faulted page from disk into a page of memory. At some point, the process has used the entire page frames it is allowed to use. When this happens, the OS must *replace* a page for each page faulted in. That is, it must select a page to throw out of primary memory to make room. How it does this is determined by the page replacement algorithm. The goal of the replacement algorithm is to reduce the fault rate by selecting the best victim page to remove.

If total memory requirements exceed the physical memory, then it may be necessary to replace pages from memory to free frames for new pages.

Page replacement algorithms:

1. FIFO Page Replacement
2. Optimal Page Replacement
3. LRU Page Replacement
4. LRU Approximation Page Replacement
    a. Additional Reference Bits Algorithm
    b. Second Chance Algorithm:
    c. Enhanced Second Chance Algorithm
5. Counting Based Page Replacement
6. Page buffering algorithm

To illustrate the page replacement algorithms, we shall use the reference string

7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

FIFO Page Replacement:

- A FIFO replacement algorithm associates with each page the time when that page was brought into memory.
- When a page must be replaced, the oldest page is chosen.
- We can create a FIFO queue to hold all pages in memory; we replace the page at the head of the queue.
- When the page is brought into the memory it is inserted at the tail of the queue.
- For the given reference string the 3 frames are initially empty.
- The first 3 references (7,0,1) causes page faults and are brought into these empty frames.
- The next reference (2) replaces page 7 because page 7 was brought in first.

The process continuous as below

Given string 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

7 7 7 2 2 2 4 4 4 0 0 0 7 7 7

0 0 0 3 3 3 2 2 2 1 1 1 0 0

1 1 1 0 0 0 3 3 3 2 2 1

- The FIFO page replacement algorithm is easy and its performance is not always good.
- This algorithm yields 15 page faults

Optimal Page Replacement:

- An optimal page replacement algorithm has lowest page fault rate of all algorithms.
- An optimal page replacement algorithm exist and is called OPT or MIN.
- It is simply replaces the page that will not be used for the longest period of time.
- This algorithm yields 9 page faults.

Given string 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

7 7 7 2 2 2 2 2 7

0 0 0 0 4 0 0 0

1 1 3 3 3 1 1

The reference to page 2 replaces page 7 because it will not be used until reference 18 ,whereas page 0 will be used at 5 and page 1 at 14.

- No replacement algorithm can process this string in 3 frames with than 9 faults.
- But this algorithm is difficult to implement it is mainly used in comparison studies.

LRU Page Replacement:

- LRU stands for least recently used
- LRU replacement associates with each page the time of that page's last use.
- When a page must be replaced LRU chooses a page that has not been used for the longest period of time.
- This strategy is apt for looking backward in time.
- The LRU produces 12 page faults .When reference to page4 occurs LRU selects page2 because that was used least recently. Then LRU algorithm replaces page 2 .This process continues for the entire reference string.
- LRU with 12 faults is better than FIFO with 15 faults.
- The major problem with this algorithm is, how to implement LRU replacement .It requires substantial hardware assistance.

- Given string 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

7 7 7 2 2 4 4 4 0 1 1 1

0 0 0 0 0 0 3 3 3 0 0
1 1 3 3 2 2 2 2 7

1. Counters:

In the simple case, we associated with each page table entry a time of use field and add to the CPU a logical clock or counter. The clock is incremented for every memory reference.We replace the page with the smallest time value. This scheme requires a search of the page table to fine for each memory access. The times must also be maintained when page tables are changed.

2. Stack:

Another approach to implementing LRU replacement is to keep a stack of page numbers. Whenever a page is referenced, it is removed from the stack and put on the top. In this way, the most recently used page is always at the top of the stack and the least recently used page is always at the bottom.

Removing a page and putting it on the top of the stack then requires changing six pointers at worst. Each update is a little more expensive, but there is no search for a replacement, the tail pointer points to the bottom of the stack, which is the LRU page.

LRU Approximation Page Replacement:

Additional Reference Bits Algorithm:

At regular intervals a timer interrupt transfers control to the operating system. The operating system shifts the reference bit for each page into the high order bit of its 8bit byte, shifting the other bits right by 1 bit and discarding the low order bit.

These 8 bit shift registers contain the history of page use for the last eight time periods. If the shift register contains 00000000 for example then the page has not been used for eight time periods a page that is used at least once in each period has a shift register value of 1111111.

The number of bits of history can be varied of course and is selected to make the updating as fast as possible. In the extreme case, the number can be reduced to zero, leaving only the reference bit itself. This algorithm is called the second chance page replacement algorithm.

Second Chance Algorithm:

The basic algorithm of second chance replacement is a FIFO replacement algorithm. When a page has been selected, however, we inspect its reference bit. If the value is 0, we proceed to replace this page, but if the reference bit is set to 1, we give the page a second chance and move on to select the next FIFO page. One way to implement the second chance algorithm is as a circular queue. A pointer indicates which page is to be replaced next.

Enhanced Second Chance Algorithm:

We can enhance the second chance algorithm by considering the reference bit and the modify bit as an ordered pair. With these two bits, we have the following four possible classes:

1. (0,0) neither recently used nor modified then best page to replace.
2. (0, 1) not recently used but modified then not quite as good, because the page will need to be written out before replacement.
3. (1, 0) recently used but clean then probably will be used again soon.
4. (1, 1) recently used and modified then probably will be used again soon, and the page will be need to be written out to disk before it can be replaced.

The major difference between this algorithm and the simpler clock algorithm is that here we give preference to those pages that have been modified to reduce the number of I/Os required.

Counting Based Page Replacement:

A counter of the number of references that have been made to each page and develop the following two schemes.

1. LFU
2. MFU

1. Least **F**requently **U**sed (**LFU):**

The least frequently used (LFU) page replacement algorithm requires that the page with the smallest count be replaced. The reason for this selection is that an actively used page should have a large reference count.

A problem arises, however, when a page is used heavily during the initial phase of a process but then is never used again. Since it was used heavily, it has a large count and remains in memory even though it is no longer needed.

2. Most **F**requently **U**sed (**MFU):**

The most frequently used (MFU) page replacement algorithm is based on the argument that the page with the smallest count was probably just brought in and has yet to be used.

Page Buffering Algorithms:

a. A system commonly has a pool of free frames.
b. In this algorithm, when a page fault occurs, a victim page is chosen as before. However the desired page is stored into a free frame before the victim is written out.
c. This procedure allows the process to restart as soon as possible, without waiting for the victim page to be written out.
d. When the victim page is later written out then its frame is added to the free frame pool.

3. Explain Thrashing? (APR '11) (APR '14) (NOV '15)

Thrashing occurs when a computer's virtual memory subsystem is in a constant state of paging, rapidly exchanging data in memory for data on disk, to the exclusion of most application level processing. This causes the performance of the computer to degrade or collapse. The situation may continue indefinitely until the underlying cause is addressed.

If a process does not have enough pages, thrashing is a high paging activity, and the page fault rate is high. This leads to low CPU utilization. In modern computers, thrashing may occur in the paging system (if there is not sufficient physical memory or the disk access time is overly long), or in the communications system especially in conflicts over internal bus access . Depending on the configuration and algorithms involved, the ***throughput*** and ***latency*** of a system may degrade by multiple orders of magnitude.

Causes

In virtual memory systems, thrashing may be caused by programs or workloads that present insufficient locality of reference: if the working set of a program or a workload cannot be effectively held within physical memory, then constant data **swapping thrashing** may occur. The term was first used during the tape operating system days to describe the sound the tapes made when data was being rapidly written to and read from them. Many older low-end computers have insufficient RAM (memory) for modern usage patterns and increasing the amount of memory can often cause the computer to run noticeably faster. This speed increase is due to the reduced amount of paging necessary.

An example of this sort of situation occurred on the IBM System/370 series mainframe computer, in which a particular instruction could consist of an execute instruction (which crosses a page boundary) that points to a move instruction (which itself also crosses a page boundary), targeting a move of data from a source that crosses a page boundary, to a target of data that also crosses a page boundary. The total number of pages thus being used by this particular instruction is eight, and all eight pages must be present in memory at the same time. If the operating system allocates fewer than eight pages of actual memory, when it attempts to swap out some part of the instruction or data to bring in the remainder, the instruction will again page fault, and it will thrash on every attempt to restart the failing instruction.

Solutions

To resolve thrashing due to excessive paging, a user can do any of the following:

- Increase the amount of RAM in the computer (generally the best long-term solution).
- Decrease the number of programs being run on the computer.
- Replace programs that are memory heavy with equivalents that use less memory.

4. What are files and explain the access methods for files? (APR '14, NOV '15)

- A file is an abstract data type. The operating system can provide system calls to create, write, read, reposition, delete and truncate files.
- A file is a named collection of related information that is recorded on secondary storage.
- A files contains either programs or data.
- File is a sequence of bits, bytes, lines as defined by the files creater and user.
- A file has certain 'structure' based on its type.
- Operating system must do for each of the six basic file operations.

  Text files: Sequence of characters organized into lines/pages.

  Source file: Sequence of subroutines and functions organized as executable statements.

  Object file: Sequence of bytes organized into blocks understandable by systems linker.

  Executable file: Series of code sections that loader can bring into memory and execute.

File attributes:

Name : only information kept in human – readable

Identifier :unique tag identifies file within file system.

Type :needed for systems that support different types.

Location : pointer to file location on device.

Size : current file size.

Protection : controls who can do reading, writing, executing.

File operations:

Time, date and user identification: data for protection, security and usage monitoring.

1. Creating a file
2. Writing a file
3. Reading a file
4. Repositioning within a file
5. Deleting a file
6. Truncating a file

Creating a file:

- Two steps are necessary to create a file
- space in the file system must be found for the file
- an entry for the new file must be made in the directory
- The directory entry records the name of the file and the location in the file system,and possibly other information.

Writing a file:

- To write a file, we make a system call specifying both the name of the file and the information to be written to the file.
- The system searches the directory to find the location of the file.
- The system must keep a write pointer to the location in the file where the next write is to take place.
- The write pointer must be updated whenever a write occurs.

Reading a file:

- To read from a file, we use a system call that specifies the name of the file and where(in memory) the next block of the file should be put.
- The directory is searched for the associated directory entry, and the system needs to keep a read pointer to the location in the file where the next read is to take place.
- Once the read has taken place, the read pointer is updated.
- The current operation location is kept as a per-process current-file-position-pointer.

Repositioning within a file:

- The directory is searched for the appropriate entry, and the current-file-position is set to a given value.
- Repositioning within a file does not need to involve any actual I/O.
- This file operation is also known as file seek.

Deleting a file:

To delete a file, we search the directory for the named file. Having found the associated directory entry, we release all file space, so that it can be reused by other files, and erase the directory entry.

Truncating a file:

The user may want to erase the contents of a file but keep its attributes. Rather than forcing the user to delete the files and the recreate it, this function allows all attributes to remain unchanged.

- These six basic operations certainly comprise the minimal set of required file operations. Other common operations include appending new information to the end of an existing file and renaming an existing file. Most of the file operations mentioned involve searching the directory for the entry associated with the named file.
- The operating system keeps a small call be used before that file is first about all open files. Pieces of information are associated with an open file.

File pointer:

This pointer is unique to each process operating on the file and therefore must be kept separate from the on disk file attributes.

File open count:

This counter tracks the number of opens and closes and reaches zero on the last close. The system can then remove the entry.

Disk location of the file:

The information needed to locate the file on disk is kept in memory to avoid having to read it from disk for each operation.

Access rights:

The information is stored on the per-process table so the operating system can allow or deny subsequent I/O requests.

Access Methods:

- It is used to access the information of file and read into computer memory.
- The information of file can be accessed in many ways.
- Some systems provide only one access methods for file.

There are 2 access methods

1. Sequential access
2. Direct access

Sequential access:

- Sequential access is simplest access method.
- Information in the file is processed in order, by one after the other.
- Read and write made up of the bulk of operations on a file
- Read operation: reads the next portion of the file and automatically advances a file pointer, which tracks the I/O location.
- Write operation: appends to the end of the file and advances to the end of newly written material.
- File can be reset to beginning and on some system ,a program may be skip forward or backward n records for integer
- Sequential access file is shown in figure Beginning current position end
- Sequential access file is based on tape model of file.

Direct access:

- Direct access is also called as relative access.
- A file is made up of fixed length logical records that allow programs to read and write records rapidly.
- It is based on disk method of a file, since disk allow random access to any file block
- In direct access, file is viewed as numbered sequence of blocks or records
- In direct access, there is no restriction for reading and writing.
- It is great use for immediate access to large amounts of information.
- In this method, the file operation must be modified to include the block number as a parameter.
- Thus, we have read n where block number is rather than read next.
- This block number provide by the user to operating system is normally a relative block number.
- Relative block number is a index relative to the beginning of the file.
- Not all operating system support both sequential and direct access for files.
- Some system require that a file as to be defined as sequential or direct access when it is created so such file can be accessed by consistent with its declaration.

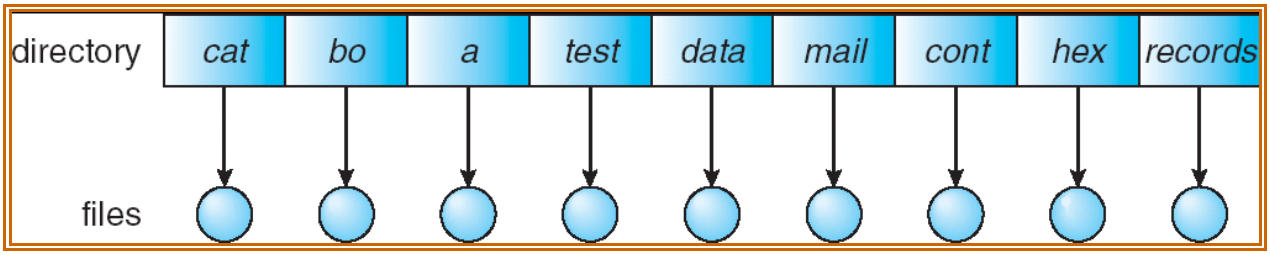
Other access methods:

It can be built on top of direct access method and involve the construction of an index for the file. The index contains pointers to various blocks. To find the desired record in the file, we first search the index and then use the pointers to access the file directly.

5. Write short notes on single and two-level Directory? (APR '11)

Single level directory:

The Single level directory is simplest directory structure because is easy to support and understand files in system. It as limitation when the number of files increase or when the system as more than one user .All files are in same directory so they must have unique names. Even a single user on Single –level directory may find difficult to remember names of all files as number of files increase. It keep tracks the files is a daunting tasks.



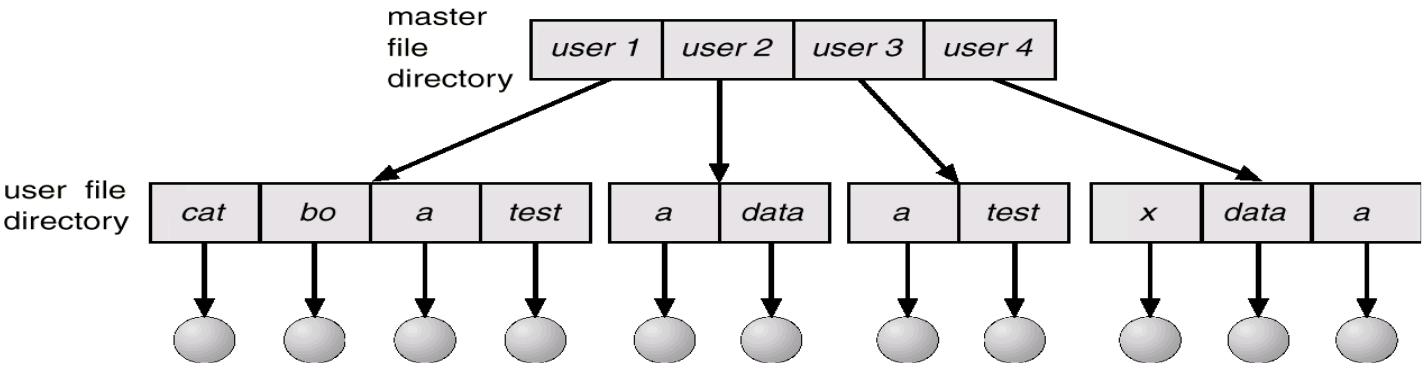
Two level directory:

- In the two level directory structure, each user has his own user file directory(UFD).
- The UFDs have similar structures, but each lists only he files of single user.

- User job starts or a user logs in, the system's mater file directory (MFD) is searched.
- Indexed by user name or account number.
- All the file names within each UFD are unique.
- To delete a file, the operating system confines its search to the local UFD. The program creates a new UFD and adds an entry for it to then MFD.

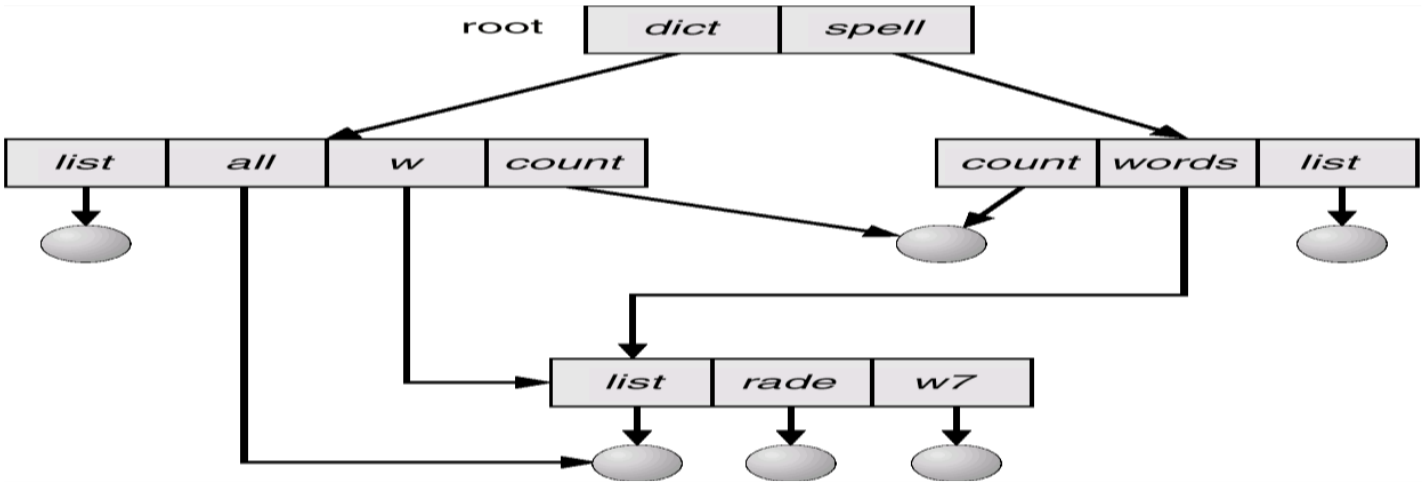
The two level directory structures solves the name collision problem, it still has disadvantages. Effectively isolates one user form another.

- Isolation is an advantage when the users are completely independent.
- A disadvantage when the users want to cooperate on some task and to access one another's files.
- Some systems simply do not allow local user files to be accessed by other users.
- This method is the one most used in UNIX and MS-DOS.



6. Describe about Acyclic Graph Directory (11 Marks) (NOV 13)

- A tree structure prohibits the sharing of files or directories.
- An acyclic graph that is, a graph with no cycles allows directories to share subdirectories and files. The same file or subdirectory may be in tow different directories.
- The acyclic graph is a natural generalization of the tree structured directory scheme.
- It is important to note that a shared file is not the same as two copies of the file.
- With two copies, each programmer can view the copy rather than the original, but if one programmer changes the file, the changes will not appears in the other's copy.
- With a shared file, only one actually file exists, so any changes made by one persons are immediately visible to the other.
- Sharing is particularly important for subdirectories; a new file created by one person will automatically appear in all the shared subdirectories.
- A link is effectively a pointer to another file or subdirectory.

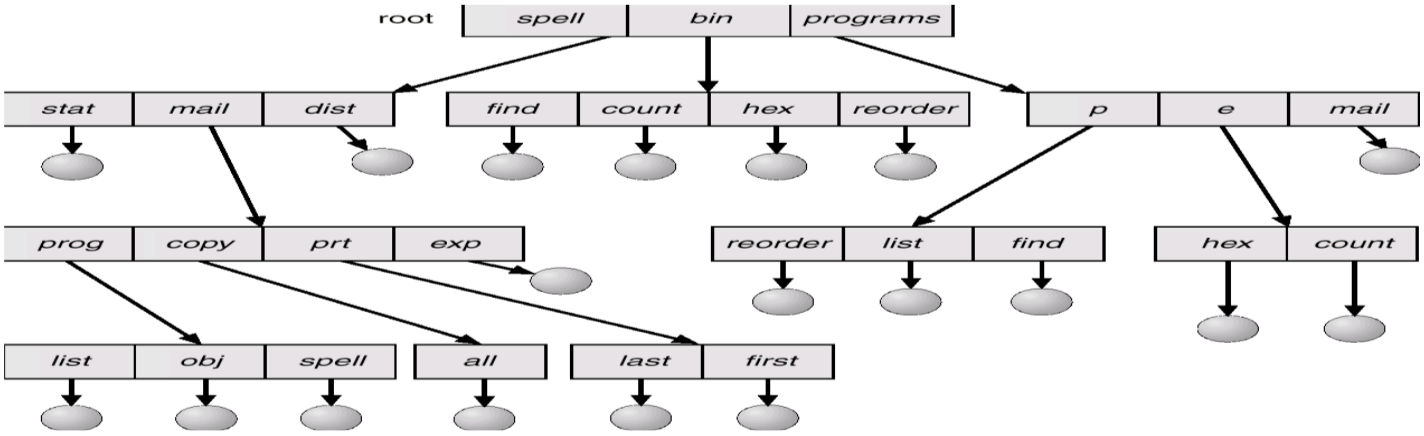


- We resolve the link by using that path name to located the real file.
- Links are easily identified by their format in the directory entry and are effectively named indirect pointers.
- The operating system structure of the system.
- An acyclic graph directory structures is more flexible than is a simple tree structure, but it is also more complex.
- Several problems must be considered carefully. A file may now have multiple absolute path names. Consequently, distinct file names may refer to the same file.
- This situation is similar to the **aliasing problem** for programming languages.

7. Explain Tree structured directories? 5 Marks (APR '14)

- This generalization allows users to create their own subdirectories and to organize their files accordingly.
- To extend the directory structure to a tree of arbitrary height.
- The tree has a root directory, and every file in the system has a unique path name. a directory contains a set of files or subdirectories.
- All directories have the same internal format.
- The current directory should contain most of the files that are of current interest to the process.
- If a file is needed that is not in the current directory, then the user usually must either specify a path name or change the current directory to be the directory holding that file.
- To change directories, a system call is provided that takes a directory name as parameter and uses it to redefine the current directory.
- Path names can be of two types: absolute and relative.
- An absolute path name begins at the root and follows a path down to the specified file, giving the directory names on the path.

- A relative path name defines a path10 form the current directory.
- A user to define her own subdirectories permits her to impose a structure on her files.
- This structure might result in separate directories for files associated with different topics or different forms of information.



8. Write notes about the protection strategies provided for files . (APR'15)

When information is stored in a computer system, we want to keep it safe from physical damage (the issue of *reliability)* and improper access (the issue of *protection).* Reliability is generally provided by duplicate copies of files. Many computers have systems programs that automatically (or through computer-operator intervention) copy disk files to tape at regular intervals (once per day or week or month) to maintain a copy should a file system be accidentally destroyed. File systems can be damaged by hardware problems (such as errors in reading or writing), power surges or failures, head crashes, dirt, temperature extremes, and vandalism. Files may be deleted accidentally. Bugs in the file-system software can also cause file contents to be lost. Reliability .Protection can be provided in many ways. For a small single-user system, protection provided by physically removing the floppy disks and locking them in a desk drawer or file cabinet. In a multiuser system, however, other mechanisms are needed.

Types of Access

The need to protect files is a direct result of the ability to access files. Systems that do not permit access to the files of other users do not need protection. Thus, we could provide complete protection by prohibiting access. Alternatively, we could provide free access with no protection. Both approaches are too extreme for general use. What is needed is Protection mechanisms provide controlled access by limiting the types of file access that can be made. Access is permitted or denied depending on several factors, one of which is the type of access requested. Several different types of operations may be controlled:

Read. Read from the file.

Write. Write or rewrite the file.

Execute. Load the file into memory and execute it.

Append. Write new information at the end of the file.

Delete. Delete the file and free its space for possible reuse.

List. List the name and attributes of the file.

Other operations, such as renaming, copying, and editing the file, may also be controlled. For many systems, however, these higher-level functions may be implemented by a system program that makes lower-level system calls. Protection is provided at only the lower level. For instance, copying a file may be implemented simply by a sequence of read requests. In this case, a user with read access can also cause the file to be copied, printed, and so on.

Many protection mechanisms have been proposed. Each has advantages and disadvantages and must be appropriate for its intended application. A small computer system that is used by only a few members of a research group, for example, may not need the same types of protection as a large corporate computer that is used for research, finance, and personnel operations.

Access control

The most common approach to the protection problem is to make access dependent on the identity of the user. Different users may need different types of access to a file or directory. The most general scheme to implement dependent access is to associate with each file and directory specifying user names and the types of access allowed for each user. When a user requests access to a particular file, the operating system checks the access list associated with that file. If that user is listed for the requested access, the access is allowed. Otherwise, a protection violation occurs, and the user job is denied access to the file.

The access control approach has the advantage of enabling complex access methodologies. The main problem with access lists is their length. If we want to allow everyone to read a file, we must list all users with read access. This technique has two undesirable consequences:

1. Constructing such a list may be a tedious and unrewarding task, especially if we do not know in advance the list of users in the system.
2. The directory entry, previously of fixed size, now must be of variable size, resulting in more complicated space management.

These problems can be resolved by use of a condensed version of the access list. To condense the length of the access-control list, many systems recognize three classifications of users in connection with each file:

Owner. The user who created the file is the owner.

Group. A set of users who are sharing the file and need similar access is a group, or work group.

Universe. All other users in the system constitute the universe.

The most common recent approach is to combine access-control lists with the more general (and easier to implement) owner, group, and universe access control scheme . For example, Solaris 2.6 and beyond use the three categories of access by default but allow access-control lists to be added to specific files and directories when more fine-grained access control is desired.

Other Protection Approaches

Another approach to the protection problem is to associate a password with each file. Just as access to the computer system is often controlled by a password, access to each file can be controlled in the same way. If the passwords are chosen randomly and changed often, this scheme may be effective in limiting access to a file.

9. Consider the following page reference string: (APR '11)

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 following replacement algorithms, assuming one, two, three, four, five, six, or seven frames?

Remember all frames are initially empty, so your first unique pages will all cost one fault each.

LRU replacement

FIFO replacement

Optimal replacement

Answer:

Number of frames	LRU	FIFO	Optimal
1	20	20	20
2	18	18	15
3	15	16	11
4	10	14	08
5	08	10	0 7
6	07	10	07
7	07	0 7	0 7

10. Explain in detail about file sharing? (APR '12)

Sharing of files on multi-user systems is desirable. Sharing may be done through a protection scheme. On distributed systems, files may be shared across an Network. Network File System (NFS) is a

common distributed file sharing method. User IDs identify users, allowing permissions and protections to be per user. Group IDs allow users to be in groups, permitting group access rights.

File Sharing –Remote File Systems

Uses networking to allow file system access between systems in manually via programs like FTP Automatically, seamlessly using distributed file systems Semi automatically via the World Wide Web. Client server: model allows clients to mount remote file systems from servers Server can serve multiple clients Client and user on client identification is insecure or complicated NFS is standard UNIX client server file sharing protocol .CIFS is standard Windows protocol Standard operating system file calls are translated into remote calls Distributed Information Systems (distributed naming services) such as LDAP, DNS, NIS implement unified access to information needed for remote computing.

File Sharing –Failure Modes

Remote file systems add new failure modes, due to network failure, server failure. Recovery from failure can involve state information about status of each remote request. Stateless protocols such as NFS include all information in each request, allowing easy recovery but less security.

File Sharing –Consistency Semantics

Consistency semantics specify how multiple users are to access a shared file simultaneously process synchronization algorithms tend to be less complex due to disk I/O and network latency (for remote file systems Andrew File System (AFS) implemented complex remote file sharing semantics Unix file system (UFS) implements: Writes to an open file visible immediately to other users of the same open file ,Sharing file pointer to allow multiple users to read and write concurrently AFS has session semantics Writes only visible to sessions starting after the file is closed.

Semantics of File Sharing

UNIX semantics: used in centralized systems. UNIX semantics: a read that follows two writes in quick

Succession sees the result of the last write. Semantics of File Sharing is issues in Distributed File Systems. In a Single File Server there is no client caching easy to implement UNIX semantics Client File Caching improves performance by decreasing demand at the server updates to the cached file are not seen by other clients.

Session Semantics: (relaxed semantics) changes to an open file are only visible to the process that modified the file. When the file is closed, changes are visible to other processes closed file is sent

back to the server. Two or more clients are caching and modifying a file a final result depends on who closes last use an arbitrary rule to decide who wins. In this file pointer sharing not possible when a process and its children run on different machines.

No file update semantics :

No File Updates Semantics: files are never updated. Allowed file operations: CREATE and READ files are atomically replaced in the directory.

Transaction Semantics: all file changes are delimited by a Begin and End transaction. All file requests within the transaction are carried out in order. The complete transaction is either carried out completely or not at all (atomicity).

UNIX Semantics every operation is instantly visible to others. Session Semantics no changes visible until file is closed. In No Updates Semantics no file updates are allowed. In Transactions atomic updates for file.

Summary:

UNIX SEMANTICS :every operation is instantly visible to others.

SESSION SEMANTICS :no changes visible until file is closed.

NO FILE UPDATE SEMANTICS :no file updates are allowed.

TRANSACTION SEMANTICS :atomic updates for file is allowed.

11. Write short notes on process creation

Process creation:

Virtual memory provides two techniques to enhance performance of creating and running processes.

- ➔ Copy –on –write
- ➔ Memory mapped files.

Copy on write:

- Demand paging is used when reading a file from disk to memory.
- Process creation using fork() system call may by pass the need for demand paging.
- Alternatively a technique copy – on – write can be used.
- This works by allowing parent and child processes to initially share the same pages.
- These shared pages are marked as copy on writ pages if either process writes to a shared page, copy of the shared page is created.
- Using this technique only the pages that are modified by either process are copied.
- Non modified pages may be shared by parent and child processors.
- It is a common technique used by windows 2000, Linux, Solaris 2.

Memory mapped files:

- Sequential read of a file on disk requires standard system calls open(), read(), write().
- Alternatively virtual memory technique memory mapping can be used.
- This allows a part of the virtual address space to be logically associated with a file.
- Initial access to the file proceeds using ordinary demand paging, resulting in page fault.

- Subsequent reads and writes to the file are handled as routine memory accesses.
- Closing the file result in all the memory mapped data being written back to disk and removed from virtual memory of the process.

Eg: Soloris 2 operating system uses this technique.

12. Explain how many frames can be allocated to processors?

ALLOCATION OF FRAMES:

- The operating system allocates all its buffer and table space from the free – frame list.
- When this space is not in use by the operating system, it can be used to support user paging.
- Three free frames can be reserved on the free- frame list at all the times.
- When a page fault occurs there will be a free frame available.
- Different problem arises when demand paging is combined with multi – programming.
- As multiprogramming puts 2(or more) processes at same time.

Minimum number of frames:

- **As the** number of frames allocated to each process decreased the page fault rate increases, slowing processes execution.
- At least a minimum number is defined by the instruction set architecture.
- The maximum number is defined by the amount of available physical memory.

Allocation algorithms:

- **Equal allocation:** split 'm' frames among n processes
- **Equal share**->m/n frames.
- Left over frames free frame buffer
- **Proportional allocation:**allocate available memory to each processes according to its size.
- $a_i = s_i / s \ *m$
- a_i frames
- s_i size of memory.
- M available frames

Global versus local allocation :

Global replacement:

- Allows process to select a replacement frames from the set of all frames even if that frame is currently allocated to some other process.
- One process can take a frame from another.
- One problem is that a process cannot control its own page fault rate.
- Results in greater system throughput ; a most common method.

Local replacement:

- Requires that each process select from only its own set of allocated frames.
- The number of frames allocated to a process does not change.
- The set of pages in memory for a process is affected by the paging behavior of only that process.

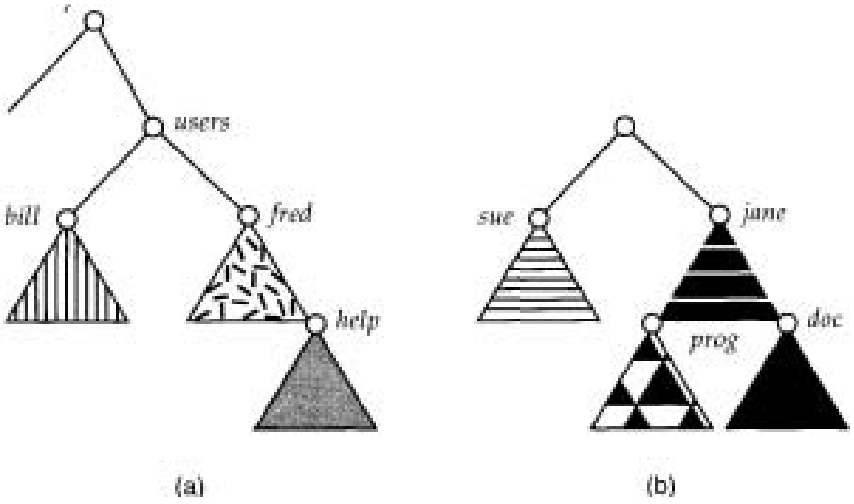
13. Explain about file system mounting

FILE SYSTEM MOUNTING:

- Similar to a file open process before using it a file system must be mounted before it can be available to processes on the system.
- The directory structure can be built out of multiple partitions which must be mounted to make them available within the file system name space.

PROCEDURE:

- **Os is** given the name of the device; location within the files structure to which to attach the file system(mount point).
- A mount point is an empty directory



File system existing Un mounted partition

MOUNT POINT:

- A system may allow the same file system to be mounted repeatedly ; at different mount points or it may only allow one mount per file system.

Pondicherry University Questions

2 Marks

1. What is virtual memory?(APR '11)(APR '12) (Ref.Pg.No.1 Qn.No.1)
2. Compare Swapper with Lazy Swapper?(NOV '13) (Ref.Pg.No.1 Qn.No.4)
3. What is thrashing? (APR 11) (NOV '15) (Ref.Pg.No.3 Qn.No.12)
4 Enumerate Search Path?(UQ NOV '13) (Ref.Pg.No.3 Qn.No.15)
5. List the various File Attributes? (UQ APR '12 &NOV '12) (Ref.Pg.No.4 Qn.No.19)
6. What is System file checker in Windows?(NOV '11) (Ref.Pg.No.7 Qn.No.31)
7. .Give any two criteria to chose a file organization? (APR '12) (Ref.Pg.No.7 Qn.No.32)
8. What is File Authentication?(NOV '11)(APR '11) (Ref.Pg.No.10 Qn.No.44)
9. What is File Lock?(NOV '11) (APR '11) (Ref.Pg.No.9 Qn.No.77)
10.What is File Control Block? (APR '14) (Ref.Pg.No.9 Qn.No.38)
11. What are the advantages and disadvantages of access control? (NOV'14) (Ref.Pg.No.6 Qn.No.25)
12. State the functions of working set strategy model (NOV'14) (Ref.Pg.No.3 Qn.No.14)
13.What are the different accessing methods of a file? (APR'15) (Ref.Pg.No.6 Qn.No.23)
14. What is the principle of optimality? (NOV '15) (Ref.Pg.No.11 Qn.No.47)

11 MARKS

1.What are files and explain the access methods for files? (APR '14) (NOV '15) (Ref.Pg.No.20 Qn.No.4)
2. Write short notes on single and two-level Directory? (APR '11) (Ref.Pg.No.23 Qn.No.5)
3. Describe about Acyclic Graph Directory (NOV 13) (Ref.Pg.No.24 Qn.No.6)
4. Explain Tree structured directories? (APR '14) (Ref.Pg.No.25 Qn.No.7)
5. Explain Thrashing? (APR '11) (APR '14) (NOV '15) (Ref.Pg.No.19 Qn.No.3)
6. Explain in detail about file sharing (APR '12) (Ref.Pg.No.28 Qn.No.10)
7. 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 following replacement algorithms, assuming one, two, three, four, five, six, or seven frames? : (APR '11) (Ref.Pg.No.28 Qn.No.9)

8. Explain page replacement in detail? (APR'15) (NOV '15) (Ref.Pg.No.15 Qn.No.2)
9. Write notes about the protection strategies provided for files. (APR'15) (Ref.Pg.No.26 Qn.No.8)

UNIT – V

File System Structure – File System Implementation – Directory Implementation – Allocation Methods – Free-space Management. Kernel I/O Subsystems - Disk Structure – Disk Scheduling – Disk Management – Swap-Space Management.

Case Study: The Linux System, Windows.

---

2 MARKS

1.Define Bootstrap Program?(NOV '12)

Bootstrapping usually refers to the starting of a self-sustaining process that is supposed to proceed without external input. In computer technology the term (usually shortened to booting) usually refers to the process of loading the basic software into the memory of a computer after power-on or general reset, especially the operating system which will then take care of loading other software as needed.

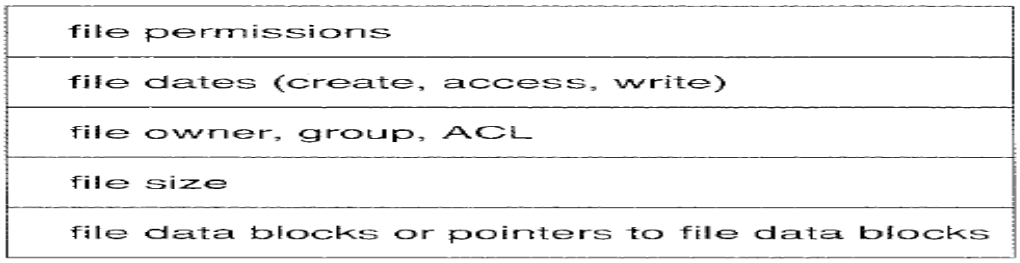
2.What is boot control block

A boot control block (per volume) can contain information needed by the system to boot an operating system from that volume. If the disk does not contain an operating system, this block can be empty. It is typically the first block of a volume. In UFS, it is called the boot block: in NTFS, it is the boot block in NTFS ,it is the partition block sector .

3.What is volume control block

A volume control block (per volume) contains volume (or partition) details, such as the number of blocks in the partition, the size of the blocks, a free-block count and free-block pointers, and a free-FCB count and FCB pointers. In UFS, this is called a superblock in NTFS, it is stored in the master file table

4.Draw file control block



5.List the methods to implement directory

Linked list

The simplest method of implementing a directory is to use a linear list of file names with pointers to the data blocks. This method is simple to program but time-consuming to execute.

Hash table

In this method, a linear list stores the directory entries, but a hash data structure is also used. The hash table takes a value computed from the file name and returns a pointer to the file name in the linear list. Therefore, it can greatly decrease the directory search time.

6.Write the types of directory allocation methods?

Three major methods of allocating disk space are

1. Contiguous allocation.
2. linked allocation.
3. indexed allocation.

7.What is Contiguous Allocation?

A single contiguous set of blocks is allocated to a file at the time of file creation; this is pre allocation strategy that uses portions of variable size. The FAT (File Allocation Table) needs just a single entry for each file, showing the starting block and the length of the file. Disk addresses define a linear ordering on the disk.

8.What is the drawback of contiguous allocation? How to overcome from it?

It suffers from external fragmentation. Compaction is used to solve the problem of external fragmentation. The original disk uses then freed completely by creating one large contiguous free space.Second problem with contiguous allocation algorithm is that with pre allocation. It is necessary to declare the size of the file at the time of creation.

9.What is linked allocation?

- Linked allocation solves all problems of contiguous allocation.
- Each block contains a pointer to the next block in the chain. The disk blocks may be scattered anywhere on the disk, The FAT contains a pointer to the first and last blocks of the file.
- To create a new file, simply create a new entry in FAT .With linked allocation each directory entry has a pointer to the first disk block of the file.

10.Write the disadvantages of Linked allocation

- The major problem is that it can be used effectively only for sequential access files. To find the ith block of a file, we must start at the beginning of that file and follow the pointers until we get to the ith block.

11.What is indexed allocation

- The FAT contains a separate one level index for each file, the index has one entry foreach portion allocated to the file.
- The ith entry in the index block points to the ith block of the file.
- The directory contains the address of the index block.

12. Write about free space management

Since disk space is limited, we need to reuse the space from deleted files for new files , if possible. (Write-once optical disks only allow one write to any given sector, and thus such reuse is not physically possible.) To keep track of free disk space, the system maintains a free-space list. The free-space list records all *free* disk blocks-those not allocated to some file or directory.

13. How free list can be implemented

1. Bit vector
2. Linked List
3. Grouping
4. Counting

14. Write about Bit vector

Frequently, the free-space list is implemented as a bit vector or bit map Each block is represented by 1 bit. If the block is free, the bit is 1; if the block is allocated, the bit is 0.

15. Write about Linked List in free space management

Another approach to free-space management is to link together all the free disk blocks, keeping a pointer to the first free block in a special location on the disk and caching it in memory. This first block contains a pointer to the next free disk block, and so on.

16. Write about Grouping in free space management

A modification of the free-list approach stores the addresses of *n* free blocks in the first free block. The first n-1 of these blocks are actually free. The last block contains the addresses of another *n* free blocks, and so on. The addresses of a large number of free blocks can now be found quickly, unlike the situation when the standard linked-list approach is used.

17. Write about Counting in free space management

Rather than keeping a list of *n* free disk addresses, we can keep the address of the first free block and the number (n) of free contiguous blocks that follow the first block. Each entry in the free-space list then consists of a disk address and a count. Although each entry requires more space than would a simple disk address, the overall list is shorter, as long as the count is generally greater than 1.

18. What do you mean by seek time? (APR '12)

The total amount of time required for information on a disk drive to be found. The lower this value is the faster the hard drive will be able to find or read data. Examples of common hard drive **seeks times** are 8ms and 10ms.

19. List the disk performance parameter?(APR '13)

1. Seek Time: Seek time is the time required to move the disk arm to the required track.

2. Rotational Delay: Disk drive generally rotates at 3600 rpm, ie to make one revolution it takes around 16.7 ms. Thus on the average, the rotational delay will be 8.3 ms.

3. Transfer Time: The transfer time to or from the disk depends on the rotational speed of the disk

20. What is the purpose of device drivers? (APR '13)

A device driver is a file that lets the computer know the configuration and specifications of a certain hardware device. Some examples of devices that need drivers are hard drives, DVD drives, and PCI cards. Without the driver file, the computer will be unable to communicate with the device. Often, Windows will alert the user if there is a new device which doesn't have a driver installed yet. It will then hopefully let you install the correct driver for it.

21. List Different types of scheduling algorithm are as follows:

1. First come first serviced scheduling algorithm

2. Shortest seek time first algorithm

3. SCAN
4. Seek (or LOOK)
5. Circular scans(C-SCAN)
6. C-Seek (or C-LOOK)

22. Write about FCFS Scheduling

- FCFS-FIRST COME FIRST SERVE is the simplest form of disk scheduling.
- Service requests in the order they are received. This algorithm is similar to the FCFS job scheduling algorithm.
- FCFS is easy to implement but does not guarantee good throughput.

23.Write about SSTF Scheduling

- This algorithm selects the request with the minimum seek time from the current head position.
- SSTF may tend to move the head away from some requests during its local minimization.
- Seek time increases with the number of cylinders traversed by the head,SSTF chooses the pending request closest to the current head position so always choosing the minimum seek time does not guarantee that the average seek time over a number of arm movements will be minimum .This choice should provide better performance than FCFS algorithm.

24.Write about Scan Scheduling:

- In this type of algorithm, the disk can starts at one end of the disk and moves towards the other end.
- At the other end the direction of head movement is reversed and servicing continues.
- This algorithm is also otherwise called as elevator algorithm. before applying this algorithm, we should know the direction of head movement, in addition to the head current position.

25.Write about C-Scan Scheduling:

- C-SCAN SCHEDULING (CIRCULAR SCAN) is a variant of scan designed to provide a more uniform wait time.
  - When the head reaches the other end, however it immediately returns to the beginning of the disk, without servicing any request on the return trip.

26.Write about LOOK scheduling:

- The versions of C-SCAN and SCAN scheduling algorithm are called C-LOOK and LOOK scheduling.
- Both the algorithms move the disk across its full width. This is because they look for a request before continuing to more in a given direction.

27.Write about C-LOOK scheduling:

- This is a modification of seek algorithm ,similar to C-SCAN .Here the arm goes only as far as the requests to no serviced and requests are serviced only in one direction.
- It is the best algorithm, since it involves minimum movement of the disk head to service all the requests. Disk head will not move, if there is no pending requests thus reducing wear and tear of the disk.

28. Write about Linux 1.0 ?

Linux 1.0 (March 1994) included these new features:

- Support for UNIX's standard TCP/IP networking protocols
- BSD-compatible socket interface for networking programming
- Device-driver support for running IP over an Ethernet
- Enhanced file system
- Support for a range of SCSI controllers for high-performance disk access
- Extra hardware support

- Version 1.2 (March 1995) was the final PC-only Linux kernel

29. Write about Linux 2.0?

Linux 2.0

- Released in June 1996, 2.0 added two major new capabilities:
  - Support for multiple architectures, including a fully 64-bit native Alpha port. It Supports for multiprocessor architectures
- Other new features included:
  - Improved memory-management code
  - Improved TCP/IP performance
  - Support for internal kernel threads, for handling dependencies between loadable modules, and for automatic loading of modules on demand
  - Standardized configuration interface

- Available for Motorola 68000-series processors, Sun Spark systems, and for PC and PowerMac systems
- 2.4 and 2.6 increased SMP support, added journaling file system, preemptive kernel, 64-bit memory support

30. Define Slack ware?

The first Linux distributions managed these packages by simply providing a means of unpacking all the files into the appropriate places; modern distributions include advanced package management Early distributions included SLS and **Slack ware .Slack ware** represents overall improvement in quality. ***Red Hat*** **and** ***Debian*** are popular distributions from commercial and noncommercial sources, respectively.

31. Write the Kernel modules are available in Linux?

- Sections of kernel code that can be compiled, loaded, and unloaded independent of the rest of the kernel
- A kernel module may typically implement a device driver, a file system, or a networking protocol
- The module interface allows third parties to write and distribute, on their own terms, device drivers or file systems that could not be distributed under the GPL
- Kernel modules allow a Linux system to be set up with a standard, minimal kernel, without any extra device drivers built in
- Three components to Linux module support:
  - **Module management** –allows modules to be loaded into memory.
  - **Driver registration**-allows modules to tell the rest of the kernel that a new driver has become available.
  - **Conflict resolution**-allows different device drivers to reserve hardware resources and to protect those resources from accidental use by another driver.

32. Define Kernel in OS? (APR '12) (APR'15) (NOV '15)

A *kernel* is the part of the operating system that mediates access to system resources. It's responsible for enabling multiple applications to effectively share the hardware by controlling access to CPU, memory, disk I/O, and networking.

An *operating system* is the kernel plus applications that enable users to get something done (i.e compiler, text editor, window manager, etc).

33. Define Module Management?

Supports loading modules into memory and letting them talk to the rest of the kernel. Module loading is split into two separate sections:

1. Managing sections of module code in kernel memory
2. Handling symbols that modules are allowed to reference

The module requestor manages loading requested, but currently unloaded, modules; it also regularly queries the kernel to see whether a dynamically loaded module is still in use, and will unload it when it is no longer actively needed

34. Define Driver Registration?

Allows modules to tell the rest of the kernel that a new driver has become available The kernel maintains dynamic tables of all known drivers, and provides a set of routines to allow drivers to be added to or removed from these tables at any time. Registration tables include the following items:

1. **Device drivers**-These drivers include character devices (such as printers, terminals) ,block devices(including all disk drivers), and network interface devices.
2. **File systems** –It implements Linux's virtual –file –system calling routines.
3. **Network protocols**-It implements the entire networking protocol, such as IPX, or a new set of packet-filtering rules for a network firewall.
4. **Binary format**-Specifies a way of recognizing , and loading ,a new type of executable file.

35. Define Conflict Resolution?

A mechanism that allows different device drivers to reserve hardware resources and to protect those resources from accidental use by another driver.

The conflict resolution module aims to:

- Prevent modules from clashing over access to hardware resources
- Prevent ***auto probes*** from interfering with existing device drivers
- Resolve conflicts with multiple drivers trying to access the same hardware

36. Write the contents of Registration tables?

Registration tables include the following items:

1. **Device drivers**-These drivers include character devices (such as printers, terminals) ,block devices(including all disk drivers), and network interface devices.

2. **File systems** –It implements Linux's virtual –file –system calling routines.
3. **Network protocols**-It implements the entire networking protocol, such as IPX, or a new set of packet-filtering rules for a network firewall.
4. **Binary format**-Specifies a way of recognizing , and loading ,a new type of executable file.

37. What are the operations are available in process management?

UNIX process management separates the creation of processes and the running of a new program into two distinct operations.

1. The fork system call creates a new process
2. A new program is run after a call to execve

Under UNIX, a process encompasses all the information that the operating system must maintain *t* track the context of a single execution of a single program. Under Linux, process properties fall into three groups: the process's identity, environment, and context.

38. Define Scheduling Context?

The **scheduling context** is the most important part of the process context; it is the information that the scheduler needs to suspend and restart the process. The kernel maintains **accounting** information about the resources currently being consumed by each process, and the total resources consumed by the process in its lifetime so far.

39..Define signal-Handler table?

The **signal-handler table** defines the routine in the process's address space to be called when specific signals arrive. The **virtual-memory context** of a process describes the full contents of the its private address space.

40. What is IPC? (APR '12)

Interprocess communication (IPC) is a set of programming interfaces that allow a programmer to coordinate activities among different program processes that can run concurrently in an operating system. This allows a program to handle many user requests at the same time. Since even a single user request may result in multiple processes running in the operating system on the user's behalf, the processes need to communicate with each other. The IPC interfaces make this possible. Each IPC method has its own advantages and limitations so it is not unusual for a single program to use all of the IPC methods.

41. Define file table?

The **file table** is an array of pointers to kernel file structures. When making file I/O system calls, processes refer to files by their index into this table .Whereas the file table lists the existing open files, the file**-system context** applies to requests to open new files. The current root and default directories to be used for new file searches are stored here.

42.Define Virtual Memory context?

The **virtual-memory context** of a process describes the full contents of the its private address space.

43. Write any two advantages of Linux?(NOV '12)

1. **Open Source:** If you develop software that requires knowledge or modification of the operating system code, Linux's source code is at your fingertips. Most Linux applications are Open Source as well.
2. **Fast and easy installation:** Most Linux distributions come with user-friendly installation and setup programs. Popular Linux distributions come with tools that make installation of additional software very user friendly as well.
3. **Compatibility**: It runs all common Unix software packages and can process all common file formats.

44. Define static Linking?

A program whose necessary library functions are embedded directly in the program's executable binary file is ***statically*** linked to its libraries. The main disadvantage of static linkage is that every program generated must contain copies of exactly the same common system library functions.

45.Define Dynamic Linking?

Dynamic linking is more efficient in terms of both physical memory and disk-space usage because it loads the system libraries into memory only once.

46. Define Interprocess Communication?

Like UNIX, Linux informs processes that an event has occurred via signals. There is a limited number of signals, and they cannot carry information: Only the fact that a signal occurred is available to a process. The Linux kernel does not use signals to communicate with processes with

are running in kernel mode, rather, communication within the kernel is accomplished via scheduling states and wait ,queue structures.

47. Write any two design principles of Linux?

1. Linux is a multi-user, multitasking system with a full set of UNIX-compatible tools
2. Its file system adheres to traditional UNIX semantics, and it fully implements the standard UNIX networking model
3. Main design goals are speed, efficiency, and standardization
4. Linux is designed to be compliant with the relevant POSIX documents; at least two Linux distributions have achieved official POSIX certification

- The Linux programming interface adheres to the SVR4 UNIX semantics, rather than to BSD behavior

48. What is Process Identity?

Process ID (PID)- The unique identifier for the process; used to specify processes to the operating system when an application makes a system call to signal, modify, or wait for another process .

Credentials- Each process must have an associated user ID and one or more group IDs that determine the process's rights to access system resources and files

Personality- Not traditionally found on UNIX systems, but under Linux each process has an associated personality identifier that can slightly modify the semantics of certain system calls

Used primarily by emulation libraries to request that system calls be compatible with certain specific flavors of UNIX

49. Brief about Process Environment Block (PEB).(NOV '13)

When Windows Kernel loads a 64bit executable image in memory it builds automatically a PEB for it. The PEB (Process Environment Block) is process specific area of user land memory that

50.Write about fork() and clone() system call?

1. **fork** creates a new process with its own entirely new process context
2. **clone** creates a new process with its own identity, but that is allowed to share the data structures of its parent.

Using **clone** gives an application fine-grained control over exactly what is shared between two threads.

51. Does Linux supports Symmetric Multiprocessing?

Linux 2.0 was the first Linux kernel to support SMP hardware; separate processes or threads can execute in parallel on separate processors. To preserve the kernel's nonpreemptible synchronization requirements, SMP imposes the restriction, via a single kernel spin lock, that only one processor at a time may execute kernel-mode code

52. List the role of process manager?(NOV '13)

The Process Manager is responsible for creating new processes in the system and managing the most fundamental resources associated with a process. These services are all provided via messages. For example, if a running process wants to create a new process, it does so by sending a message containing the details of the new process to be created. Note that since messages are network-wide, you can easily create a process on another node by sending the process-creation message to the Process Manager on that node.

53. Write about Character Devices

A device driver which does not offer random access to fixed blocks of data. A character device driver must register a set of functions which implement the driver's various file I/O operations. The kernel performs almost no preprocessing of a file read or write request to a character device, but simply passes on the request to the device. The main exception to this rule is the special subset of character device drivers which implement terminal devices, for which the kernel maintains a standard interface

54.Write about few design principles of windows XP

1. Extensibility — layered architecture
2. Portability —XP can be moved from on hardware architecture to another with relatively few changes
3. Reliability —XP uses hardware protection for virtual memory, and software protection mechanisms for operating system resources
4. Compatibility — applications that follow the IEEE 1003.1 (POSIX) standard can be complied to run on XP without changing the source code

- Performance —XP subsystems can communicate with one another via high-performance message passing.

55.Write about XP Architecture?

Layered system of modules

Protected mode — HAL, kernel, executive

User mode — collection of subsystems

Environmental subsystems emulate different operating systems

Protection subsystems provide security functions

56.write about system components of XP?

Foundation for the executive and the subsystems

Never paged out of memory; execution is never preempted

Four main responsibilities: thread scheduling, interrupt and exception handling ,low-level processor synchronization, recovery after a power failure.

Kernel is object-oriented, uses two sets of objects

Dispatcher objects :control dispatching and synchronization (events, mutants, mutexes, semaphores, threads and timers)

Control objects :(asynchronous procedure calls, interrupts, power notify, power status, process and profile objects)

57.Write about Kernel and Process and Threads?

The process has a virtual memory address space, information (such as a base priority), and an affinity for one or more processors. Threads are the unit of execution scheduled by the kernel's dispatcher. Each thread has its own state, including a priority, processor affinity, and accounting information .A thread can be one of six states: ready, standby, running, waiting, transition, and terminated.

58.What is Kernel Scheduling

The dispatcher uses a 32-level priority scheme to determine the order of thread execution Priorities are divided into two classes

1. The real-time class contains threads with priorities ranging from 16 to 31
2. The variable class contains threads having priorities from 0 to 15

59.What is soft real time

Real-time threads are given preferential access to the CPU; but XP does not guarantee that a real-time thread will start to execute within any particular time limit This is known as *soft real-time*

60.write about i/ o subsystem

The I/O manager is responsible for file systems,cache management ,device drivers,network drivers,Keeps track of which installable file systems are loaded, and manages buffers for I/O requests Works with VM Manager to provide memory-mapped file I/O Controls the XP cache manager, which handles caching for the entire I/O system Supports both synchronous and asynchronous operations, provides time outs for drivers, and has mechanisms for one driver to call another

61.What type of security is provided by Windows XP (APR'15)

The object-oriented nature of XP enables the use of a uniform mechanism to perform runtime access validation and audit checks for every entity in the system Whenever a process opens a handle to an object, the security reference monitor checks the process's security token and the object's access control list to see whether the process has the necessary rights. Security of an NTFS volume is derived from the XP object model .Each file object has a security descriptor attribute stored in this MFT record .This attribute contains the access token of the owner of the file, and an access control list that states the access privileges that are granted to each user that has access to the file

62. Write about file compression in XP

To compress a file, NTFS divides the file's data into *compression units*, which are blocks of 16 contiguous clusters. For sparse files, NTFS uses another technique to save space Clusters that contain all zeros are not actually allocated or stored on disk .Instead, gaps are left in the sequence of virtual cluster numbers stored in the MFT entry for the file. When reading a file, if a gap in the virtual cluster numbers is found, NTFS just zero-fills that portion of the caller's buffer

63.write about memory management

A heap in the Win32 environment is a region of reserved address space

A Win 32 process is created with a 1 MB *default heap*

Access is synchronized to protect the heap's space allocation data structures from damage by concurrent updates by multiple threads

Because functions that rely on global or static data typically fail to work properly in a multithreaded environment, the thread-local storage mechanism allocates global storage on a per-thread basis. The mechanism provides both dynamic and static methods of creating thread-local storage

64.Write about interprocess communication in XP

Win32 applications can have interprocess communication by sharing kernel objects An alternate means of interprocess communications is message passing, which is particularly popular for Windows GUI applications

One thread sends a message to another thread or to a window

A thread can also send data with the message

Every Win32 thread has its own input queue from which the thread receives messages This is more reliable than the shared input queue of 16-bit windows, because with separate queues, ne stuck application cannot block input to the other applications

65. What type of operating system is Windows XP? Describe two of its major features.

A 32/64 bit preemptive multitasking operating system supporting multiple users.

i. The ability automatically to repair application and operating system problems.

ii. Better networking and device experience (including digital photography and video).

66. List the design goals of Windows XP. Describe two in detail.

Design goals include security, reliability, Windows and POSIX application compatibility, high performance, extensibility, portability and international support.

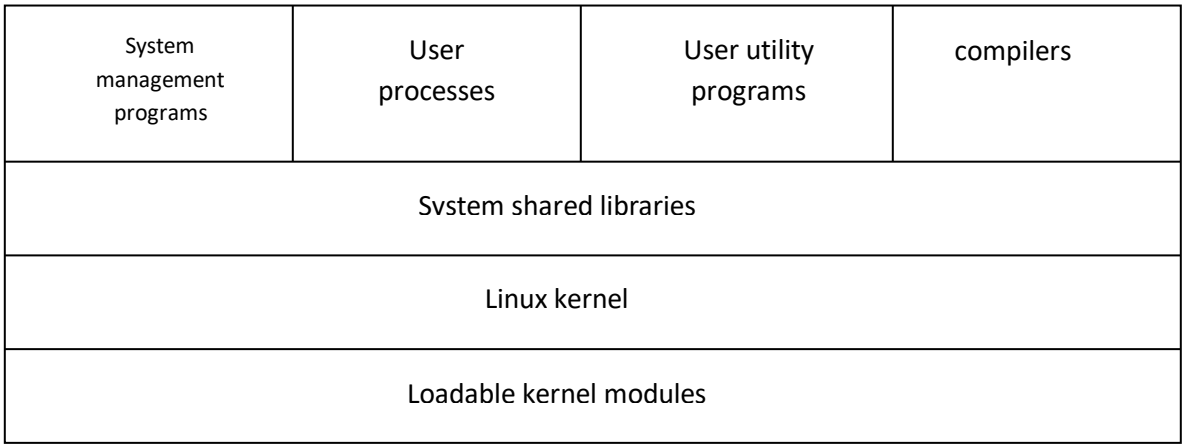
i. Reliability was perceived as a stringent requirement and included extensive driver verification, facilities for catching programming errors in user-level code, and a rigorous certification process for third-party drivers, applications, and devices.

ii. Achieving high performance required examination of past problem areas such as I/O performance, server CPU bottlenecks, and the scalability of multithreaded and multiprocessor environments.

67. What are the responsibilities of the I/O manager?

The I/O manager is responsible for file systems, device drivers, and network drivers. The I/O manager keeps track of which device drivers, filter drivers, and file systems are loaded and manages buffers for I/O requests. It furthermore assists in providing memory mapped file I/O and controls the cache manager for the whole I/O system.

68.Sketch the components of a Linux system (NOV'14)



69. Define physical record and logical record. (NOV '15)

A physical record often is unstructured and has a fixed size related to the kind of physical media that stores it, and possibly to the location of the record on the media.

A logical record often is structured (has various program-specific fields) and might be stored in some number of full or partial physical records.

For example, a hard disk drive might be divided up into sectors (physical records) of 512 bytes each. If an application needs 1200 bytes of data for an address book entry (logical record) this will occupy portions of 3-4 sectors on disk.

11 MARKS

1. Explain the allocation methods for disk space in file allocation? (11) (NOV 12, 13) (APR '11)(APR'15)

Allocation methods:

Three major methods of allocating disk space are

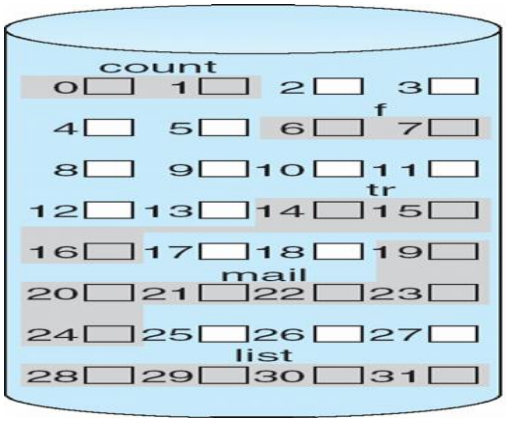
1. Contiguous allocation.
2. linked allocation.
3. indexed allocation.

Contiguous allocation

A single contiguous set of blocks is allocated to a file at the time of file creation; this is pre allocation strategy that uses portions of variable size. The FAT (File Allocation Table) needs just a single entry for each file, showing the starting block and the length of the file. Disk addresses define a linear ordering on the disk.

It suffers from external fragmentation. Compaction is used to solve the problem of external fragmentation.The original disk uses then freed completely by creating one large contiguous free space.

Second problem with contiguous allocation algorithm is that with pre allocation. It is necessary to declare the size of the file at the time of creation.



directory

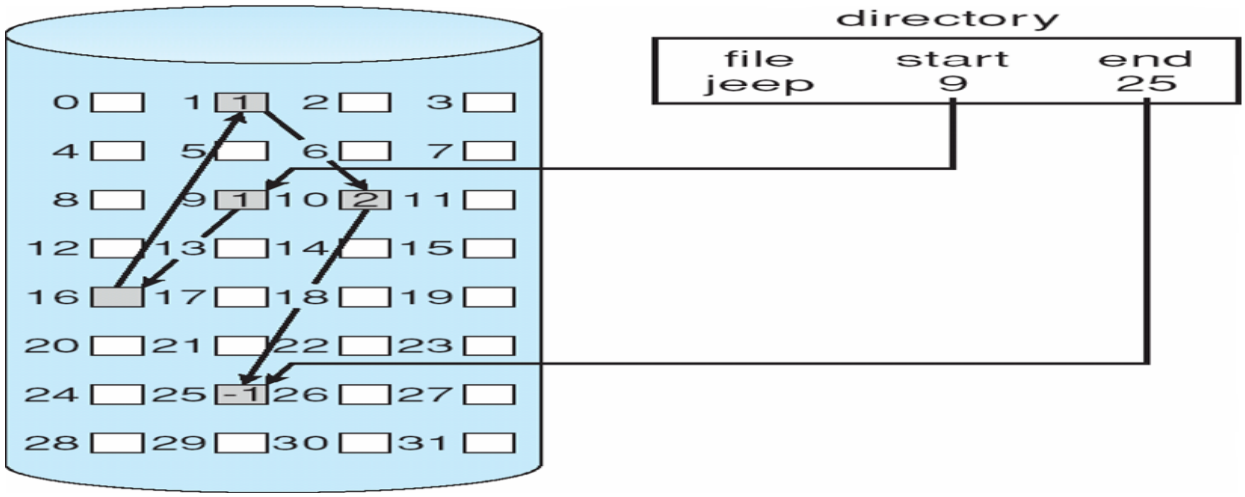
file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Disadvantage:

- Dynamic allocation is not possible.
- The contiguous disk space allocation problem can be seen to be a particular application of the general dynamic storage allocation.
- Any management system can be used but some are slower than others.

- The other problem is external fragmentation because when it exist free space is broken into chunks.

Linked allocation:



- Linked allocation solves all problems of contiguous allocation.
- Each block contains a pointer to the next block in the chain. The disk blocks may be scattered anywhere on the disk, The FAT contains a pointer to the first and last blocks of the file.
- To create a new file, simply create a new entry in FAT .With linked allocation each directory entry has a pointer to the first disk block of the file.
- This pointer is initialized to nil to signify an empty file. Size field is also set to zero. There is no external fragmentation.
- The size of the file does not need to be declared when that file is created. A file can continue to grow as long as free blocks are available. It is never necessary to compact disk space.

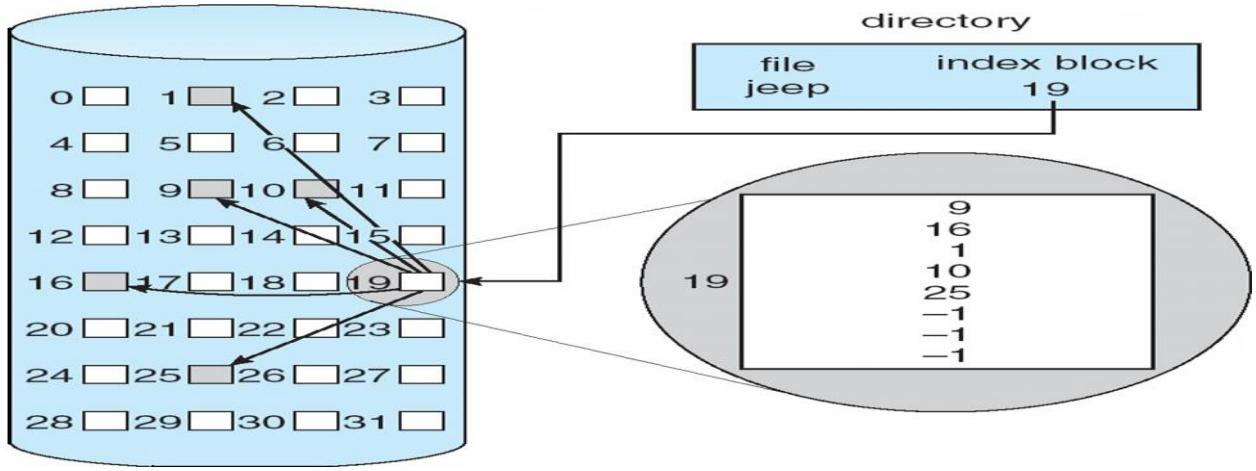
Disadvantage:

The major problem is that it can be used effectively only for sequential access files. To find the ith block of a file, we must start at the beginning of that file and follow the pointers until we get to the ith block.

- Each access to a pointer requires a disk read and sometimes a disk seek
- Space is required for pointers.
- Pointers are used much smaller percentage of the file disk space.
- Other problem in linked allocation is reliability since the files are linked together by pointer scattered all over the disk.
- An important variation on linked allocation method is the use of a file allocation table(FAT).
- FAT is simple but efficient method of disk space allocation is used by MS-DOS and OS /2 operating systems.
- FAT is used same way as linked list.

- FAT allocation scheme can result in a significant number of disk head seeks, unless the FAT is cached.
- The disk head must move to the start of the volume to read the FAT.

Indexed allocation:



- The FAT contains a separate one level index for each file, the index has one entry foreach portion allocated to the file.
- The ith entry in the index block points to the ith block of the file.
- The directory contains the address of the index block.
- Typically the file indexes are not physically stored as part of the FAT .Rater the file index for a file is kept in a separate block and the entry for the file in the FAT points to that block.
- Allocation may be on the basis of either fixed size block or variable size potions.
- When the file is created; all pointers in the index block are set to nil.
- When i^{th} block is first written, a block is obtained from the free space manager and its address is put in the i^{th} index block entry.
- Allocation by blocks eliminates external fragmentation, where as allocation by variable size portions improves locality.
- Indexed allocation supports direct access and sequential access to the file and it is the most popular form of file allocation.
- Indexed allocation does not suffer from wasted space. The pointer overhead of the index block is generally greater than the pointer overhead of linked allocation.

2. Explain Disk Scheduling in detail with example? (APR '11) (APR '13) (NOV '15)

Disk Scheduling the amount of head movement needs to satisfy a series of I/O request can affect the performance. If the desired disk drive and controller are available; the request can be

serviced immediately. If a device or controller is busy; any new requests for service will be placed on the queue of pending requests for that drive.

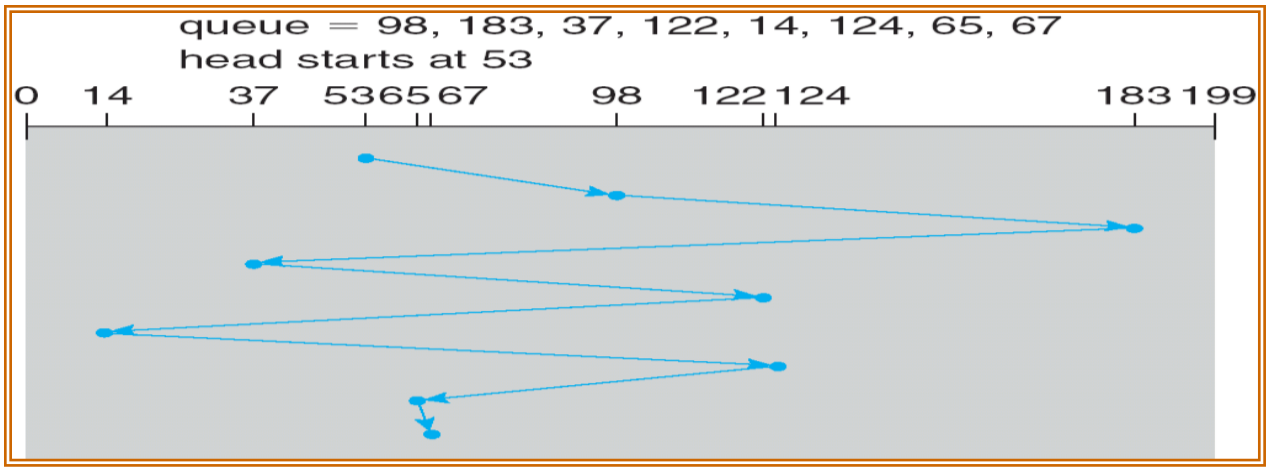
When one request is completed; the OS chooses which pending request to service next.

Different types of scheduling algorithm are as follows:

1. First come first serviced scheduling algorithm
2. Shortest seek time first algorithm
3. SCAN
4. Seek (or LOOK)
5. Circular scans(C-SCAN)
6. C-Seek (or C-LOOK)

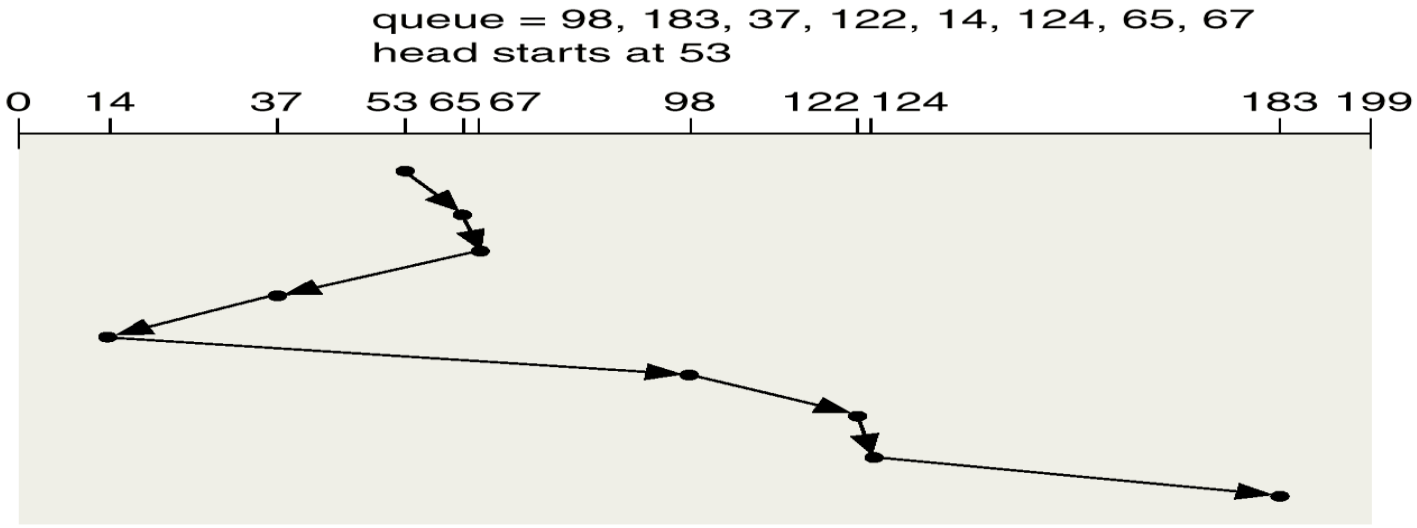
FCFS Scheduling:

- FCFS-FIRST COME FIRST SERVE is the simplest form of disk scheduling.
- Service requests in the order they are received. This algorithm is similar to the FCFS job scheduling algorithm.
- FCFS is easy to implement but does not guarantee good throughput.
- It does not provide the fastest service.
- It takes no special action to minimize the overall seek time.
- This can be explained with an example as follows:
- Consider disk queue with requests for I/O to blocks in cylinders are, 98, 183,37,122,14,124,65,67.
- Here if the disk head is initially at 53, then it will move from 53 to 98 and then to this happens for a total head movements of 640 cylinders.
- Here the total head movements can be decreased simultaneously. suppose for cylinders 37 &14 has to be serviced together, before or after the requests at 122 & 124,then the head movements can be decreased.



SSTF Scheduling:

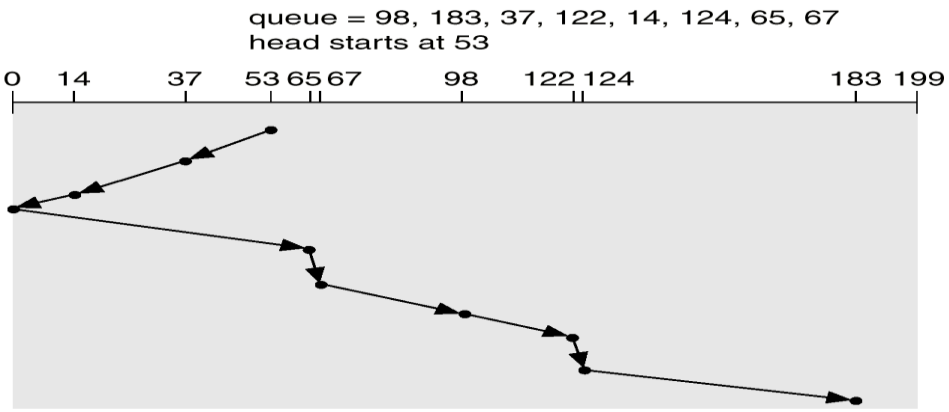
- This algorithm selects the request with the minimum seek time from the current head position.
- SSTF may tend to move the head away from some requests during its local minimization.
- Seek time increases with the number of cylinders traversed by the head,SSTF chooses the pending request closest to the current head position so always choosing the minimum seek time does not guarantee that the average seek time over a number of arm movements will be minimum .This choice should provide better performance than FCFS algorithm.



- From the above FCFS example, the closest request to the initial head position 53 is at cylinder 65.once we are at cylinder 65,the next closest request is at cylinder 67.from these, the request at cylinder 37 is closer than 98,so 37 is served next then finally we will reach 183.
- This scheduling method results in a total head movement of only 236 cylinders little more than more one-third of the distance needed for FCFS scheduling of this request queue.

Scan Scheduling:

- In this type of algorithm, the disk can starts at one end of the disk and moves towards the other end.
- At the other end the direction of head movement is reversed and servicing continues.
- This algorithm is also otherwise called as elevator algorithm. before applying this algorithm, we should know the direction of head movement, in addition to the head current position.
- If the disk arm is moving towards 0, then the head will move towards the other end of the disk servicing the request at 65, 67, 98,122,124 &183.first we have to service all
- The request goingup,and then reversing it.



C-Scan Scheduling:

- C-SCAN SCHEDULING (CIRCULAR SCAN) is a variant of scan designed to provide a more uniform wait time.
  - When the head reaches the other end, however it immediately returns to the beginning of the disk, without servicing any request on the return trip.
  - It essentially treats the cylinder as a circular list.

LOOK scheduling:

- The versions of C-SCAN and SCAN scheduling algorithm are called C-LOOK and LOOK scheduling.
- Both the algorithms move the disk across its full width. This is because they look for a request before continuing to more in a given direction.

C-LOOK scheduling:

- This is a modification of seek algorithm ,similar to C-SCAN .Here the arm goes only as far as the requests to no serviced and requests are serviced only in one direction.
- It is the best algorithm, since it involves minimum movement of the disk head to service all the requests. Disk head will not move, if there is no pending requests thus reducing wear and tear of the disk.

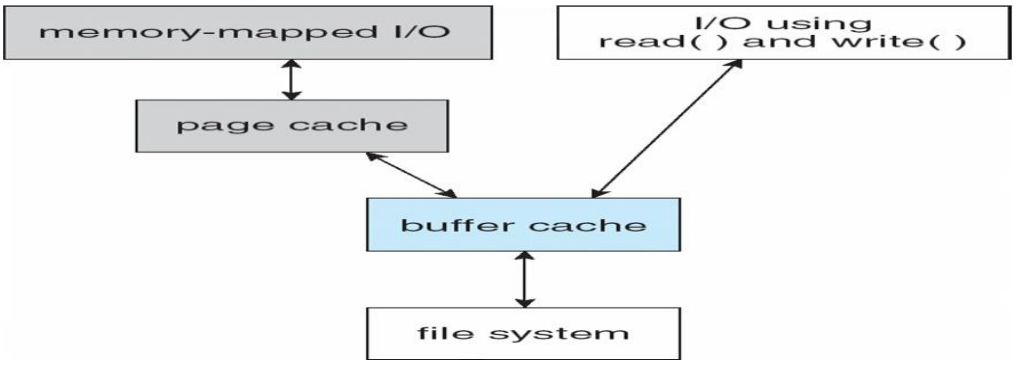
3. Discuss the various techniques used to improve the efficiency and performance of secondary storage? 6 Marks (APR '14)

Efficiency dependent on:

- Disk allocation and directory algorithms
- Types of data kept in file's directory entry
- Pre-allocation or as-needed allocation of metadata structures
- Fixed-size or varying-size data structures

In a Performance is Keeping data and metadata close together and Buffer cache – separate section of main memory for frequently used blocks ,Synchronous writes sometimes requested by apps or needed by OS, No buffering / caching – writes must hit disk before acknowledgement ,Asynchronous writes more common, buffer-able, faster Free-behind and read-ahead – techniques to optimize sequential access ,Reads frequently slower than writes.

A **page cache** caches pages rather than disk blocks using virtual memory techniques and addresses. Memory-mapped I/O uses a page cache. Routine I/O through the file system uses the buffer (disk) cache. This leads to the following figure



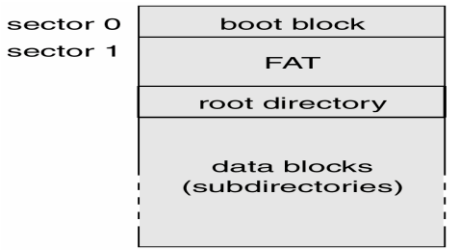
In an the above figure A **unified buffer cache** uses the same page cache to cache both memory-mapped pages and ordinary file system I/O to avoid **double caching. Consistency checking** is used to compares data in directory structure with data blocks on disk, and tries to fix inconsistencies. Can be slow and sometimes fails. Use system programs to **back up** data from disk to another storage device (magnetic tape, other magnetic disk, optical).Recover lost file or disk by **restoring** data from backup.

4. Briefly explain Disk management and swap space management? (NOV '12)

Low level formatting or physical formatting: Dividing a disk into sectors that the disk controller can read and write. To use a disk to hold files, the operating system still needs to record its own data structures on the disk. Partition the disk into one or more groups of cylinders. Logical

formatting or "making a file system". Boot block initializes system. The bootstrap is stored in ROM.Bootstrap loader program. Methods such as sector sparing used to handle bad blocks.

MS-DOS Disk Layout



Methods such as sector sparing (also known as forwarding) used to handle bad blocks. Spare sectors set aside on low-level formatting – Controller told to replace a bad sector logically with one of the spare sectors – To retain effectiveness of disk-scheduling optimization, provide spare sectors in each cylinder and also provides some spare cylinders. Use spare sector from same cylinder if possible.

Sector slipping is moves blocks following bad block downward (occupying spare sector) to free up block following bad Block skips bad block, using freed up block to hold that sector's information.

Swap-Space Management

Swap space Virtual memory uses disk space as an extension of main memory. Swap-space can be carved out of the normal file system, or, more commonly, it can be in a separate disk partition. Swap-space management allocates swap space 4.3BSD. When process starts, holds text segment (the program) and data segment. Kernel uses swap maps to track swap-space use. Solaris 2 allocates swap space only when a page is forced out of physical memory, not when the virtual memory page is first created. File data written to swap space until write to file system requested other dirty pages go to swap space due to no other home. Text segment pages thrown out and reread from the file system as needed

5. Explain in detail about Kernel architecture of Linux? (APR '13)(NOV '13)

Linux is a modern, free operating system based on UNIX standards. First developed as a small but self-contained kernel in 1991 by Linux Torvalds, with the major design goal of UNIX compatibility. Its history has been one of collaboration by many users from all around the world, corresponding almost exclusively over the Internet. It has been designed to run efficiently and reliably on common PC hardware, but also runs on a variety of other platforms. The core Linux operating system kernel is entirely original, but it can run much existing free.

The Linux Kernel

Version 0.01 (May 1991) had no networking, ran only on 80386-compatible Intel processors and on PC hardware, had extremely limited device-drive support, and supported only the Minix file system

Linux Distributions:

Standard, precompiled sets of packages, or *distributions*, include the basic Linux system, system installation and management utilities, and ready-to-install packages of common UNIX tools.The first distributions managed these packages by simply providing a means of unpacking all the files into the appropriate places; modern distributions include advanced package management. Early distributions included SLS and **Slack ware. Slack ware** represents overall improvement in quality. ***Red Hat*** **and** ***Debian*** is popular distributions from commercial and noncommercial sources, respectively

The RPM Package file format permits compatibility among the various Linux distributions

Linux Licensing

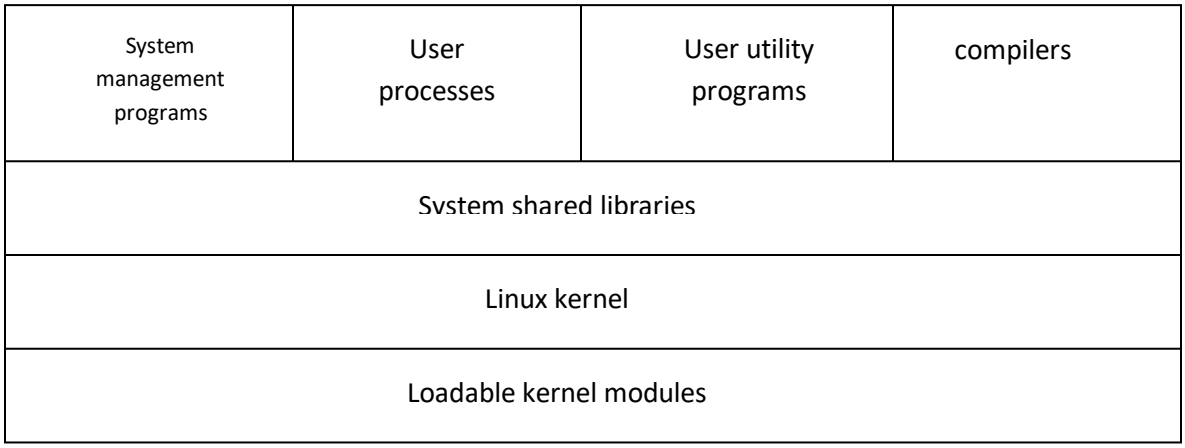
- The Linux kernel is distributed under the GNU General Public License (GPL), the terms of which are set out by the Free Software Foundation
- Anyone using Linux, or creating their own derivative of Linux, may not make the derived product proprietary; software released under the GPL may not be redistributed as a binary-only product

Design Principles

1. Linux is a multi-user, multitasking system with a full set of UNIX-compatible tools
2. Its file system adheres to traditional UNIX semantics, and it fully implements the standard UNIX networking model
3. Main design goals are speed, efficiency, and standardization
4. Linux is designed to be compliant with the relevant POSIX documents; at least two Linux distributions have achieved official POSIX certification
5. The Linux programming interface adheres to the SVR4 UNIX semantics, rather than to BSD behavior

COMPONENTS OF A LINUX SYSTEM

Like most UNIX implementations, Linux is composed of three main bodies of code; the most important distinction between the kernel and all other components. The **kernel** is responsible for maintaining the important abstractions of the operating system Kernel code executes in *kernel mode* with full access to all the physical resources of the computer. All kernel code and data structures are kept in the same single address space. The **system libraries** define a standard set of functions through which applications interact with the kernel, and which implement much of the operating-system functionality that does not need the full privileges of kernel code. The **system utilities** perform individual specialized management tasks . **Components of a Linux system shown below**



Kernel Modules

Sections of kernel code that can be compiled, loaded, and unloaded independent of the rest of the kernel

- A kernel module may typically implement a device driver, a file system, or a networking protocol
- The module interface allows third parties to write and distribute, on their own terms, device drivers or file systems that could not be distributed under the GPL
- Kernel modules allow a Linux system to be set up with a standard, minimal kernel, without any extra device drivers built in

Three components to Linux module support:

- **Module management** –allows modules to be loaded into memory.
- **Driver registration**-allows modules to tell the rest of the kernel that a new driver has become available.
- **Conflict resolution**-allows different device drivers to reserve hardware resources and to protect those resources from accidental use by another driver.

Module Management

Supports loading modules into memory and letting them talk to the rest of the kernel. Module loading is split into two separate sections:

1. Managing sections of module code in kernel memory
2. Handling symbols that modules are allowed to reference

The module requestor manages loading requested, but currently unloaded, modules; it also regularly queries the kernel to see whether a dynamically loaded module is still in use, and will unload it when it is no longer actively needed

Driver Registration

Allows modules to tell the rest of the kernel that a new driver has become available the kernel maintains dynamic tables of all known drivers, and provides a set of routines to allow drivers to be added to or removed from these tables at any time. Registration tables include the following items:

Device drivers-

These drivers include character devices (such as printers, terminals) block devices (including all disk drivers), and network interface devices.

File systems –

It implements Linux's virtual –file –system calling routines.

Network protocols-

It implements the entire networking protocol, such as IPX, or a new set of packet-filtering rules for a network firewall.

Binary format-

Specifies a way of recognizing, and loading ,a new type of executable file.

Conflict Resolution

A mechanism that allows different device drivers to reserve hardware resources and to protect those resources from accidental use by another driver.

The conflict resolution module aims to:

- Prevent modules from clashing over access to hardware resources
- Prevent ***auto probes*** from interfering with existing device drivers
- Resolve conflicts with multiple drivers trying to access the same hardware

6. Explain in Detail about Process management? (NOV'14)

UNIX process management separates the creation of processes and the running of a new program into two distinct operations.

1. The fork system call creates a new process
2. A new program is run after a call to execute

Under UNIX, a process encompasses all the information that the operating system must maintain *t* track the context of a single execution of a single program. Under Linux, process properties fall into three groups: the process's identity, environment, and context Process Identity

Process ID (PID)-

The unique identifier for the process; used to specify processes to the operating system when an application makes a system call to signal, modify, or wait for another process

Credentials-

Each process must have an associated user ID and one or more group IDs that determine the process's rights to access system resources and files

Personality-

Not traditionally found on UNIX systems, but under Linux each process has an associated personality identifier that can slightly modify the semantics of certain system calls

Used primarily by emulation libraries to request that system calls be compatible with certain specific flavors of UNIX

Process Environment

The process's environment is inherited from its parent, and is composed of two null-terminated vectors:

1. The argument vector lists the command-line arguments used to invoke the running program; conventionally starts with the name of the program itself
2. The environment vector is a list of "NAME=VALUE" pairs that associates named environment variables with arbitrary textual values

Passing environment variables among processes and inheriting variables by a process's children are flexible means of passing information to components of the user-mode system software. The environment-variable mechanism provides a customization of the operating system that can be set on a per-process basis, rather than being configured for the system as a whole

Process Context

- The (constantly changing) state of a running program at any point in time
- The **scheduling context** is the most important part of the process context; it is the information that the scheduler needs to suspend and restart the process.
- The kernel maintains **accounting** information about the resources currently being consumed by each process, and the total resources consumed by the process in its lifetime so far.
- The **file table** is an array of pointers to kernel file structures. When making file I/O system calls, processes refer to files by their index into this table .Whereas the file table lists the existing open files, the file**-system context** applies to requests to open new files. The current root and default directories to be used for new file searches are stored here.
- The **signal-handler table** defines the routine in the process's address space to be called when specific signals arrive. The **virtual-memory context** of a process describes the full contents of the private address space.

Processes and Threads

Linux uses the same internal representation for processes and threads; a thread is simply a new process that happens to share the same address space as its parent .A distinction is only made when a new thread is created by the **clone** system call.

- **fork** creates a new process with its own entirely new process context
- **Clone** creates a new process with its own identity, but that is allowed to share the data structures of its parent.

Using **clone** gives an application fine-grained control over exactly what is shared between two threads.

7. Explain in detail about Process Scheduling?

Linux uses two process-scheduling algorithms:

1.A time-sharing algorithm for fair preemptive scheduling between multiple processes

2. A real-time algorithm for tasks where absolute priorities are more important than fairness

A process's scheduling class defines which algorithm to apply .For time-sharing processes; Linux uses a prioritized, credit based algorithm. The crediting rule factors in both the process's history and its priority This crediting system automatically prioritizes interactive or I/O-bound processes. Linux implements the FIFO and round-robin real-time scheduling classes; in both cases, each process has a priority in addition to its scheduling class.

The scheduler runs the process with the highest priority; for equal-priority processes, it runs the process waiting the longest. FIFO processes continue to run until they either exit or block .A round-robin process will be preempted after a while and moved to the end of the scheduling queue, so that round-roping processes of equal priority automatically time-share between themselves.

Symmetric Multiprocessing

Linux 2.0 was the first Linux kernel to support SMP hardware; separate processes or threads can execute in parallel on separate processors. To preserve the kernel's nonpreemptible synchronization requirements, SMP imposes the restriction, via a single kernel spin lock, that only one processor at a time may execute kernel-mode code

The job of allocating CPU time to different tasks within an operating system. While scheduling is normally thought of as the running and interrupting of processes, in Linux, scheduling also includes the running of the various kernel tasks .Running kernel tasks encompasses both tasks that are requested by a running process and tasks that execute internally on behalf of a device driver. new scheduling algorithm – preemptive, priority-based

- Real-time range
- Nice value

Kernel Synchronization

A request for kernel-mode execution can occur in two ways:

1. A running program may request an operating system service, either explicitly via a system call, or implicitly, for example, when a page fault occurs
2. A device driver may deliver a hardware interrupt that causes the CPU to start executing a kernel-defined handler for that interrupt.

Kernel synchronization requires a framework that will allow the kernel's critical sections to run without interruption by another critical section.

Linux uses two techniques to protect critical sections:

1.Normal kernel code is nonpreemptible when a time interrupt is received while a process is executing a kernel system service routine, the kernel's need reached flag is set so that the scheduler will run once the system call has completed and control is about to be returned to user mode.

2. The second technique applies to critical sections that occur in an interrupt service routine By using the processor's interrupt control hardware to disable interrupts during a critical section, the kernel guarantees that it can proceed without the risk of concurrent access of shared data structures.

To avoid performance penalties, Linux's kernel uses a synchronization architecture that allows long critical sections to run without having interrupts disabled for the critical section's entire duration. Interrupt service routines are separated into a *top half* and a *bottom half.*

The top half is a normal interrupt service routine, and runs with recursive interrupts disabled. The bottom half is run, with all interrupts enabled, by a miniature scheduler. That ensures that bottom halves never interrupt themselves this architecture is completed by a mechanism for disabling selected bottom halves while executing normal, foreground kernel code.

Interrupt Protection Levels

Top-half interrupt handlers
Bottom-half interrupt handlers
Kernel system service routines (preemptible)
User mode programs (preemtible)

Fig 5.2 Interrupt protocol Level

Each level may be interrupted by code running at a higher level, but will never be interrupted by code running at the same or a lower level. User processes can always be preempted by another process when a time-sharing scheduling interrupt occurs.

8. Explain how interprocess communication is carried out in Linux OS? (NOV '12) (APR'13) (APR'15)

Interprocess Communication

Many operating systems provide mechanisms for interprocess communication (IPC)

Processes must communicate with one another in multiprogrammed and networked environments

For example, a Web browser retrieving data from a distant server

Essential for processes that must coordinate activities to achieve a common goal.

Signals

Software interrupts that notify a process that an event has occurred. Do not allow processes to specify data to exchange with other processes. Processes may catch, ignore or mask a signal

- Catching a signal involves specifying a routine that the OS calls when it delivers the signal
- Ignoring a signal relies on the operating system's default action to handle the signal
- Masking a signal instructs the OS to not deliver signals of that type until the process clears the signal mask

Message Passing

Message-based interprocess communication Messages can be passed in one direction at a time .One process is the sender and the other is the receiver Message passing can be bidirectional

Each process can act as either a sender or a receiver. Messages can be blocking or nonblocking

Blocking requires the receiver to notify the sender when the message is received.Nonblocking enables the sender to continue with other processing. Popular implementation is a pipe A region of memory protected by the OS that serves as a buffer, allowing two or more processes to exchange data.

LUNIX Processes

- LUNIX processes

  All processes are provided with a set of memory addresses, called a virtual address space

  A process's PCB is maintained by the kernel in a protected region of memory that user processes cannot access

  A LUNIX PCB stores:

  - The contents of the processor registers

- PID
- The program counter
- The system stack

All processes are listed in the process table

9. Explain in detail about Linux advantage?

Linux Advantages

1. Low cost: You don't need to spend time and money to obtain licenses since Linux and much of its software come with the GNU General Public License. You can start to work immediately without worrying that your software may stop working anytime because the free trial version expires. Additionally, there are large repositories from which you can freely download high quality software for almost any task you can think of.
2. Stability: Linux doesn't need to be rebooted periodically to maintain performance levels. It doesn't freeze up or slow down over time due to memory leaks and such. Continuous up-times of hundreds of days (up to a year or more) are not uncommon.
3. Performance: Linux provides persistent high performance on workstations and on networks. It can handle unusually large numbers of users simultaneously, and can make old computers sufficiently responsive to be useful again.
4. Network friendliness: Linux was developed by a group of programmers over the Internet and has therefore strong support for network functionality; client and server systems can be easily set up on any computer running Linux. It can perform tasks such as network backups faster and more reliably than alternative systems.
5. Flexibility: Linux can be used for high performance server applications, desktop applications, and embedded systems. You can save disk space by only installing the components needed for a particular use. You can restrict the use of specific computers by installing for example only selected office applications instead of the whole suite.
6. Compatibility: It runs all common Unix software packages and can process all common file formats.
7. Choice: The large number of Linux distributions gives you a choice. Each distribution is developed and supported by a different organization. You can pick the one you like best; the core functionalities are the same; most software runs on most distributions.
8. Fast and easy installation: Most Linux distributions come with user-friendly installation and setup programs. Popular Linux distributions come with tools that make installation of additional software very user friendly as well.

9. Full use of hard disk: Linux continues work well even when the hard disk is almost full.
10. Multitasking: Linux is designed to do many things at the same time; e.g., a large printing job in the background won't slow down your other work.
11. Security: Linux is one of the most secure operating systems. "Walls" and flexible file access permission systems prevent access by unwanted visitors or viruses. Linux users have to option to select and safely download software, free of charge, from online repositories containing thousands of high quality packages. No purchase transactions requiring credit card numbers or other sensitive personal information are necessary.
12. Open Source: If you develop software that requires knowledge or modification of the operating system code, Linux's source code is at your fingertips. Most Linux applications are Open Source as well.

10. Explain how the memory and file management is implemented in windows XP? (APR'13)

Memory Management

Virtual memory manager (VMM):

Executive component responsible for managing memory. Lazy allocation: Avoid allocating memory until necessary. Perfecting: Move pages from disk to main memory before they are needed

Page file: Stores pages that do not fit in main memory, Windows XP supports up to 16 page files

Memory Organization

In an 2-bit virtual address space consists of Windows 64-Bit Edition has 64-bit address space. 4GB virtual address space per process. Process can access only user space.VMM stores page tables and other data in system space 2GB user space, 2GB system space, 4KB pages. In an 2-level hierarchical memory map consists of Page directory table, page table, page frame. It includes Page directory entries (PDEs) point to page table consists of one page directory table per process, Location in page directory register, Page table- Page table entries (PTEs) point to page frames Page frame Contains page of data.

Windows File system:

Windows supports a number of file systems, including the file allocation table (FAT) that runs on Windows 95, MS-DOS, and OS/2. But the developers of Windows also designed a new file System, the Windows File System (NTFS), that is intended to meet high-end requirements for Workstations and servers. Examples of high-end applications: Client/server applications such as file servers, compute servers, and database servers, Resource-intensive engineering and scientific applications. Network applications for large corporate systems .NTFS is a flexible and powerful file system built, as

which shall see, on an elegantly simple file system model. The most noteworthy features of NTFS include: Recoverability:

High on the list of requirements for the new Windows file system was the ability to recover from system crashes and disk failures. In the event of such failures, NTFS is able to reconstruct disk volumes and return them to a consistent state. It does this by using a transaction processing model for changes to the file system; each significant change is treated as an atomic action that is either entirely performed or not performed at all. Each transaction that was in process at the time of a failure is subsequently backed out or brought to completion. In addition, NTFS uses redundant storage for critical file system data, so that failure of a disk sector does not cause the loss of data describing the structure and status of the file system.

Security:

NTFS uses the Windows object model to enforce security. An open file is implemented as a file object with a security descriptor that defines its security attributes.

Large disks and large files:

NTFS supports very large disks and very large files more efficiently than most other file systems, including FAT.

Multiple data streams:

The actual contents of a file are treated as a stream of bytes. In NTFS it is possible to define multiple data streams for a single file. An example of the utility of this feature is that it allows Windows to be used by remote Macintosh systems to store and retrieve files. On Macintosh, each file has two components: the file data and a resource fork that contains information about the file. NTFS treats these two components as two data streams.

General indexing facility:

NTFS associates a collection of attributes with each file. The set of file descriptions in the file management system is organized as a relational database, so that files can be indexed by any attribute.

11. Explain in detail about Security? (NOV'14)

The *pluggable authentication modules (PAM)* system is available under Linux. PAM is based on a shared library that can be used by any system component that needs to authenticate users .Access control under UNIX systems, including Linux, is performed through the use of unique numeric identifiers (**uid** and **gid**). Access control is performed by assigning objects a *protections mask*, which specifies which access modes—read, write, or execute—are to be granted to processes with owner, group, or world access

Linux augments the standard UNIX **setid** mechanism in two ways:

It implements the POSIX specification's saved *user-id* mechanism, which allows a process to repeatedly drop and reacquire its effective uid .It has added a process characteristic that grants just a subset of the rights of the effective uid. Linux provides another mechanism that allows a client to selectively pass access to a single file to some server process without granting it any other privileges.

12. Explain in detail about Windows XP System Architecture?(NOV '12) (APR'15)

The architecture of windows xp consist of 2 modes

Protected mode

User mode

The main layers of protected mode are hardware abstraction layer, kernel and executive.

The collection of subsystem and services are called as user mode.

User mode subsystems fall into two categories.

Environment subsystem – which emulate different operating system

Protection subsystem – which provide security function

Protected mode:

1. **Hardware abstraction layer :**

The HAL is the layer of software that hides hardware differences from upper levels of the operating system to help make Windows XP portable.

The HAL exports a virtual machine interface that is used by the kernel dispatcher , executive and the drivers.

The advantage of this approach is that only a single version of each device driver is required.

The HAL also provide support for symmetric multiprocessing.

2. **Kernel:**

The kernel of windows xp provides the foundations for the executive and the subsystems.

The kernel remains in memory and its execution is never preempted. It has 4 responsibilities.

They are thread scheduling, interrupt and execution handling, low-level processor synchronization and recovery after a power failure.

The kernel is object oriented. Kernel has kernel dispatcher.

Kernel dispatcher:

Kernel dispatcher provides the foundation for the executive and subsystems.

The dispatcher is never paged out of memory and its execution is never preempted.

Its main responsibilities are thread scheduling , implementation of synchronization primitives , software interrupts and execution dispatching.

- Thread scheduling:

Windows xp uses the concepts of processor and thread for executable code.

The process has a virtual memory address and information used to initialize each thread.

Each process has one or more thread each of which is an executable unit dispatched by the kernel.

There are 6 possible thread states. They are

Ready – indicates waiting to run

Standby – highest priority ready thread is moved to standby state

Running –executing on a processor

Waiting –waiting for an dispatcher object

Transition –a new thread is in transition state while its waits for resources necessary for execution.

Termination –finishes execution.

- Implementation of synchronization primitives:

The os data structures are managed as object using common facilities for allocation, reference counting and security.

- Software interrupt:

2 types of software interrupt

- Asynchronous procedure call:

  It is used to begin execution of a new thread, terminate processes and to deliver notification that an asynchronous I/O has completed.

- Deferred procedure call:

  It is used to postpone interrupt processing.

- Exception and interrupts:

The Windows XP has several exception including

1. Memory access violation
2. Integer overflow
3. Floating point overflow or underflow
4. Integer divide by zero
5. Floating point divide by zero etc.

This exception are handled by the exception dispatcher when an exception occurs in kernel mode, the exception dispatcher simply calls a routine to locate the exception handler.

3. Executive :

The Windows XP executive provides a set of services that all environment subsystems use.

The services are grouped as follows:

- Object manager ,
- virtual memory manager ,
- process manager ,
- local procedure call facility ,
- I/O manager,
- security references monitor ,
- plug and play and security manager ,
- register and booting.

➢ Object manager

Windows XP uses a generic set of interface s for managing the kernel entities that is manipulating by user mode program.

Windows XP calls these entities objects and the executive component that manipulate them is the object manager.

The job of the object manager is to supervise the use of all the managed objects.

When a thread wants to use an object, it calls the object managers open method to get a reference to the object.

➢ Virtual memory manager:

Then executive component that manages the virtual address space, physical memory allocation and paging is the virtual memory manager.

The design of VM manager assumes that the underlying hardware support virtual to physical mapping, a paging mechanism , transparent cache coherence on multiprocessor system and allow multiple page table entries to map to the same physical page frame.

The VM manager for IA32 processor has a page directory that contains 1024 page directory entries (PDE) of size 4 bytes. Each PTE point to a 4 kb page frames.

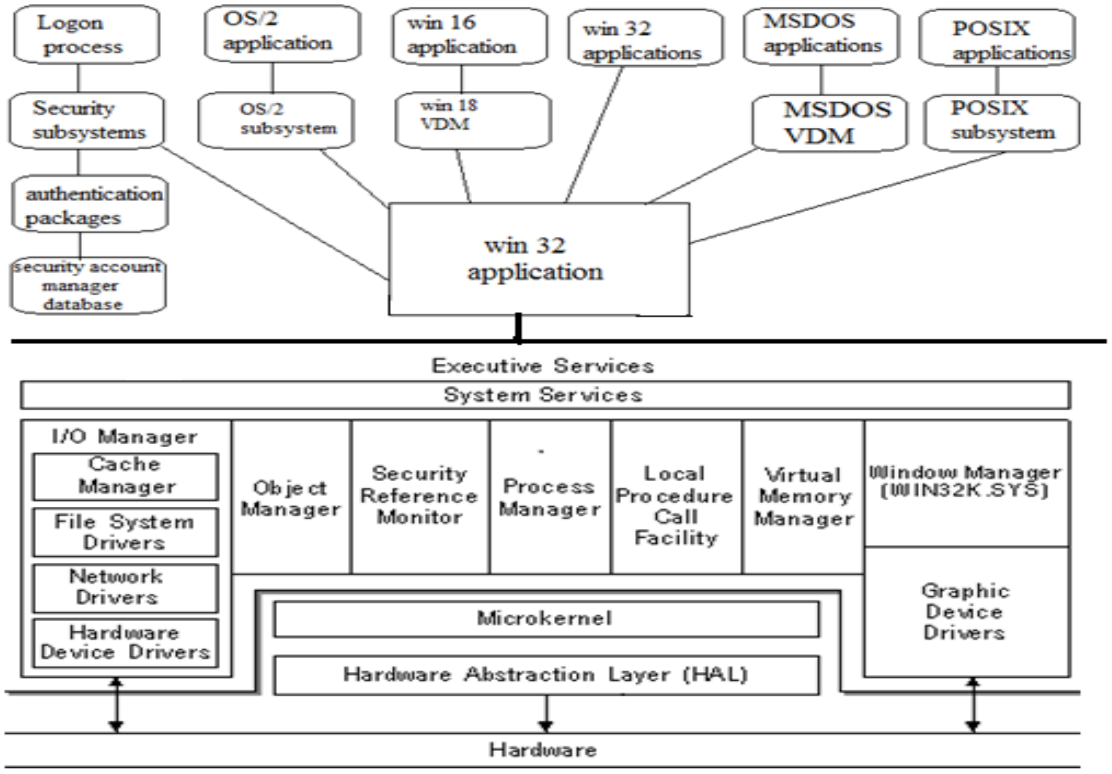


Fig 5.3 Block Diagram of Windows XP

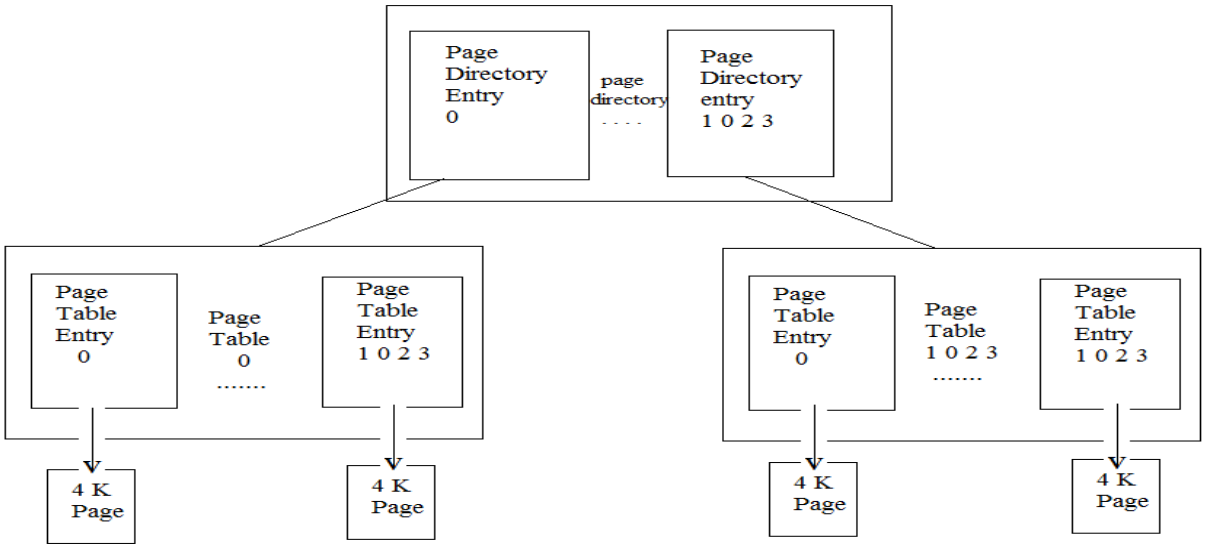


Fig 5.4 PTE point to a 4 kb page frame.

A 32 bit virtual memory address is splitted into 3 value

i. First 10 bit – used as an index into page directory

ii. Next 10 bit – used to select a file from page table

iii. The remaining 12 bits are the offset of the specific in the page frame.

PDE	PTE	Page offset

Fig 5.5 Page Directory

- Process manager:

The windows xp process manager provides services for creating, deleting and using processes, threads and jobs.

It has no knowledge about parent child relationship or process hierarchies.

The pro manager is also not involved in the scheduling of processes other than setting the priorities

And the affirmatives in processes and threads when they are created.

Thread scheduling takes place in kernel dispatcher.

- Local procedure call facility:

The implementation of Windows XP uses a client server model.

The OS uses local procedure call (LPC) to pass request and result between client and server process within a single machine.

The various Windows XP subsystems.

When an LPC channel is created, one of three message passing techniques must be specified.

1. The first tech is suitable for small messages. In this parts message queue is used as intermediate storage, the message are copied from one process to the other.
2. The second tech is for large messages. Message send through the ports message queue contain a pointer and size information referring to the section object.

   This avoids the need to copy large messages. The sender places data into the shared section and receiver views them directly.
3. The third tech of LPC message passing uses the API that read / writes directly into a process address space.

- I/O manager :

I/O manager is responsible for file system, device drivers and network drivers.

It keeps track of which device drivers, filter drivers and file systems are loaded and also manages buffer for I/O request.

Device driver are arranged as a list for each the I/O manager convert the request it receive into standard form called as I/O request packet (IRP)

- Cache manager:

In many OS, caching is done by the file system. The cache manager works closely with the VM manager to provide cache services for all components under the control of I/O manager.

- Security reference monitor:

  The security reference monitor (SRM) is also responsible for manipulating the privilege in security tokens.

  Whenever a process opens a handle to an object the security reference monitor (SRM) checks the process.

- Plug and play and power manager:

  The OS uses the Plug and play manager to recognize and adapt the change in the hardware configuration.

  Windows XP also moves to system to a state requiring lower power consumption.

2. User mode:

1. **Environmental subsystem:**

   It is a user modes processes layered over the native windows xp executive services to enable windows xp to run program developed for other os including 16 bit windows , MS DOS and POSIX.

   - MS DOS environment:

     The MS-DOS environment does not have of other windows xp environment subsysytems.

     It is provided by a win 32 application called the virtual DOS machine (VDM)

   - 16-bit windows environment:

     The win16 execution environment is provided by a VDM that incorporates additional software called windows on windows that provides the windows 3.1 kernel routines and stub routine for window manager and graphical device interface functions.

   POSIX subsystem:

   The POSIX system is designed to run POSIX application written to follow the POSIX standard, which is based on the UNIX model.

   POSIX applications can be started by the win 32 sub system or by another POSIX application.

2. **Logon and security subsystems:**

   Before a user can access objects on windows xp, that user must be authenticated by the logon service, win logon.

   It is responsible for responding to secure attention sequence.

The local security authority subsystem (LSASS) is the process that generates access tokens to represent users on the subsystems.

It calls an authentication package to perform authentication using information from the log on subsystem or network server.

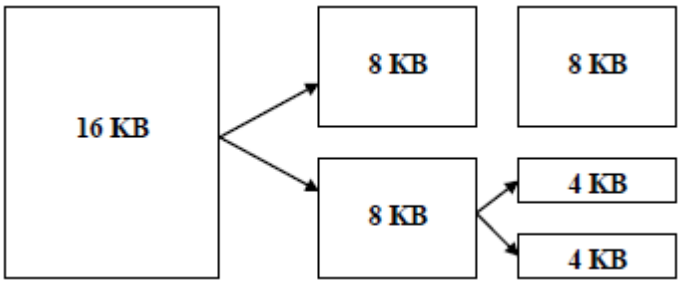
The security subsystem then generates the access token for the user ID containing the appropriate privileges , quota limits and group IDS.

The default authentication package for windows xp domains in Kerberos.

13. Describe about memory allocation stages. (11)(NOV '13)

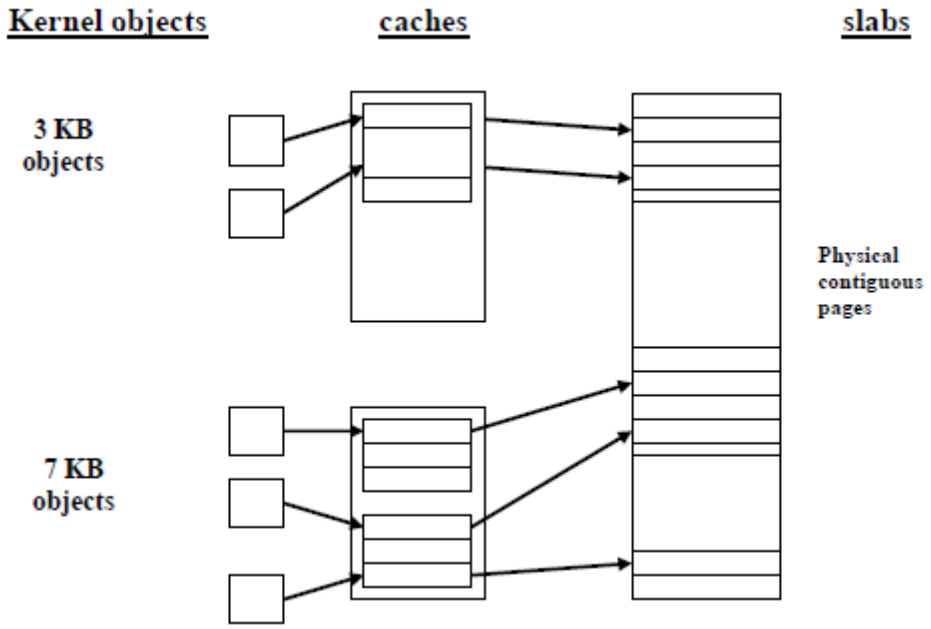
Linux's physical memory-management system deals with allocating and freeing pages, groups of pages, and small blocks of memory. It has additional mechanisms for handling virtual memory, memory mapped into the address space of running processes. Splits memory into 3 different zones due to hardware characteristics.

Splitting of Memory in a Buddy Heap



Managing Physical Memory

The page allocator allocates and frees all physical pages; it can allocate ranges of physically-contiguous pages on request. The allocator uses a buddy-heap algorithm to keep track of available physical pages. Each allocatable memory region is paired with an adjacent partner. Whenever two allocated partner regions are both freed up they are combined to form a larger region. If a small memory request cannot be satisfied by allocating an existing small free region, then a larger free region will be subdivided into two partners to satisfy the request Memory allocations in the Linux kernel occur either statically (drivers reserve a contiguous area of memory during system boot time) or dynamically (via the page allocation). Also uses slab allocate for kernel memory.



Virtual Memory

The VM system maintains the address space visible to each process: It creates pages of virtual memory on demand, and manages the loading of those pages from disk or their swapping back out to disk as required.

The VM manager maintains two separate views of a process's address space: A logical view describing instructions concerning the layout of the address space.

The address space consists of a set of no overlapping regions, each representing a continuous, page-aligned subset of the address space. A physical view of each address space which is stored in the hardware page tables for the process. Virtual memory regions are characterized by:

1. The backing store, which describes from where the pages for a region come; regions are usually backed by a file or by nothing (*demand-zero* memory).
2. The region's reaction to writes (page sharing or copy-on-write)

The kernel creates a new virtual address space. When a process runs a new program with the **exec** system call. Upon creation of a new process by the **fork** system call. On executing a new program, the process is given a new, completely empty virtual-address space; the program-loading routines populate the address space with virtual-memory regions.

Creating a new process with **fork** involves creating a complete copy of the existing process's virtual address space. The kernel copies the parent process's VMA descriptors, then creates a new set of page tables for the child. The parent's page tables are copied directly into the child's, with the reference count of each page covered being incremented. After the fork, the parent and child share the same physical pages of memory in their address spaces. The VM paging system

relocates pages of memory from physical memory out to disk when the memory is needed for something else. The VM paging system can be divided into two sections:

1. The page out-policy algorithm decides which pages to write out to disk, and when

2. When, the paging mechanism actually carries out the transfer, and pages data back into physical memory as needed. The Linux kernel reserves a constant, architecture-dependent region of the virtual address space of every process for its own internal use .

This kernel virtual-memory area contains two regions:

1. A static area that contains page table references to every available physical page of memory in the system, so that there is a simple translation from physical to virtual addresses when running kernel code

2. The reminder of the reserved section is not reserved for any specific purpose; its page table entries can be modified to point to any other areas of memory.

Executing and Loading User Programs

Linux maintains a table of functions for loading programs; it gives each function the opportunity to try loading the given file when an exec system call is made. The registration of multiple loader routines allows Linux to support both the ELF and a. Out binary formats. Initially, binary-file pages are mapped into virtual memory .

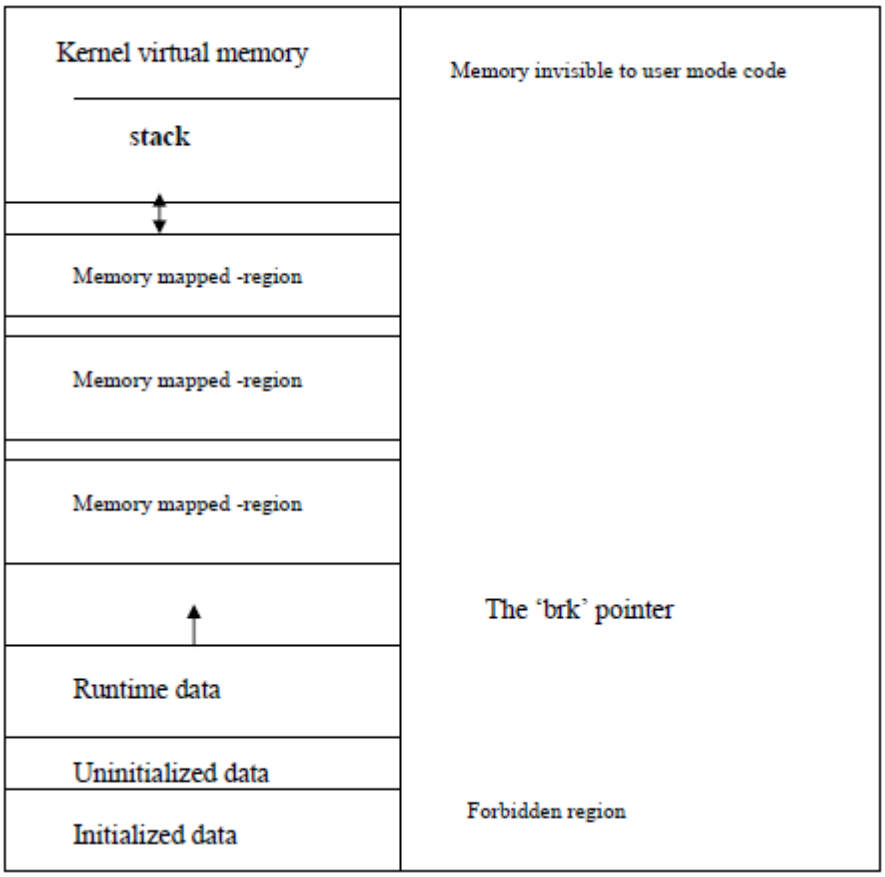
Only when a program tries to access a given page will a page fault result in that page being loaded into physical memory. An ELF-format binary file consists of a header followed by several page-aligned sections. The ELF loader works by reading the header and mapping the sections of the file into separate regions of virtual memory

Static and Dynamic Linking

A program whose necessary library functions are embedded directly in the program's executable binary file is ***statically*** linked to its libraries. The main disadvantage of static linkage is that very program generated must contain copies of exactly the same common system library functions.

Dynamic linking is more efficient in terms of both physical memory and disk-space usage because it loads the system libraries into memory only once.

Memory Layout for ELF Programs



14. Explain how the following mechanisms are implemented in Windows XP?

(1)Thread (2) IPC (APR '12)

Thread:

The idea is to have separate threads of control (hence the name) running in the same address space. An *address space* is a memory management concept. For now think of an address space as the memory in which a process runs and the mapping from the virtual addresses (addresses in the program) to the physical addresses (addresses in the machine). Each thread is somewhat like a process (e.g., it is scheduled to run) but contains less state (e.g., the address space belongs to the process in which the thread runs.

The Thread Model:

A process contains a number of resources such as address space, open files, accounting information, etc. In addition to these resources, a process has a thread of control, e.g., program counter, register contents, stack. The idea of threads is to permit multiple threads of control to execute within one process. This is often called **multithreading** and threads are often called **lightweight processes**. Because threads in the same process share so much state, switching between them is much less expensive than switching between separate processes.

Individual threads within the same process are not completely independent. For example there is no memory protection between them. This is typically not a security problem as the threads are cooperating and all are from the same user (indeed the same process). However, the shared resources do make debugging harder. For example one thread can easily overwrite data needed by another and if one thread closes a file other threads can't read from it.

Implementing Threads in the Kernel:

Move the thread operations into the operating system itself. This naturally requires that the operating system itself be (significantly) modified and is thus not a trivial undertaking.

- Thread-create and friends are now system calls and hence much slower than with user-mode threads. They are, however, still much faster than creating/switching/etc processes since there is so much shared state that does not need to be recreated.
- A thread that blocks causes no particular problem. The kernel can run another thread from this process or can run another process.
- Similarly a page fault, or infinite loop in one thread does not automatically block the other threads in the process.

Hybrid Implementations:

One can write a (user-level) thread library even if the kernel also has threads. This is sometimes called the M:N model since M user mode threads run on each of N kernel threads. Then each kernel thread can switch between user level threads. Thus switching between user-level threads within one kernel thread is very fast (no context switch) and we maintain the advantage that a blocking system call or page fault does not block the entire multi-threaded application since threads in other processes of this application are still runnable.

IPC:

- Many operating systems provide mechanisms for interprocess communication (IPC)
- Processes must communicate with one another in multiprogrammed and networked environments
- For example, a Web browser retrieving data from a distant server
- Essential for processes that must coordinate activities to achieve a common goal

Message-based interprocess communication

- Messages can be passed in one direction at a time
  - One process is the sender and the other is the receiver
- Message passing can be bidirectional
- Each process can act as either a sender or a receiver
- Messages can be blocking or nonblocking

- Blocking requires the receiver to notify the sender when the message is received
- Nonblocking enables the sender to continue with other processing

➢ Popular implementation is a pipe

- A region of memory protected by the OS that serves as a buffer, allowing two or more processes to exchange data

15. Explain in detail about Security?

The security of NTFS volume is derived from the windows XP object model. Each NTFS file references a security description which contains

- Access token of the owner of the file.
- Access control list.
- Access privileges.

Traversal checks are inherently more expensive.

1.Volume management and fault tolerance:

➢ Ftdisk is the fault tolerant disk driver.

➢ Ftdisk provides several ways to combine multiple disk drives into one logical volume.

➢ It improves performances, capacity or reliability.

2.Volume set:

➢ In windows XP logical volume is called a volume set.

➢ It is one of the way to combine multiple disk.

➢ They concatenate all multiple disks logically to form large logical volume.

➢ It consist of 32 physical partitions.

➢ It can be extended without disturbing the data already stored in file system.

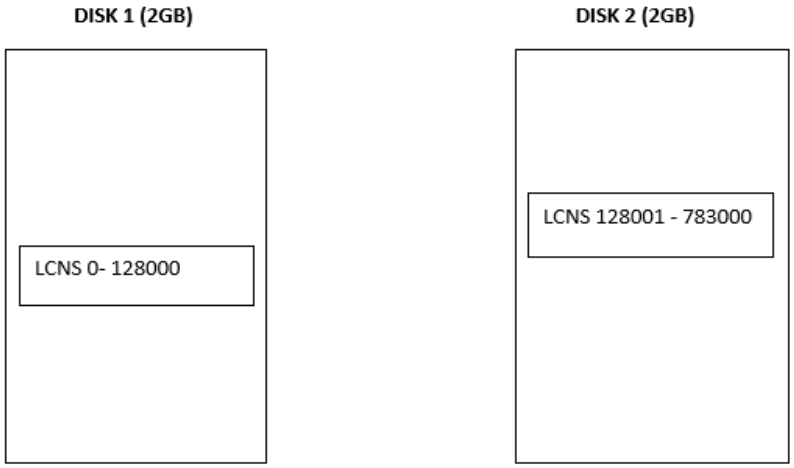


Fig 5.6 Volume set of two drives

3.Stripe set:

- It is another way to combine multiple physical partitions and to interleave their blocks in round robin fashion.
- It is also called as disk stription or RAID level 0.
- Ft disk uses a stripe set of 64 KBS.
- A stripe set forms one large logical volume but the physical layout can improve the I/O bandwidth, for a large I/O all the disk can transfer data in parallel.

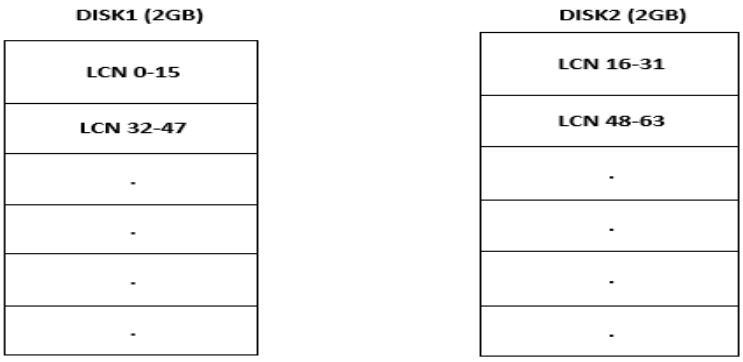


Fig 5.7 Stripe set on two drives

4Stripe set with parity:

- Stripe set with parity is also called as RAID level 5.
- If the stripe set has eight disks then, for each of the seven data strips, on seven separate disks, there is a parity stripe in the eight disks.
- The parity stripe contains the bit-wise exclusive OR of the data stripes.
- If any one of the data stripe is destroyed the system can reconstruct the data by calculating the exclusive OR of the remaining seven.
- The data loss occurs very less.
- As all the seven data stripes have seven parity the I/O load is high.

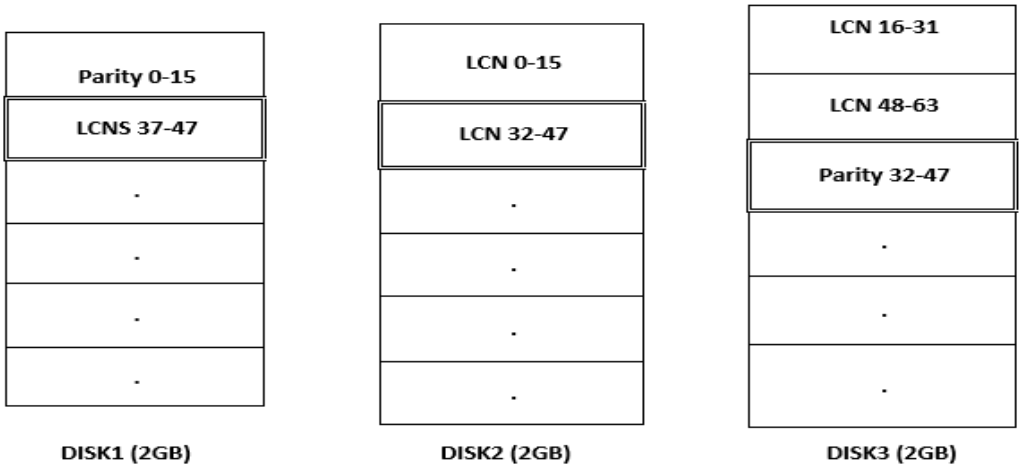


Fig 5.8 Stripe set with parity

5. Data Mirroring:

- Disk mirroring is also called as mirror sets or RAID level1 or duplex sets.
- A mirror set comprises of two equal sized partitions on two disks such that their data are identical.
- When an application writes data to a mirror set, Ftdisk writes the data in both partition.
- If one partition fails Ftdisks has another copy safely stored on the mirror.
- It improves the performance.
- We can attach the disks of a mirror set to separate disk controller. This arrangement is called duplex set.

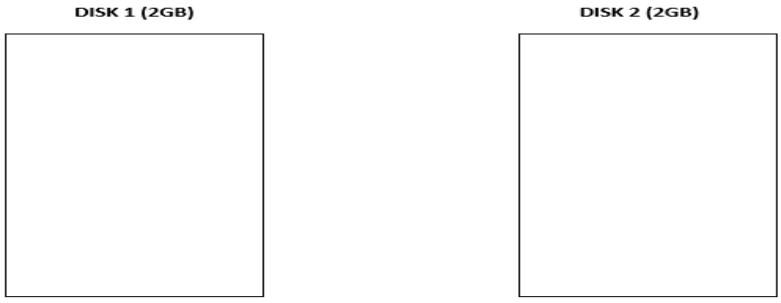


Fig 5.9 Mirror set on two drives

6. Sector sparing and cluster remapping:

- Disk uses a hardware technique called sector sparing.
- When a disk is formatted, it create a map from logical block number to good sector on the disk.
- If a sector fails, disk instructs the disk drive to substitute a space.
- NTFS uses a software technique called cluster remapping.

- If a disk block goes bad, NTFS substitutes a different unallocated block by changing any affected pointer in the MFT.
- If read fails the missing data is reconstructed and stored into a new location that is obtained by sector sparing or cluster remapping.

7. Compression and Encryption:

- NTFS can perform data compression on individual files or on all data files in a directory.
- NTFS divides the files data into compression units, which are blocks of 16 contiguous clusters.
- When each compression unit is written, a data compression algorithm is used.
- NTFS supports encryption of files.
- In this technique it provides security to the data using encryption algorithm.
- At the next end the message can be retrieved using decryption algorithm.

16. Write in detail about Kernel I/O subsystem

Kernels provide many services related to I/0. Several services-scheduling,buffering, caching, spooling, device reservation, and error handling -are provided by the kernel's I/0 subsystem and build on the hardware and device driver infrastructure. The I/O subsystem is also responsible for protecting itself from errant processes and malicious users.

I/0 **Scheduling**

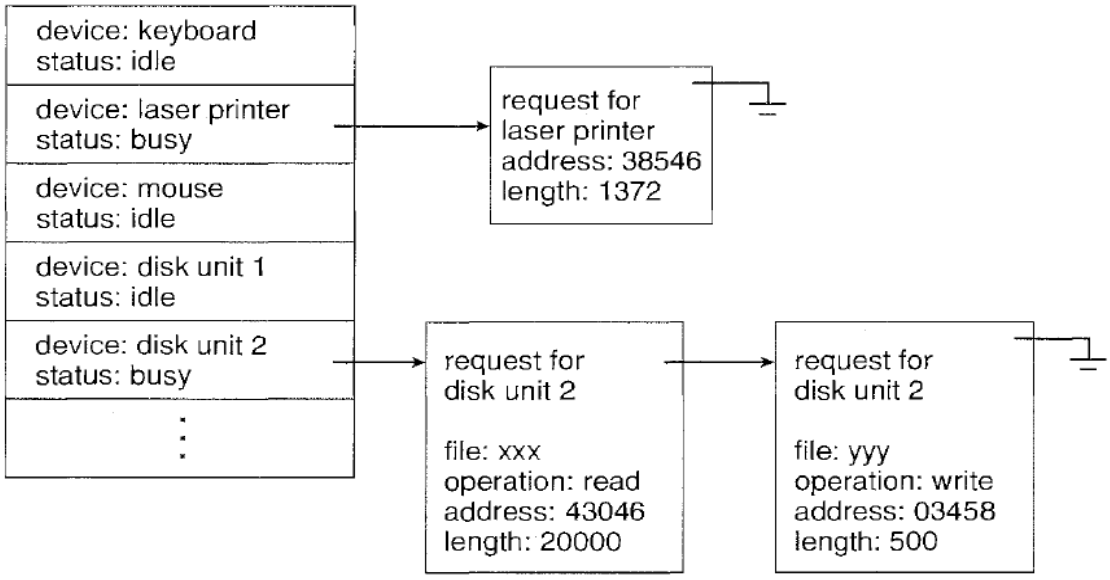
To schedule a set of I/O requests means to determine a good order in which to execute them. The order in which applications issue system calls rarely is the best choice. Scheduling can improve overall system performance, can share device access fairly among processes, and can reduce the average waiting time for I/0 to complete. Here is a simple example to illustrate. Suppose that a disk arm is near the begilming of a disk and that three applications issue blocking read calls to that disk Application 1 requests a block near the end of the disk, application 2 requests one near the beginning, and application 3 requests one in the middle of the disk The operating system can reduce the distance that the disk arm travels by serving the applications in the order 2, 3, 1. Rearranging the order of service in this way is the essence of I/0 scheduling.

Operating-system developers implement scheduling by maintaining a wait queue of requests for each device. When an application issues a blocking I/0 system call, the request is placed on the queue for that device. The I/0 scheduler rearranges the order of the queue to improve the overall system efficiency and the average response time experienced by applications. The operating system may also try to be fair, so that no one application receives especially poor service, or it may give

priority service for delay-sensitive requests. For instance, requests from the virtual memory subsystem may take priority over application requests

When a kernel supports asynchronous I/0, it must be able to keep track of many I/0 requests at the same time. For this purpose, the operating system might attach the wait queue to a device status table. The kernel manages this table, which contains an entry for each I/0 device, as shown below.

Each table entry indicates the device's type, address, and state (not functioning, idle, or busy). If the device is busy with a request, the type of request and other parameters will be stored in the table entry for that device. One way in which the I/0 subsystem improves the efficiency of the computer is by scheduling I/0 operations. Another way is by using storage space in main memory or on disk via techniques called buffering, caching, and spooling.



Buffering

A buffer is a memory area that stores data being transferred between two devices or between a device and an application. Buffering is done for three reasons. One reason is to cope with a speed mismatch between the producer and consumer of a data stream. Suppose, for example, that a file is being received via modem for storage on the hard disk The modem is about a thousand times slower than the hard disk So a buffer is created in main mernory to accumulate the bytes received from the modem. When an entire buffer of data has arrived, the buffer can be written to disk in a single operation. Since the disk write is not instantaneous and the modem still needs a place to store additional incoming data, two buffers are used. After the modem fills the first buffer, the disk write is requested. The modem then starts to fill the second buffer while the first buffer is written to disk By the time the modem has filled the second buffer, the disk write from the first one should have

completed, so the modem can switch back to the first buffer while the disk writes the second one. This double buffering decouples the producer of data from the consun1.er, thus relaxing timing requirements between them.

A second use of buffering is to provide adaptations for devices that have different data-transfer sizes. Such disparities are especially common in computer networking, where buffers are used widely for fragmentation and reassembly of messages. At the sending side, a large message is fragmented into small network packets. The packets are sent over the network, and the receiving side places them in a reassembly buffer to form an image of the source data.

A third use of buffering is to support copy semantics for application I/0.An example will clarify the meaning of "copy semantics." Suppose that an application has a buffer of data that it wishes to write to disk It calls the write() systemcalt providing a pointer to the buffer and an integer specifying the number of bytes to write. After the system call returns, what happens if the application changes the contents of the buffer? With copy semantics the version of the data written to disk is guaranteed to be version at the time of the application system calt independent of any subsequent changes in the application's buffer. A simple way in which the operating system can guarantee copy semantics is for the write () system call to copy the application data into a kernel buffer before returning control to the application. The disk write is performed from the kernel buffer, so that subsequent changes to the application buffer have no effect. Copying of data between kernel buffers and application data space is common in operating systems, despite the overhead that this operation introduces, because of the clean semantics. The same effect can be obtained more efficiently by clever use of virtual memory mapping and copy-on-write page protection.

Caching

A cache is a region of fast memory that holds copies of data. Access to the cached copy is more efficient than access to the original. For instance, the instructions of the currently running process are stored on disk, cached ilc physical memory,and copied again ill the CPU's secondary and primary caches. The difference between a buffer and a cache is that a buffer may hold the only existing copy of a data item, whereas a cache, by definition, holds a copy on faster storage of an item that resides elsewhere

Spooling and Device Reservation

A spool is a buffer that holds output for a device, such as a printer, that cannot accept interleaved data streams. Although a printer can serve only one job at a time, several applications may wish to print their output concurrently, without having their output mixed together. The operating system solves this problem by intercepting all output to the printer. Each application's output is spooled to a separate disk file. When an application finishes printing, the spooling system

queues the corresponding spool file for output to the printer.The spooling system copies the queued spool files to the printer one at a time. In some operating systems, spooling is managed by a system daemon process. In others, it is handled by an in-kernel thread. In either case, the operating system provides a control interface that enables users and system administrators to display the queue, remove unwanted jobs before those jobs print, suspend printing while the printer is serviced, and so on.

Error handling

An operating system that uses protected memory can guard against many kinds of hardware and application errors, so that a complete system failure is not the usual result of each minor mechanical glitch. Devices and I/0 transfers can fail in many ways, either for transient reasons, as when a network becomes overloaded, or for "permanent" reasons, as when a disk controller becomes defective. Operating systems can often compensate effectively for transient failures. For instance, a disk read() failure results in a read() retry, and a network send() error results in a res end(), if the protocol so specifies. Unfortunately, if an important component experiences a permanent failure, the operating system is unlikely to recover.

17. Discuss on file system structure in the linux systems. (NOV '15)

File Systems

Linux retains UNIX's standard file-system model. In UNIX, a file does not have to be an object stored on disk or fetched over a network from a remote file server. Rather, UNIX files can be anything capable of handling the input or output of a stream of data. Device drivers can appear as files, and interprocess communication channels or network connections also look like files to the user. The Linux kernel handles all these types of file by hiding the implementation details of any single file type behind a layer of software, the virtual file system (VFS).

The Virtual File System

The Linux VFS is designed around object-oriented principles. It has two components: a set of definitions that specify what file-system objects are allowed to look like and a layer of software to manipulate the objects. The VFS defines four main object types:

- An **inode object** represents an individual file.
- A **file object** represents an open file.
- A **superblock object** represents an entire file system.
- A **dentry object** represents an individual directory entry.

For each of these four object types, the VFS defines a set of operations. Every object of one of these types contains a pointer to a function table. The function table lists the addresses of the actual functions that implement the defined operations for that object.

- int open (. . .) — Open a file.
- ssize_t read(. . .) —Read from a file.
- ssize_t write (. . .) —Write to a file.
- int mmap (. . .) — Memory-map a file.

The complete definition of the file object is specified in the struct file_operations, which is located in the file /usr/include/linux/fs.h. An implementation of the file object (for a specific file type) is required to implement each function specified in the definition of the file object. The VFS software layer can perform an operation on one of the file-system objects by calling the appropriate function from the object's function table, without having to know in advance exactly what kind of object it is dealing with. The VFS does not know, or care, whether an inode represents a networked file, a disk file, a network socket, or a directory file. The appropriate function for that file's readQ operation will always be at the same place in its function table, and the VFS software layer will call that function without caring how the data are actually read.

The inode and file objects are the mechanisms used to access files. An inode object is a data structure containing pointers to the disk blocks that contain the actual hie contents, and a file object represents a point of access to the data in an open file. A process cannot access an inode's contents without first obtaining a file object pointing to the inode. The file object keeps track of where in the file the process is currently reading or writing, to keep track of sequential file I/O.

The Linux ext2fs File System

The standard on-disk file system used by Linux is called **ext2fs,** for historical reasons. Linux was originally programmed with a Minix-compatible file system, to ease exchanging data with the Minix development system, but that file system was severely restricted by 14-character file-name limits and a maximum file-system size of 64 MB. The Minix file system was superseded by a new file system, which was christened the **extended file system (extfs).** A later redesign of this file system to improve performance and scalability and to add a few missing features led to the **second extended file system (ext2fs).**

It uses a similar mechanism for locating the data blocks belonging to a specific file, storing data-block pointers in indirect blocks throughout the file system with up to three levels of indirection. As in FFS, directory files are stored on disk just like normal files, although their contents are interpreted differently.

Each block in a directory file consists of a linked list of entries; each entry contains the length of the entry, the name of a file, and the inode number of the inode to which that entry refers. The main differences between ext2fs and FFS lie in their disk-allocation policies. In FFS, the disk is allocated to files in blocks of 8 KB. These blocks are subdivided into fragments of 1 KB for storage of small files or partially filled blocks at the ends of files. In contrast, ext2fs does not use fragments at all but performs all its allocations in smaller units. The default block size on ext2fs is 1 KB, although 2-KB and 4-KB blocks are also supported.

To maintain high performance, the operating system must try to perform I/O operations in large chunks whenever possible by clustering physically adjacent I/O requests. Clustering reduces the per-request overhead incurred by device drivers, disks, and disk-controller hardware.

Journaling

Many different types of file systems are available for Linux systems. One popular feature in a file system is **journaling,** whereby modifications to the file system are sequentially written to a journal. A set of operations that performs a specific task is a **transaction.** Once a transaction is written to the journal, it is considered to be committed, and the system call modifying the file system (i.e. write ()) can return to the user process, allowing it to continue execution. Meanwhile, the journal entries relating to the transaction are replayed across the actual file-system structures. As the changes are made, a pointer is updated to indicate which actions have completed and which are still incomplete.

When an entire committed transaction is completed, it is removed from the journal. The journal, which is actually a circular buffer, may be in a separate section of the file system, or it may even be on a separate disk spindle. It is more efficient, but more complex, to have it under separate read-write heads, thereby decreasing head contention and seek times.

If the system crashes, there will be zero or more transactions in the journal. Those transactions were never completed to the file system even though they were committed by the operating system, so they must be completed. The transactions can be executed from the pointer until the work is complete, and the file-system structures remain consistent. The only problem occurs when a transaction has been aborted. That is, it was not committed before the system crashed. Any changes from those transactions that were applied to the file system must be undone, again preserving the consistency of the file system. This recovery is all that is needed after a crash, eliminating all problems with consistency checking.

The Linux proc File System

The flexibility of the Linux VFS enables us to implement a file system that does not store data persistently at all but rather simply provides an interface to some other functionality. The Linux **process file system,** known as the /proc file system, is an example of a file system whose contents are not actually stored anywhere but are computed on demand according to user file I/O requests.

A /proc file system is not unique to Linux. SVR4 UNIX introduced a /proc file system as an efficient interface to the kernel's process debugging support:

Each subdirectory of the file system corresponded not to a directory on any disk but rather to an active process on the current system. A listing of the file system reveals one directory per process, with the directory name being the ASCII decimal representation of the process's unique process identifier (PID).

Linux implements such a /proc file system but extends it greatly by adding a number of extra directories and text files under the file system's root directory. These new entries correspond to various statistics about the kernel and the associated loaded drivers. The /proc file system provides a way for programs to access this information as plain text files, which the standard UNIX user environment provides powerful tools to process. For example, in the past, the traditional UNIX ps command for listing the states of all running processes has been implemented as a privileged process that reads the process state directly from the kernel's virtual memory. Under Linux, this command is implemented as an entirely unprivileged program that simply parses and formats the information from /proc. The /proc file system must implement two things: a directory structure and the file contents within.

Pondicherry University Questions

2 MARKS

1.Define Bootstrap Program?(NOV '12) (Ref.Pg.No.1 Qn.No.1)

2.What do you mean by seek time? (APR '12) (Ref.Pg.No.4 Qn.No.18)

3.List the disk performance parameter?(APR '13) (Ref.Pg.No.4 Qn.No.19)

4.What is the purpose of device drivers? (APR '13) (Ref.Pg.No.4 Qn.No.20)

5. List the role of process manager?(NOV '13) (Ref.Pg.No.12 Qn.No.52)

6.What is IPC? (APR '12) (Ref.Pg.No.9 Qn.No.40)

7.Write any two advantages of Linux?(NOV '12) (Ref.Pg.No.10 Qn.No.43)

8.Brief about Process Environment Block (PEB).(NOV '13) (Ref.Pg.No.11 Qn.No.49)

9.Sketch the components of a Linux system (NOV'14) (Ref.Pg.No.16 Qn.No.68)

10.Define Kernel in OS? (APR '12) (APR'15) (NOV '15) (Ref.Pg.No.7 Qn.No.32)

11.What type of security is provided by Windows XP (APR'15) (Ref.Pg.No.14 Qn.No.61)

12. Define physical record and logical record. (NOV '15) (Ref.Pg.No.16 Qn.No.69)

11 Marks:

1. Explain Disk Scheduling in detail with example? (APR '11) (APR '13) (NOV '15) (Ref.Pg.No.19 Qn.No.2)

2. Discuss the various techniques used to improve the efficiency and performance of secondary storage? 6 Marks (APR '14) (Ref.Pg.No.22 Qn.No.3)

3. Briefly explain Disk management and swap space management? (NOV '12) (Ref.Pg.No.23 Qn.No.4)

4. Explain in detail about Kernel architecture of Linux? (APR '13)(NOV '13) (Ref.Pg.No.24 Qn.No.5)

5. Explain in Detail about Process management? (NOV'14) (Ref.Pg.No 27 Qn.No.6)

6. Explain how interprocess communication is carried out in Linux OS? (NOV '12) (APR'13) (APR'15) (Ref.Pg.No.30 Qn.No.8)

7. Explain how the memory and file management is implemented in windows XP? (APR'13) (Ref.Pg.No.33 Qn.No.10)

8. Describe about memory allocation stages. (11)(NOV '13) (Ref.Pg.No.40 Qn.No.13)

9 Explain how the following mechanisms are implemented in Windows XP? (1)Thread (2) IPC (APR '12) (Ref.Pg.No.43 Qn.No.14)

10.Explain in detail about Security? (NOV'14) (Ref.Pg.No.34 Qn.No.11)

11.Explain the allocation methods for disk space in file allocation? (11) (NOV 12, 13) (APR '11)(APR'15) (Ref.Pg.No.17 Qn.No.1)

12.Explain in detail about Windows XP System Architecture?(NOV '12) (APR'15) (Ref.Pg.No.34 Qn.No.12)

13. Discuss on file system structure in the linux systems. (NOV '15) (Ref.Pg.No.51 Qn.No.17)