B.E -COMPUTER SCIENCE AND ENGINEERING 2014-2015

SEMESTER IV

14BECS405 OPERATING SYSTEMS 3H-4C

Instruction Hours/week: L:3 T:0 P:4 **Marks:** Internal**:40** External**:60** Total**:100**

End Semester Exam:3 Hours

COURSE OBJECTIVES:

- To Study the basic concepts and functions of operating systems.
- To understand the structure and functions of OS.
- To Learn about Processes, Threads and Scheduling algorithms.
- To understand the principles of concurrency and Deadlocks.
- To learn various memory management schemes.
- To Study I/O management and File systems.

COURSE OUTCOMES:

Upon completion of this course the student will be able to:

- Understand the different concepts and functions of Operating Systems.
- Design various Scheduling algorithms.
- Apply the principles of concurrency.
- Design deadlock, prevention and avoidance algorithms.
- Compare and contrast various memory management schemes.
- Design and Implement a prototype file systems.

UNIT 1: (9)

Introduction: Concept of Operating Systems, Generations of Operating systems, Types of Operating Systems, OS Services, System Calls, Structure of an OS - Layered, Monolithic, Microkernel Operating Systems, Concept of Virtual Machine. Case study on UNIX and WINDOWS Operating System. **Processes:** Definition, Process Relationship, Different states of a Process, Process State transitions, Process Control Block (PCB), Context switching

Thread: Definition, Various states, Benefits of threads, Types of threads, Concept of multithreads,

UNIT 2: **(9)**

Process Scheduling: Foundation and Scheduling objectives, Types of Schedulers, Scheduling criteria: CPU utilization, Throughput, Turnaround Time, Waiting Time, Response Time; Scheduling algorithms: Pre-emptive and Non pre-emptive, FCFS, SJF, RR; Multiprocessor scheduling: Real Time scheduling: RM and EDF.

Inter-process Communication: Critical Section, Race Conditions, Mutual Exclusion, Hardware Solution,Strict Alternation, Peterson's Solution,The Producer\Consumer Problem, Semaphores, Event Counters, Monitors,Message Passing, Classical IPC Problems: Reader's & Writer Problem, Dinning Philosopher Problem etc.

UNIT 3: **(9)**

Deadlocks: Definition, Necessary and sufficient conditions for Deadlock, Deadlock Prevention, Deadlock Avoidance: Banker's algorithm, Deadlock detection and Recovery.

Memory Management: Basic concept, Logical and Physical address map, Memory allocation: Contiguous Memory allocation – Fixed and variable partition– Internal and External fragmentation and Compaction; Paging: Principle of operation – Page allocation – Hardware support for paging, Protection and sharing, Disadvantages of paging.

UNIT 4: **(9)**

Virtual Memory: Basics of Virtual Memory – Hardware and control structures – Locality of reference, Page fault , Working Set , Dirty page/Dirty bit – Demand paging, Page Replacement algorithms: Optimal, First in First Out (FIFO), Second Chance (SC), Not recently used (NRU) and Least Recently used (LRU).

UNIT 5: **(9)**

I/O Hardware: I/O devices, Device controllers, Direct memory access Principles of I/O Software: Goals of Interrupt handlers, Device drivers, Device independent I/O software, Secondary-Storage Structure: Disk structure, Disk scheduling algorithms

File Management: Concept of File, Access methods, File types, File operation, Directory structure, File ,free space management (bit vector ,linked list ,grouping)directory implementation (linear list ,hash table)efficiency and performance.

Disk Management: Disk structure, Disk scheduling - FCFS, SSTF, SCAN, C-SCAN, Disk reliability, Disk formatting, Boot-block, Bad blocks

Total Hours:45

TEXT BOOKS:

1. D M Dhamdhere, “Operating Systems: A Concept-based Approach”, Second Edition, Tata McGraw-Hill Education, 2007.
2. William Stallings, “Operating Systems: Internals and Design Principles”, Seventh Edition, Prentice Hall, 2011.

REFERENCES:

1. Abraham Silberschatz, Peter B. Galvin, Greg Gagne, “Operating System Concepts Essentials”, John Wiley & Sons Inc., 2010.
2. D M Dhamdhere, “Operating Systems: A Concept-Based Approach”, Second Edition, Tata McGraw-Hill Education, 2007.
3. Charles Crowley, “Operating Systems: A Design-Oriented Approach”, Tata McGraw Hill Education”, 1996.

WEBSITE:

1. http://nptel.ac.in/.

KARPAGAM ACADEMY OF HIGHER EDUCATION

Faculty of Engineering

Department of Computer Science and Engineering

Lecture Plan

Faculty Name	**S.Shahul Hammed**	**Subject Code**	**: 14BECS405**
Subject Name	**Operating Systems**	**Class**	**: III B.E CSE B Section**

S.No	Topic Name	No. of Periods	Teaching Aids	Books	Page No. of Text Book
Unit - I Introduction					
1	Introduction to OS concepts	1	BB/PPT	T1	
2	OS Structures ,Kernal and Shell	1	BB/PPT	T1	Pg.No. 55
3	Evolution of operating systems -Mainframes systems-Desktops systems-multiprocessor systems	1	BB/PPT	T1	Pg.No. 7-12
4	Distributed systems-Clustered systems-Real time systems-Handheld systems	1	BB/PPT	T1	Pg.No. 12-19
5	Hardware protection-System Components-Operating System services	1	BB/PPT	T1	Pg.No. 42
6	System Calls-System Programs-Process concepts	1	BB/PPT	T1	Pg.No. 63
7	Process Scheduling	1	BB/PPT	T1	Pg.No. 99
8	Operations on Processes	1	BB/PPT	T1	Pg.No. 103
9	cooperating Processes	1	BB/PPT	T1	Pg.No. 107
10	Interprocess communication	1	BB/PPT	T1	Pg.No. 109
11	Tutorial 1: Interprocess Communication	1	BB/PPT	T1	Pg.No. 109
	Total	**11**			
Unit - II Scheduling					
12	Threads-Overview	1	BB/PPT	T1	Pg.No.

					129
13	Threading Issues-CPU Scheduling	1	BB/ PPT	T1	Pg.No. 135
14	Basic concepts-Scheduling Criteria	1	BB/ PPT	T1	Pg.No. 155
15	Scheduling Algorithms	1	BB/ PPT	T1	Pg.No. 157
16	Multiple Processor Scheduling-Real time scheduling	1	BB/ PPT	T1	Pg.No. 169
17	The critical section problem	1	BB/ PPT	T1	Pg.No. 191
18	Synchronisation Hardware	1	BB/ PPT	T1	Pg.No. 197
19	Semaphores	1	BB/ PPT	T1	Pg.No. 201
20	Classic problems of sychronisation	1	BB/ PPT	T1	Pg.No. 206
21	Critical Regions	1	BB/ PPT	T1	Pg.No. 211
22	Monitors	1	BB/ PPT	T1	Pg.No. 216
	Total	**11**			
Unit - III Deadlocks					
23	System Model-Deadlock Characterization	1	BB/ PPT	T1	Pg.No243-245
24	Methods of Handling Deadlocks	1	BB/ PPT	T1	Pg.No-248
25	Deadlock Prevention	1	BB/ PPT	T1	Pg.No-250
26	Deadlock Avoidance	1	BB/ PPT	T1	Pg.No-253
27	Deadlock detection-Recovery from deadlocks	1	BB/ PPT	T1	Pg.No-260-264
28	Storage Management-Swapping	1	BB/ PPT	T1	Pg.No-280
29	Contiguous Memory Allocation	1	BB/ PPT	T1	Pg.No-283
30	Paging-Segmentation	1	BB/ PPT	T1	Pg.No-287-303
31	Segmentation and paging	1	BB/ PPT	T1	Pg.No-309

	Total	9			
Unit - IV Virtual Memory					
32	Virtual Memory	1	BB/ PPT	T1	Pg.No-317
33	Demand Paging	1	BB/ PPT	T1	Pg.No-320
34	Process Creation	1	BB/ PPT	T1	Pg.No-328
35	Page Replacement	1	BB/ PPT	T1	Pg.No-330
36	Allocation of Frames	1	BB/ PPT	T1	Pg.No-344
37	Thrasing	1	BB/ PPT	R3	Pg.No-348
38	File concept-Access Methods	1	BB/ PPT	T1	Pg.No-371
39	Directory Sturcture	1	BB/ PPT	R3	Pg.No-383
40	File sharing	1	BB/ PPT	T1	Pg.No-395
41	Protection	1			
	Total	**10**			
Unit - V File Systems					
42	File system Structure	1	BB/ PPT	T1	Pg.No-411
43	File system Implementation	1	BB/ PPT	T1	Pg.No-413
44	Directory Implementation	1	BB/ PPT	T1	Pg.No-420
45	Allocation methods-Free space management	1	BB/ PPT	T1	Pg.No-421
46	Kernal I/O subsystems	1	BB/ PPT	R2	Pg.No-472
47	Disk Sructure-Disk Scheduling	1	BB/ PPT	T1	Pg.No-491
48	Disk management-Swap space management	1	BB/ PPT	T1	Pg.No-498
49	Case study :The Linux system	1	BB/ PPT	w1	
50	Windows 2000	1	BB/ PPT	T1	Pg.No-743
51	Seminar-Introduction -UNIX	1		PPT	
	Total	**10**			
	Total Hours	**51**			

Hours Allocated

Number of hours allocated for Lecture 45

Number of hours planned for Lecture 51

Text Books:

T1: Abraham Silberschatz,Peter Baer Galvin and Greg Gagne"Operating systems concepts"John WILEY &Sons(ASIA) Pvt.Ltd,2009

References:

R1 Harvey M. Deitel Operating Systems Pearson Education Pvt.

R2 Andrew S Tanenbaum"Modern operating Systems",Prentice Hall of India Pvt Limited

R3 William Stallings "Operating systems" Prentice Hall of India 2009

LECTURE NOTES

UNIT- I INTRODUCTION

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

1.1 Introduction :

What is an Operating System?

• An operating system is a program that manages the computer hardware.

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

• The purpose of an operating system is to provide an environment in which a user can execute programs.

Goals of an Operating System

• The primary goal of an operating system is thus to make the computer system convenient to use.

• The secondary goal is to use the computer hardware in an efficient manner.

Components of a Computer System

• An operating system is an important part of almost every computer system.

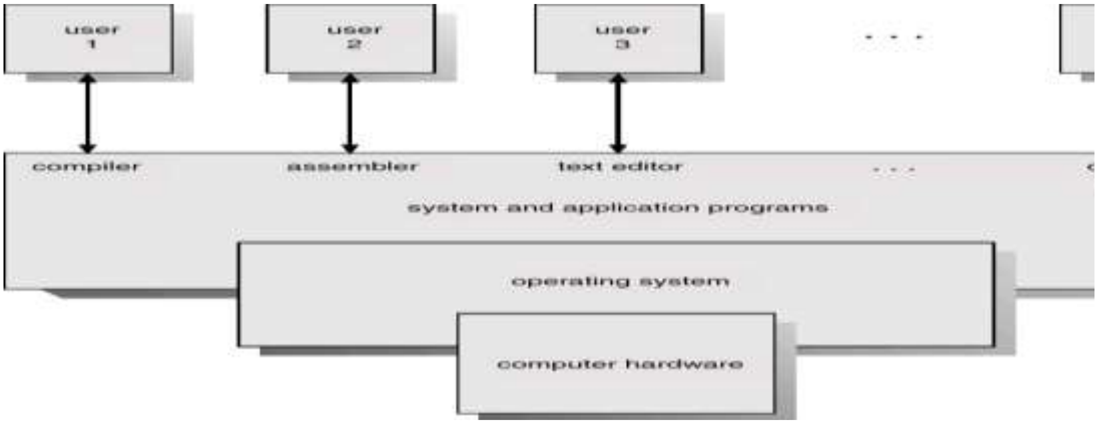
• A computer system can be divided roughly into four components.

i. Hardware

ii. Operating system

iii. The application programs

iv. Users



• The hardware - the central processing unit **(CPU),** the memory, and the Input/output **(I/O*)*** devices-provides the basic computing resources.

• The application programs- such as word processors, spreadsheets, compilers, and web browsers- define the ways in which these resources are used to solve the computing problems of the users.

• An operating system is similar to a *government. The* OS simply provides an environment within which other programs can do useful work.

Abstract view of the components of a computer system.

• Operating system can be viewed as a resource allocator.

• The OS acts as the manager of the resources (such as CPU time, memory space, file storage space, I/O devices) and allocates them to specific programs and users as necessary for tasks.

• An operating system is a control program. It controls the execution of user programs to prevent errors and improper use of computer.

1.2 Mainframe Systems

• Early computers were physically enormous machines run from a console.

• The common input devices were card readers and tape drives.

• The common output devices were line printers, tape drives, and card punches.

• The user did not interact directly with the computer systems.

• Rather, the user prepared a job - which consisted of the program, the data, and some control information about the nature of the job (control cards)-and submitted it to the computer operator.

• The job was usually in the form of punch cards.

• The operating system in these early computers was fairly simple.

• Its major task was to transfer control automatically from one job to the next.

• The operating system was always resident in memory

Memory layout for a simple batch system.

A batch operating system, thus normally reads a stream of separate jobs.

• When the job is complete its output is usually printed on a line printer.

• The definitive feature of batch system is the lack of interaction between the user and the job while the job is executing.

• Spooling is also used for processing data at remote sites.

<u>Multiprogrammed Systems</u>

• A pool of jobs on disk allows the OS to select which job to run next, to increase CPU utilization.

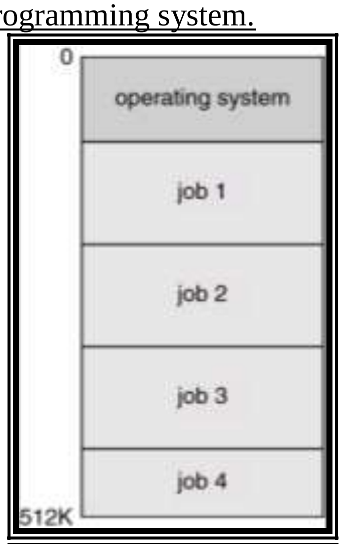
• Multiprogramming increases CPU utilization by organizing jobs such that the CPU always has one to execute.

• The idea is as follows: The operating system keeps several jobs in memory simultaneously. This set of jobs is a subset of the jobs kept in the job pool.

The operating system picks and begins to execute one of the jobs in the memory.

The operating system picks and begins to execute one of the jobs in the memory.

<u>Memory layout for a multiprogramming system.</u>



<u>Time-Sharing Systems</u>

• Time sharing (or multitasking) is a logical extension of multiprogramming. 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.

• A time-shared operating system allows many users to share the computer simultaneously. Since each action or command in a time-shared system tends to be short, only a little CPU time is needed for each user. As the system switches rapidly from one user to the next, each user is given the impression that the entire computer system is dedicated to her use, even though it is being shared among many users.

1.3 Desktop Systems

• As hardware costs have decreased, it has once again become feasible to have a computer system dedicated to a single user. These types of computer systems are usually referred to as personal computers(PCS). They are
microcomputers that are smaller and less expensive than mainframe computers.

• Operating systems for these computers have benefited from the development of operating systems for mainframes in several ways.

1.4 Multiprocessor Systems

• **Multiprocessor systems** (also known as **parallel systems** or **tightly coupled systems)** have more than one processor in close communication, sharing the computer bus, the clock, and sometimes memory and peripheral devices.

• Multiprocessor systems have three main advantages.

o **Increased throughput.**

o **Economy of scale.**

o **Increased reliablility.**

• **If** functions can be distributed properly among several processors, then the failure of one processor will not halt the system, only slow it down. If we have ten processors and one fails, then each of the remaining nine processors must pick up a share of the work of the failed processor. Thus, the entire system runs only 10 percent slower, rather than failing altogether. This ability to continue providing service proportional to the level of surviving hardware is called **graceful degradation.** Systems designed for graceful degradation are also called **fault tolerant.**

• Continued operation in the presence of failures requires a mechanism to allow the failure to be detected, diagnosed, and, if possible, corrected.

• The most common multiple-processor systems now use **symmetric multiprocessing** (SMP), in whch each processor runs an identical copy of the operating system, and these copies communicate with one another as needed.

• Some systems use **asymmetric multiprocessing,** in which each processor is assigned a specific task. A master processor controls the system; the other processors either look to the master for instruction or have predefined tasks. This scheme defines a master-slave relationship. The master processor schedules and allocates work to the slave processors.

1.5 Distributed Systems

• In contrast to the tightly coupled systems, the processors do not share memory or a clock. Instead , each processor has its own local memory.

• The processors communicate with one another through various communication lines, such as high speed buses or telephone lines. These systems are usually referred to as loosely coupled systems, or distributed systems.

Advantages of distributed systems

• Resource Sharing

• Computation speedup

• Reliability

• Communication

1.6 Clustered Systems

- Clustering allows two or more systems to share storage.
- Provides high reliability.
- *Asymmetric clustering*: one server runs the application while other servers standby.
- *Symmetric clustering*: all N hosts are running the application.

1.7 Real-Time Systems

• Systems that control scientific experiments, medical imaging systems, industrial control systems, and certain display systems are real-time systems. Some automobile-engine fuel-injection systems, home-appliance controllers, and weapon systems are also real-time systems. A real-time system has well-defined, fixed time constraints.

• Real-time systems come in two flavors: hard and soft.

• A hard real-time system guarantees that critical tasks be completed on time. This goal requires that all delays in the system be bounded, from the retrieval of stored data to the time that it takes the operating system to finish any request made of it. Such time constraints dictate the facilities that are available in hard real-time systems.

• A less restrictive type of real-time system is a soft real-time system, where a critical real-time task gets priority over other tasks, and retains that priority until it completes.

• Soft real-time systems, however, have more limited utility than hard real-time systems. They are useful, in several areas, including multimedia, virtual reality, and advanced scientific projects.

1.8 Handheld Systems

- Personal Digital Assistants (PDAs)
- Cellular telephones
- Issues:
  - Limited memory
  - Slow processors
  - Small display screens.

<u>1.9 Hardware Protection</u>

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

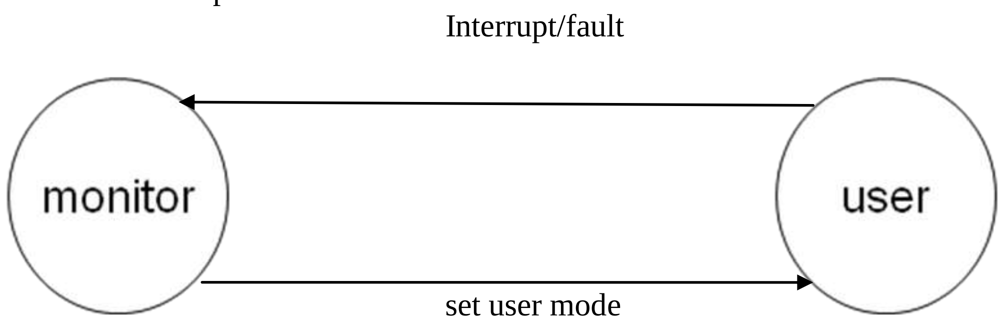
Dual-Mode Operation

- 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.

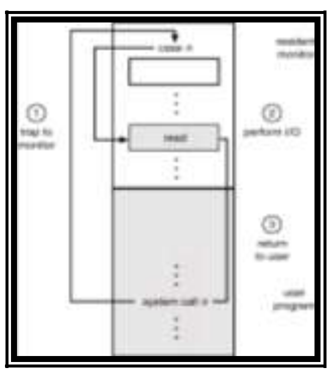
- *Mode bit* added to computer hardware to indicate the current mode: monitor (0) or user (1).

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



Privileged instructions can be issued only in monitor mode.

Use of A System Call to Perform I/O

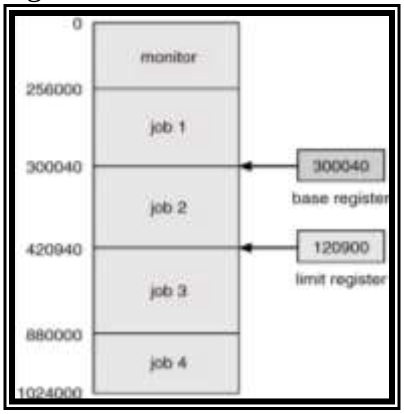


Memory Protection

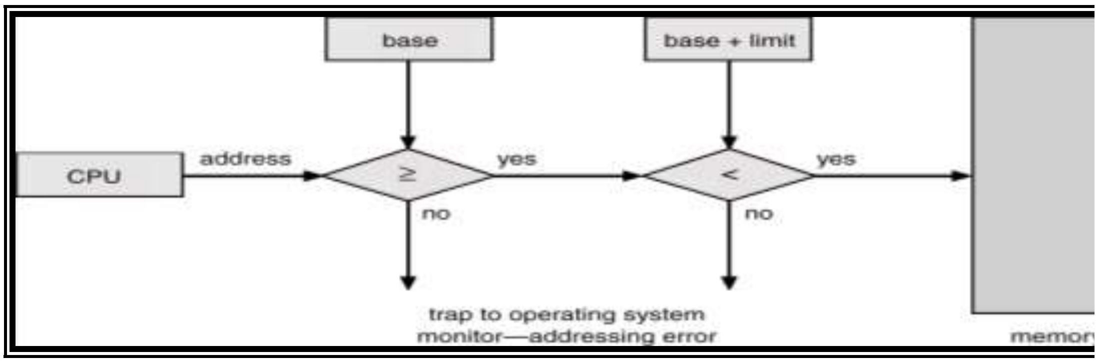
- Must provide memory protection at least for the interrupt vector and the interrupt service routines.
- In order to have memory protection, add two registers that determine the range of legal addresses a program may access:

- Base register – holds the smallest legal physical memory address.
- Limit register – contains the size of the range
- Memory outside the defined range is protected.

Use of A Base and Limit Register



Hardware Address 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
- *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.
- Time also used to compute the current time.
- Load-timer is a privileged instruction.

1.10 System Components

There are eight major operating system components.They are :

- Process management
- Main-memory management
- File management
- I/O-system management

- Secondary-storage management
- Networking
- Protection system
- Command-interpreter system

(i) Process Management

• A **process** can be thought of as a program in execution. A batch job is a process. A time shared user program is a process.

• A process needs certain resources-including CPU time, memory, files, and I/O devices-to accomplish its task.

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

• A process is the unit of work in a system.

• The operating system is responsible for the following activities in connection with process management:

• Creating and deleting both user and system processes

• Suspending and resuming processes

• Providing mechanisms for process synchronization

• Providing mechanisms for process communication

• Providing mechanisms for deadlock handling

(ii) Main – Memory Management

• Main memory is a large array of words or bytes, ranging in size from hundreds of thousands to billions. Each word or byte has its own address.

• Main memory is a repository of quickly accessible data shared by the CPU and I/O devices.

• To improve both the utilization of the CPU and the speed of the computer's response to its users, we must keep several programs in memory.

• The operating system is responsible for the following activities in connection with memory management:

• Keeping track of which parts of memory are currently being used and by whom.

• Deciding which processes are to be loaded into memory when memory space becomes available.

• Allocating and deallocating memory space as needed.

(iii) File Management

• File management is one of the most visible components of an operating system.

• The operating system is responsible for the following activities in connection with file management:

• Creating and deleting files

• Creating and deleting directories

- Supporting primitives for manipulating files and directories
- Mapping files onto secondary storage
- Backing up files on stable (nonvolatile) storage media

(iv) I/O System management

- One of the purposes of an operating system is to hide the peculiarities of specific hardware devices from the user. This is done using the I/O subsystem.
- The I/O subsystem consists of
- A memory-management component that includes buffering, caching, and spooling
- A general device-driver interface
- Drivers for specific hardware devices

(v) Secondary storage management

- Because main memory is too small to accommodate all data and programs, and because the data that it holds are lost when power is lost,the computer system must provide **secondary storage** to back up main memory.
- The operating system is responsible for the following activities in connection with disk management:
- Free-space management
- Storage allocation
- Disk scheduling

(vi) Networking

- **A distributed system** is a collection of processors that do not share memory, peripheral devices, or a clock.
- Instead, each processor has its own local memory and clock, and the processors communicate with one another through various communication lines, such as high-speed buses or networks.
- The processors in the system are connected through a **communication network,** which can be configured in a number of different ways.

(vii) Protection System

- Various processes must be protected from one another's activities. For that purpose, mechanisms ensure that the files, memory segments, CPU, and other resources can be operated on by only those processes that have gained proper authorization from the operating system.
- Protection is any mechanism for controlling the access of programs, processes, or users to the resources defined by a computer system.
- Protection can improve reliability by detecting latent errors at the interfaces between component subsystems.

(viii) Command-Interpreter System

- One of the most important systems programs for an operating system is the command interpreter.
- It is the interface between the user and the operating system.
- Some operating systems include the command interpreter in the kernel. Other operating systems, such as MS-DOS and UNIX, treat the command interpreter as a special program that is running when a job is initiated, or when a user first logs on (on time-sharing systems).
- Many commands are given to the operating system by control statements.
- When a new job is started in a batch system, or when a user logs on to a time-shared system, a program that reads and interprets control statements is executed automatically.
- This program is sometimes called the **control-card interpreter** or the **command-line interpreter**, and is often known as the **shell**.

1.11 Operating-System Services

The OS provides certain services to programs and to the users of those programs.

1. **Program execution:** The system must be able to load a program into memory and to run that program. The program must be able to end its execution, either normally or abnormally (indicating error).

2. **I/O operations:** A running program may require I/O. This I/O may involve a file or an I/O device.

3. **File-system manipulation:** The program needs to read, write, create, delete files.

4. **Communications :** In many circumstances, one process needs to exchange information with another process. Such communication can occur in two major ways. The first takes place between processes that are executing

on the same computer; the second takes place between processes that are executing on different computer systems that are tied together by a computer network.

5. **Error detection:** The operating system constantly needs to be aware of possible errors. Errors may occur in the CPU and memory hardware (such as a memory error or a power failure), in I/O devices (such as a parity error on tape, a connection failure on a network, or lack of paper in the printer), and in the user program (such as an arithmetic overflow, an attempt to access an illegal memory location, or a too-great use of CPU time). For each type of error, the operating system should take the appropriate action to ensure correct and consistent computing.

6. **Resource allocation:** Different types of resources are managed by the Os. When there are multiple users or multiple jobs running at the same time, resources must be allocated to each of them.

7. **Accounting:** We want to keep track of which users use how many and which kinds of computer resources. This record keeping may be used for accounting or simply for accumulating usage statistics.

8. **Protection:** The owners of information stored in a multiuser computer system may want to control use of that information. Security of the system is also important.

1.12 System Calls

- System calls provide the interface between a process and the operating system.
- These calls are generally available as assembly-language instructions.
- System calls can be grouped roughly into five major categories:

1. Process control
2. file management
3. device management
4. information maintenance
5. Communications

Process Control

- end,abort
- load, execute
- Create process and terminate process
- get process attributes and 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 attribtues, 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

Two types of communication models
(a) Message passing model
(b) Shared memory model

1.13 System Programs

• System programs provide a convenient environment for program development and execution.

• They can be divided into several categories:

1. **File management:** These programs create, delete, copy, rename, print, dump, list, and generally manipulate files and directories.

2. **Status information:** The status such as date, time, amount of available memory or diskspace, number of users or similar status information.

3. **File modification:** Several text editors may be available to create and modify the content of files stored on disk or tape.

4. **Programming-language support:** Compilers, assemblers, and interpreters for common programming languages are often provided to the user with the operating system.

5. **Program loading and execution:** The system may provide absolute loaders, relocatable loaders, linkage editors, and overlay loaders.

6. **Communications:** These programs provide the mechanism for creating virtual connections among processes, users, and different computer systems. (email, FTP, Remote log in)

7. **Application programs:** Programs that are useful to solve common problems, or to perform common operations.

Eg. Web browsers, database systems.

1.14 Process Concept

• A process can be thought of as a program in execution.

• A process is the unit of the unit of work in a modern time-sharing system.

• A process generally includesthe process stack, which contains temporary data (such as method parameters,return addresses, and local variables), and a data section, which contains global variables.

Difference between program and process

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

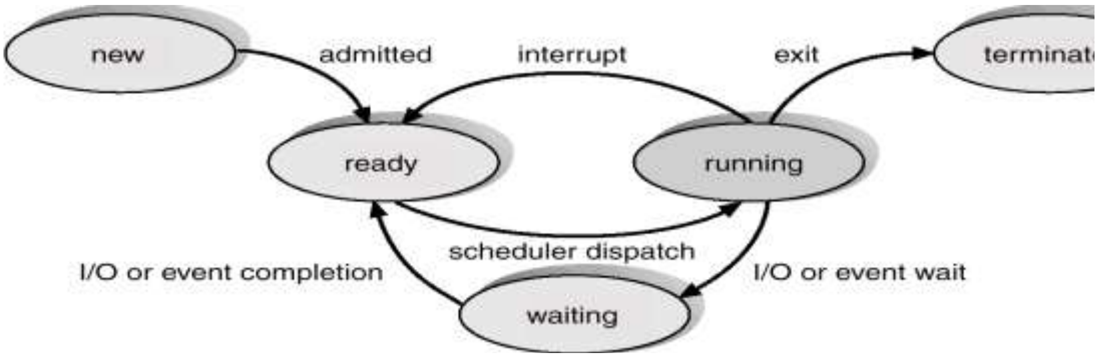
Process States:

• 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**: The process is being created.
- **Running**: Instructions are being executed.
- **Waiting**: The process is waiting for some event to occur (such as an I/O completion or reception of a signal).
- **Ready**: The process is waiting to be assigned to a processor.
- **Terminated**: The process has finished execution.



Process Control Block

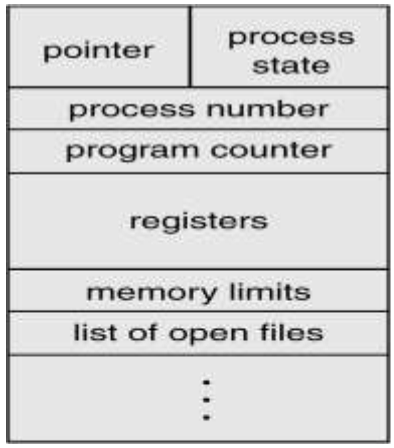
• Each process is represented in the operating system by a process control block (PCB)-also called a task control block.

• A PCB defines a process to the operating system.

• It contains the entire information about a process.

• Some of the information a PCB contans are:

- **Process state**: The state may be new, ready, running, waiting, halted, and SO on.
- **Program counter**: The counter indicates the address of the next instruction to be executed for this process.
- **CPU registers**: The registers vary in number and type, depending on the computer architecture.
- **CPU-scheduling information**: This information includes a process priority, pointers to scheduling queues, and any other scheduling parameters.
- **Memory-management information**: This information may include such information as the value of the base and limit registers, the page tables, or the segment tables, depending on the memory system used by the operating system.
- **Accounting information**: This information includes the amount of CPU and real time used, time limits, account numbers, job or process numbers, and so on.
- **Status information**: The information includes the list of I/O devices allocated to this process, a list of open files, and so on.



1.15 Process Scheduling

• The objective of multiprogramming is to have some process running at all times, so as to maximize CPU utilization.

Scheduling Queues

There are 3 types of scheduling queues .They are :

1. Job Queue
2. Ready Queue
3. Device Queue

• As processes enter the system, they are put into a **job queue.**

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

• The list of processes waiting for an I/O device is kept in a **device queue** for that particular device.

• A new process is initially put in the ready queue. It waits in the ready queue until it is selected for execution (or dispatched).

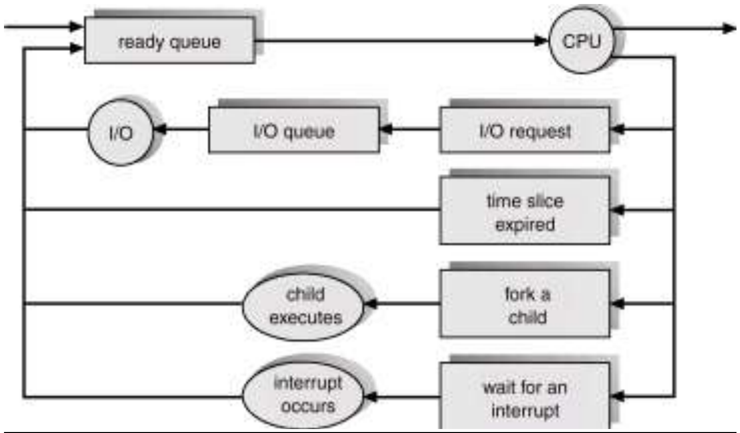
• Once the process is assigned tothe CPU and is executing, one of several events could occur:

• The process could issue anI/O request, and then be placed in an I/O queue.

• The process could create a new subprocess and wait for its termination.

• The process could be removed forcibly from the CPU, as a result of aninterrupt, and be put back in the ready Queue.

• A common representation of process scheduling is a queueing diagram.



Schedulers

• A process migrates between the various scheduling queues throughout its lifetime.

• The operating system must select, for scheduling purposes, processes from these queues in some fashion.

• The selection process is carried out by the appropriate scheduler.

There are three different types of schedulers.They are:
1. Long-term Scheduler or Job Scheduler
2. Short-term Scheduler or CPU Scheduler
3. Medium term Scheduler

• The **long-term scheduler**, or **job scheduler**, selects processes from this pool and loads them into memory for execution. It is invoked very infrequently.It controls the degree of multiprogramming.

• The **short-term scheduler**, or **CPU scheduler**, selects from among the processes that are ready to execute, and allocates the CPU to one of them. It is invoked very frequently.

• Processes can be described as either **I/O bound** or **CPU bound**.

• An **I\O-bound process** spends more of its time doing I/O than it spends doing computations.

• A **CPU-bound process**, on the other hand, generates I/O requests infrequently,using more of its time doing computation than an I/O-bound process uses.

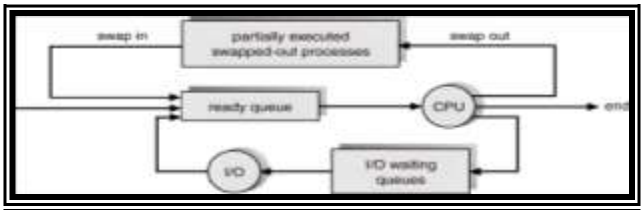
• The system with the best performance will have a combination of CPU-bound and I/O-bound processes.

Medium term Scheduler

• Some operating systems, such as time-sharing systems, may introduce an additional, intermediate level of scheduling.

• The key idea is medium-term scheduler, removes processes from memory and thus reduces the degree of multiprogramming.

• At some later time, the process can be reintroduced into memory and its execution can be continued where it left off. This scheme is called swapping.



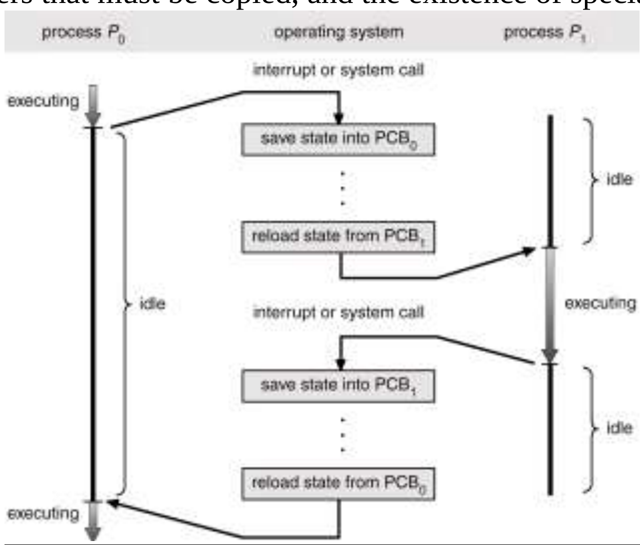
Context Switch

• 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 a context switch.

• Context-switch time is pure overhead, because the system does no useful work while switching.

• Its speed varies from machine to machine, depending on the memory speed, the number of registers that must be copied, and the existence of special instructions.



1.16 Operations on Processes

1. Process Creation

• A process may create several new processes, during the course of execution.

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

• When a process creates a new process, two possibilities exist **in terms of execution:**

1. The parent continues to execute concurrently with its children.
2. The parent waits until some or all of its children have terminated.

• There are also two possibilities **in terms of the address space** of the new process:

1. The child process is a duplicate of the parent process.
2. The child process has a program loaded into it.

• In UNIX, each process is identified by its process identifier, which is a unique integer. A new process is created by the **fork** system call.

2. Process Termination

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

• At that point, the process may return data (output) to its parent process (via the **wait** system call).

• A process can cause the termination of another process via an appropriate system call.

• A parent may terminate the execution of one of its children for a variety of

reasons, such as these:

1. The child has exceeded its usage of some of the resources that it has been allocated.
2. The task assigned to the child is no longer required.
3. The parent is exiting, and the operating system does not allow a child to continue if its parent terminates. On such systems, if a process terminates (either normally or abnormally), then all its children must also be terminated. This phenomenon, referred to as **cascading termination**, is normally initiated by the operating system.

1.17 Cooperating Processes

• The concurrent processes executing in the operating system may be either **independent** processes or **cooperating** processes.

• A process is independent if it cannot affect or be affected by the other processes executing in the system.

• A process is cooperating if it can affect or be affected by the other processes executing in the system.

• Benefits of Cooperating Processes

1. Information sharing
2. Computation speedup
3. Modularity
4. Convenience

Example

Producer – Consumer Problem

• A producer process produces information that is consumed by a consumer process.

• For example, a print program produces characters that are consumed by the printer driver. A compiler may produce assembly code, which is consumed by an assembler.

• To allow producer and consumer processes to run concurrently, we must have available a buffer of items that can be filled by the producer and emptied by the consumer.

o **unbounded-buffer**: places no practical limit on the size of the buffer.

o **bounded-buffer** : assumes that there is a fixed buffer size.

Shared data

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

The shared buffer is implemented as a circular array with two logical pointers: **in** and **out.** The variable **in** points to the next free position in the buffer; **out** points to the first full position in the buffer. The buffer is empty when **in** == **out** ; the buffer is full when **((in** + 1) % **BUFFERSIZE)** == **out.**

Producer Process

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

Consumer process

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

<u>1.18 Interprocess Communication</u>

• Operating systems provide the means for cooperating processes to communicate with each other via an interprocess communication (PC) facility.

• IPC provides a mechanism to allow processes to communicate and to synchronize their actions.IPC is best provided by a message passing system.

Basic Structure:

• If processes P and Q want to communicate, they must send messages to and receive messages from each other; a communication link must exist between them.

• Physical implementation of the link is done through a hardware bus , network etc,

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

1. Direct or indirect communication
2. Symmetric or asymmetric communication
3. Automatic or explicit buffering
4. Send by copy or send by reference

5. Fixed-sized or variable-sized messages

Naming

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

1. **Direct Communication**

• Each process that wants to communicate must explicitly name the recipient or sender of the communication.

• A communication link in this scheme has the following properties:

i. A link is established automatically between every pair of processes that want to communicate. The processes need to know only each other's identity to communicate.

ii. A link is associated with exactly two processes.

iii. Exactly one link exists between each pair of processes.

• There are two ways of addressing namely

  Symmetry in addressing

  Asymmetry in addressing

• In symmetry in addressing, the send and receive primitives are defined as:

send(P, message) Send a message to process P
receive(Q, message) Receive a message from Q

• In asymmetry in addressing , the send & receive primitives are defined as:

send (p, message)   send a message to process p
receive(id, message)   receive message from any process, id is set to the name of the process with which communication has taken place

2. Indirect Communication

• With indirect communication, the messages are sent to and received from mailboxes, or ports.

• The send and receive primitives are defined as follows:

send (A, message) Send a message to mailbox A.
receive (A, message) Receive a message frommailbox A.

• A communication link has the following properties:

i. A link is established between a pair of processes only if both members of the pair have a shared mailbox.

ii. A link may be associated with more than two processes.

iii. A number of different links may exist between each pair of communicating processes, with each link corresponding to one mailbox

3. Buffering

• A link has some capacity that determines the number of message that can reside in it temporarily. This property can be viewed as a queue of messages attached to the link.

• There are three ways that such a queue can be implemented.

• **Zero capacity** : Queue length of maximum is 0. No message is waiting in a queue. The sender must wait until the recipient receives the message. (message system with no buffering)

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

• **Unbounded capacity**: The queue has potentially infinite length. Thus any number of messages can wait in it. The sender is never delayed

4. Synchronization

• Message passing may be either blocking or non-blocking.

1. **Blocking Send** - The sender blocks itself till the message sent by it is received by the receiver.

2. **Non-blocking Send** - The sender does not block itself after sending the message but continues with its normal operation.

3. **Blocking Receive** - The receiver blocks itself until it receives the message.

4. **Non-blocking Receive** – The receiver does not block itself.

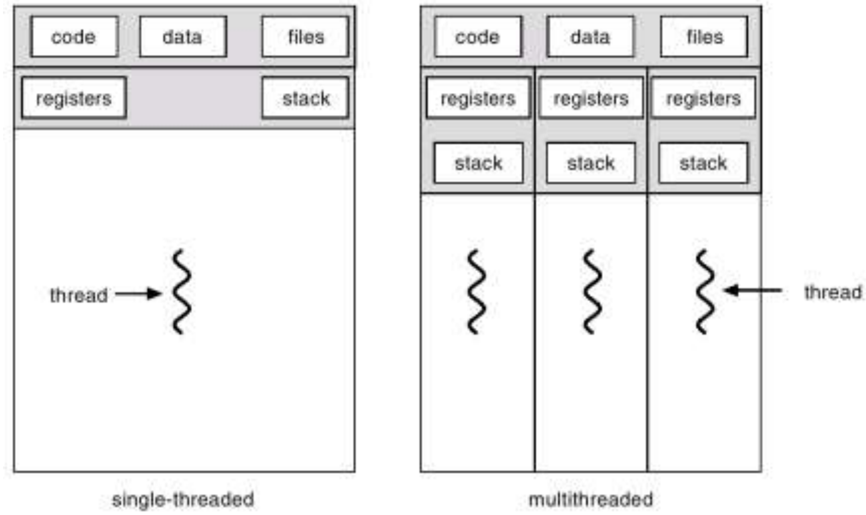
UNIT -II SCHEDULING

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.

2.1 Threads Overview

• A thread is the **basic unit of CPU utilization**.

• It is sometimes called as a **lightweight process**.
• It consists 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 resources such as open files and signals.



A traditional or heavy weight process has a single thread of control.
• If the process has multiple threads of control,it can do more than one task at a time.

Benefits of multithreaded programming

• Responsiveness
• Resource Sharing
• Economy
• Utilization of MP Architectures

User thread and Kernel threads

User threads

• Supported above the kernel and implemented by a thread library at the user level.

• Thread creation , management and scheduling are done in user space.

• Fast to create and manage

• When a user thread performs a blocking system call ,it will cause the entire process to block even if other threads are available to run within the application.

• Example: POSIX Pthreads,Mach C-threads and Solaris 2 UI-threads.

Kernel threads

• Supported directly by the OS.

• Thread creation , management and scheduling are done in kernel space.

• Slow to create and manage

• When a kernel thread performs a blocking system call ,the kernel schedules another thread in the application for execution.

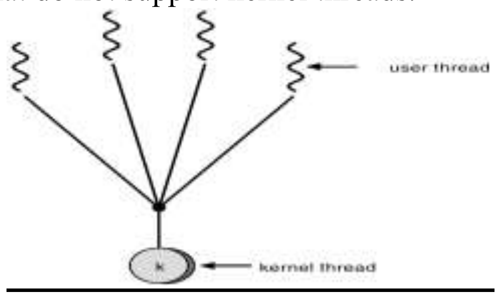
• Example: Windows NT, Windows 2000 , Solaris 2,BeOS and Tru64 UNIX support kernel threads.

Multithreading models

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

1. Many-to-One:

• Many user-level threads mapped to single kernel thread.
• Used on systems that do not support kernel threads.

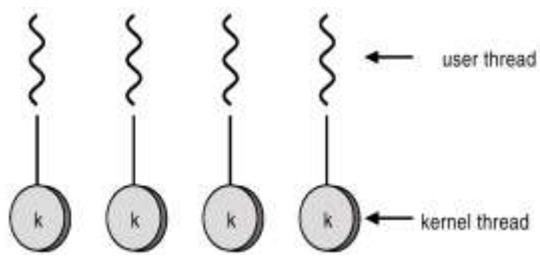


2.One-to-One:

• Each user-level thread maps to a kernel thread.
• Examples

- Windows 95/98/NT/2000
- OS/2

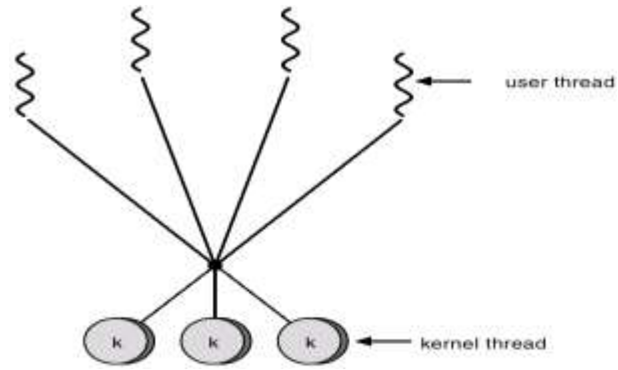
One-to-one Model



3.Many-to-Many Model:

• Allows many user level threads to be mapped to many kernel threads.
• Allows the operating system to create a sufficient number of kernel threads.
• Solaris 2
• Windows NT/2000

Many-to-Many Model



<u>2.2 Threading Issues:</u>

1. fork() and exec() system calls.

A fork() system call may duplicate allthreads or duplicate only the thread that invoked fork().
If a thread invoke exec() system call ,the program specified in the parameter to exec will replace the entire process.

2. Thread cancellation.

It is the task of terminating a thread before it has completed .
A thread that is to be cancelled is called a target thread.
There are two types of cancellation namely
1. Asynchronous Cancellation – One thread immediately terminates the target thread.

2. Deferred Cancellation – The target thread can periodically check if it should terminate , and does so in an orderly fashion.

3. Signal handling

1. A signal is a used to notify a process that a particular event has occurred.

2. A generated signal is delivered to the process.
a. Deliver the signal to the thread to which the signal applies.

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

c. Deliver the signal to certain threads in the process.

d. Assign a specific thread to receive all signals for the process.
3. Once delivered the signal must be handled.
a. Signal is handled by
i. A default signal handler

ii. A user defined signal handler

4. Thread pools

Creation of unlimited threads exhaust system resources such as CPU time or memory. Hence we use a thread pool.
In a thread pool , a number of threads are created at process startup and placed in the pool. When there is a need for a thread the process will pick a thread from the pool and assign it a task.
After completion of the task,the thread is returned to the pool.

5. Thread specific data

Threads belonging to a process share the data of the process. However each thread might need its own copy of certain data known as thread-specific data

2.3 **CPU Scheduling**

2.4.Basic Concepts

• CPU scheduling is the basis of multi programmed operating systems.

• The objective of multiprogramming is to have some process running at all times, in order to maximize CPU utilization.

• Scheduling is a fundamental operating-system function.

• Almost all computer resources are scheduled before use.

CPU-I/O Burst Cycle

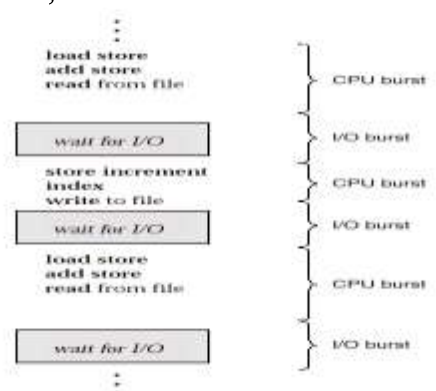
• Process execution consists of a **cycle** of CPU execution and I/O wait.

• Processes alternate between these two states.

• Process execution begins with a **CPU burst.**

• That is followed by an I/O **burst,** then another CPU burst, then another I/O burst, and so on.

• Eventually, the last CPU burst will end with a system request to terminate execution, rather than with another I/O burst.



CPU Scheduler

• Whenever the CPU becomes idle, the operating system must select one of the processes in the ready queue to be executed.

• The selection process is carried out by the **short-term scheduler** (or CPU scheduler).

• The ready queue is not necessarily a first-in, first-out (FIFO) queue. It may be a FIFO queue, a priority queue, a tree, or simply an unordered linked list.

Preemptive Scheduling

• CPU scheduling decisions may take place under the following four circumstances:

1. When a process switches from the running state to the waiting state
2. When a process switches from the running state to the ready state
3. When a process switches from the waiting state to the ready state
4. When a process terminates

• Under 1 & 4 scheduling scheme is non preemptive.

• Otherwise the scheduling scheme is preemptive.

Non-preemptive Scheduling

• In non preemptive scheduling, once the CPU has been allocated a process, the process keeps the CPU until it releases the CPU either by termination or by switching to the waiting state.

• This scheduling method is used by the Microsoft windows environment.

Dispatcher

• The dispatcher is the module that gives control of the CPU to the process selected by the short-term scheduler.

• This function involves:

1. Switching context
2. Switching to user mode
3. Jumping to the proper location in the user program to restart that program

2.5 Scheduling Criteria

1. CPU utilization: The CPU should be kept as busy as possible. CPU utilization may range from 0 to 100 percent. In a real system, it should range from 40 percent (for a lightly loaded system) to 90 percent (for a heavily used system).

2. Throughput: Itis the number of processes completed per time unit. For long processes, this rate may be 1 process per hour; for short transactions, throughput might be 10 processes per second.

3. Turnaround time: The interval from the time of submission of a process to the time of completion is the turnaround time. Turnaround time 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.

4. Waiting time: Waiting time is the sum of the periods spent waiting in the ready queue.

5. Response time: It is the amount of time it takes to start responding, but not the time that it takes to output that response.

2.6 CPU Scheduling Algorithms

1. First-Come, First-Served Scheduling
2. Shortest Job First Scheduling
3. Priority Scheduling
4. Round Robin Scheduling

First-Come, First-Served Scheduling

- The process that requests the CPU first is allocated the CPU first.
- It is a non-preemptive Scheduling technique.
- The implementation of the FCFS policy is easily managed with a FIFO queue.

Example:

Process	Burst Time
P1	24
P2	3
P3	3

- If the processes arrive in the order PI, P2, P3, and are served in FCFS order, we get the result shown in the following Gantt chart:

Gantt Chart

P1	P2	P3

0 24 27 30

Average waiting time = (0+24+27) / 3 = 17 ms
Average Turnaround time = (24+27+30) / 3 = 27 ms

- The FCFS algorithm is particularly troublesome for time – sharing systems, where it is important that each user get a share of the CPU at regular intervals.

Shortest Job First Scheduling

- The CPU is assigned to the process that has the smallest next CPU burst.
- If two processes have the same length next CPU burst, FCFS scheduling is used to break the tie.

Example :

Process	Burst Time
P1	6
P2	8
P3	7
P4	3

Gantt Chart

P4	**P1**	**P3**	**P2**

0 3 9 16 24

Average waiting time is (3 + 16 + 9 + 0)/4 = 7 milliseconds.
Average turnaround time = (3+9+16+24) / 4 = 13 ms

- Preemptive & non preemptive scheduling is used for SJF

Example :

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

Preemptive Scheduling

P1	P2	P4	P1	P3

0 1 5 10 17 26

Average waiting time :

P1 : 10 – 1 = 9

P2 : 1 – 1 = 0

P3 : 17 – 2 = 15

P4 : 5 – 3 = 2

AWT = (9+0+15+2) / 4 = 6.5 ms

• Preemptive SJF is known as shortest remaining time first

Non-preemtive Scheduling

P1	P2	P4	P3

0 8 12
17 26

AWT = 0 + (8 – 1) + (12 – 3) + (17 – 2) / 4 = 7.75 ms

Priority Scheduling

• The SJF algorithm is a special case of the general priority-scheduling algorithm.

• A priority is associated with each process, and the CPU is allocated to the process with the highest priority.(smallest integer □ highest priority).

Example :

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

P2	P5	P1	P3	P4

0 1 6 16 18
19

SJF is a priority scheduling where priority is the predicted next CPU burst time.

• Priority Scheduling can be preemptive or non-preemptive.

• **Drawback** □ □ Starvation– low priority processes may never execute.

• **Solution** □ □ Aging– It is a technique of gradually increasing the priority of processes that wait in the system for a long time.

Round-Robin Scheduling

• The round-robin (RR) scheduling algorithm is designed especially for timesharing systems.

• It is similar to FCFS scheduling, but preemption is added to switch between processes.

• A small unit of time, called a time quantum (or time slice), is defined.

• The ready queue is treated as a circular queue.

Example :

Process Burst Time
P1 24
P2 3
P3 3
Time Quantum = 4 ms.

P1	**P2**	**P3**	**P1**	**P1**	**P1**	**P1**	**P1**

0 4 7 10 14 18
22 26 30

Waiting time
P1 = 26 – 20 = 6
P2 = 4
P3 = 7 (6+4+7 / 3 = 5.66 ms)

• The average waiting time is 17/3 = 5.66 milliseconds.

• The performance of the RR algorithm depends heavily on the size of the time–quantum.

• If time-quantum is very large(infinite) then RR policy is same as FCFS policy.

• If time quantum is very small, RR approach is called processor sharing and appears to the users as though each of n process has its own processor running at 1/n the speed of real processor.

Multilevel Queue Scheduling

• It partitions the ready queue into several separate queues .

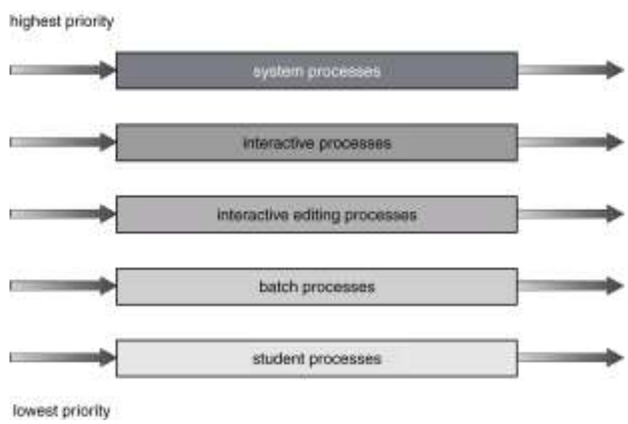
• The processes are permanently assigned to one queue, generally based on some property of the process, such as memory size, process priority, or process type.

• There must be scheduling between the queues, which is commonly implemented as a fixed-priority preemptive scheduling.

• For example the foreground queue may have absolute priority over the background queue.

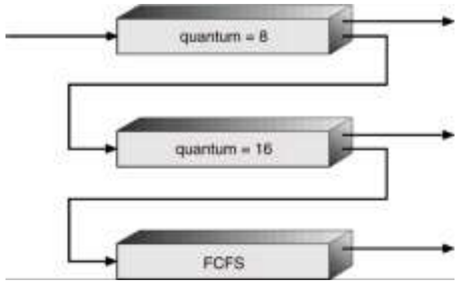
Example : of a multilevel queue scheduling algorithm with five queues
1. System processes
2. Interactive processes
3. Interactive editing processes
4. Batch processes
5. Student processes

Each queue has absolute priority over lower-priority queue.



Multilevel Feedback Queue Scheduling

• It allows a process to move between queues.
• The idea is to separate processes with different CPU-burst characteristics.
• If a process uses too much CPU time, it will be moved to a lower-priority queue.
• This scheme leaves I/O-bound and interactive processes in the higher-priority queues.
• Similarly, a process that waits too long in a lower priority queue may be moved to a higher-priority queue.
• This form of aging prevents starvation.
Example:
• Consider a multilevel feedback queue scheduler with three queues, numbered from 0 to 2 .
• The scheduler first executes all processes in queue 0.
• Only when queue 0 is empty will it execute processes in queue 1.
• Similarly, processes in queue 2 will be executed only if queues 0 and 1 are empty.
• A process that arrives for queue 1 will preempt a process in queue 2.
• A process that arrives for queue 0 will, in turn, preempt a process in queue 1.



• A multilevel feedback queue scheduler is defined by the following parameters:
1. The number of queues
2. The scheduling algorithm for each queue
3. The method used to determine when to upgrade a process to a higher priority queue
4. The method used to determine when to demote a process to a lower-priority queue
5. The method used to determine which queue a process will enter when that process needs service

2.7 Multiple Processor Scheduling

• If multiple CPUs are available, the scheduling problem is correspondingly more complex.
• If several identical processors are available, then load-sharing can occur.
• It is possible to provide a separate queue for each processor.
• In this case however, one processor could be idle, with an empty queue, while another processor was very busy.
• To prevent this situation, we use a common ready queue.
• All processes go into one queue and are scheduled onto any available processor.
• In such a scheme, one of two scheduling approaches may be used.
1. **Self Scheduling** - Each processor is self-scheduling. Each processor examines the common ready queue and selects a process to execute. We must ensure that two processors do not choose the same process, and that processes are not lost from the queue.

2. **Master – Slave Structure** - This avoids the problem by appointing one processor as scheduler for the other processors, thus creating a master-slave structure.

2.8 Real-Time Scheduling

• Real-time computing is divided into two types.
1. Hard real-time systems
2. Soft real-time systems
• Hard RTS 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**.

• Soft real-time computing is less restrictive. It requires that critical processes recieve priority over less fortunate ones.

• 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.

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

• The high-priority process would be waiting for a lower-priority one to finish. This situation is known as **priority inversion**.

2.9 The Critical-Section Problem:

• There are n processes that are 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.

Requirements to be satisfied for a Solution to the Critical-Section Problem:

1. **Mutual Exclusion -** If process Pi is executing in its critical section, then no other processes can be executing in their critical sections.
2. **Progress -** 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.
3. **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.

General structure of process Pi

```
do {
              entry section
             critical section
                 exit section

             exit section
remainder section
} while (1);
```

exit section

Two Process solution to the Critical Section Problem

Algorithm 1:

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

```
        remainder section
    } while (1);
```

CONCLUSION: Satisfies mutual exclusion, but not progress and bounded waiting

Algorithm 2:

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

CONCLUSION: Satisfies mutual exclusion, but not progress and bounded waiting

Algorithm 3:

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

CONCLUSION: Meets all three requirements; solves the critical-section problem for two processes.

Multiple –process solution or n- process solution or Bakery Algorithm :

• Before entering its critical section, process receives a number. Holder of the smallest number enters the critical section.

• If processes Pi and Pj receive the same number, if i < j, then Pi is served first; else Pj is served first.

• (a,b) < (c,d) if a < c or if a = c and b < d

• boolean choosing[n];

int number[n];

Data structures are initialized to false and 0 respectively

```
do {
flag[i]=true;
turn = j;
while (flag[j]&& turn==j) ;
flag[i]=false;
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);
1.Mutual Exclusion is satisfied.
number[i] = 0;
2.Progress and Bounded waiting are also satisfied as the processes enter the critical section on a FCFS basis.

2.10 Synchronization Hardware:

- Test and modify the content of a word atomically
.

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

Mutual Exclusion with Swap

- Shared data (initialized to false):
  boolean lock;
  boolean waiting[n];
- Process P_i

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

2.11 Semaphores

- Synchronization tool that does not require busy waiting.
- Semaphore *S* – integer variable
- can only be accessed via two indivisible (atomic) operations

  wait (*S*):
    while *S*≤ 0 do *no-op*;
    ***S*--;**
  signal (*S*):
    ***S*++;**

Critical Section of *n* Processes

- Shared data:

semaphore mutex; //initially *mutex* = 1

- Process *Pi:*

```
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* executed in P_i
- Use semaphore *flag* initialized to 0
- Code:

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

Deadlock & starvation:

Example: Consider a system of two processes , P0 & P1 each

accessing two semaphores ,S & Q, set to the value 1.

P0	P1
Wait (S)	Wait (Q)
Wait (Q)	Wait (S)
Signal(S)	Signal(Q)
Signal(Q)	Signal(S)

- Suppose that P0 executes wait(S), then P1 executes wait(Q). When P0 executes wait(Q), it must wait until P1 executes signal(Q).Similarly when P1 executes wait(S), it must wait until P0 executes signal(S). Since these signal operations cannot be executed, P0 & P1 are deadlocked.
- Another problem related to deadlock is indefinite blocking or starvation, a situation where a process wait indefinitely within the semaphore. Indefinite blocking may occur if we add or remove processes from the list associated with a semaphore in LIFO order.

Types of Semaphores

- *Counting* semaphore - any positive integer value
- *Binary* semaphore - integer value can range only between 0 and 1

Classical Problems of Synchronization

- Bounded-Buffer Problem
- Readers and Writers Problem
- Dining-Philosophers Problem

Bounded Buffer Problem

Shared data

semaphore full, empty, mutex;

// initially full = 0, empty = n, mutex = 1

Structure of Producer Process

```
do {
…
produce an item in nextp
…
wait(empty);
wait(mutex);
…
add nextp to buffer
…
signal(mutex);
signal(full);
} while (1);
```

Structure of 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);
```

Readers-Writers Problem

Shared data

semaphore wrt, mutex;

// initially wrt=1, mutex = 1,readcount=0

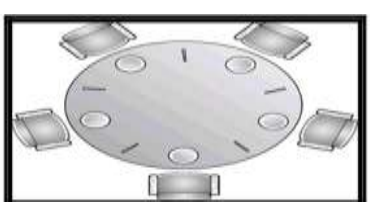
Structure of Writer Process

```
do{
wait(wrt);
…
writing is performed
…
      signal(wrt);
}while(1);
```

Structure of Reader Process

```
do{
wait(mutex);
readcount++;
if (readcount == 1)
wait(rt);
signal(mutex);
…
reading is performed
…
wait(mutex);
readcount--;
if (readcount == 0)
signal(wrt);
      signal(mutex);
}while(1);
```

Dining-Philosophers Problem



Shared data

semaphore chopstick[5];

// Initially all values are 1

Structure of Philosopher i

```
do {
```

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

Critical Region

✓The problems with semaphores are :

•Correct use of semaphore operations:

○ signal (mutex) …. wait (mutex)

▪ Several processes may be executing in their critical sections simultaneously, violating the mutual-exclusion requirement

○ wait (mutex) … wait (mutex)

▪ A deadlock will occur

○ Omitting of wait (mutex) or signal (mutex) (or both)

▪ Either mutual exclusion is violated or a deadlock will occur

✓ Hence we use high level synchronization construct called as critical

region. ✓ A shared variable v of type T is declared as

○ **v: shared T**

✓ Variable v is accessed only inside

the 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

2.12 Monitors

✓ A high-level abstraction that provides a convenient and effective mechanism for process synchronization

✓ Only one process may be active within the monitor at a time

monitor monitor-name

{

// shared variable

declarations

procedure body P1 (…) {

…. }

…

procedure body Pn (…)

{……}

{

initialization code

}

}

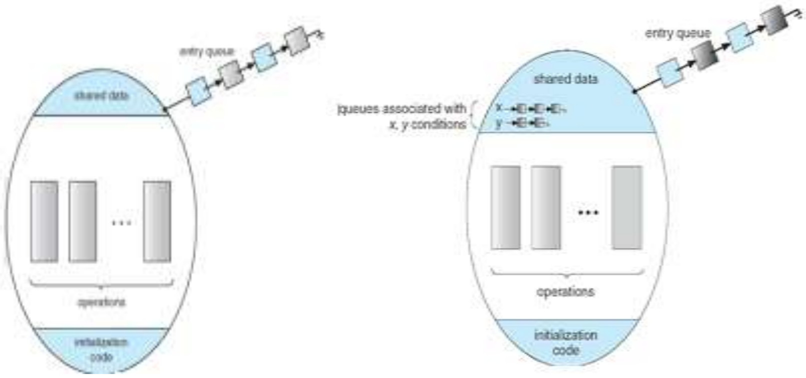
✓ To allow a process to wait within the monitor, a condition variable must be declared as condition x, y;

✓ Two operations on a condition variable:

✓ x.wait () -a process that invokes the operation isuspended.

✓ x.signal () -resumes one of the suspended processes(if any)

Schematic view of a monitor



Solution to Dining Philosophers Problem

monitor DP

{

```
enum { THINKING; HUNGRY,
EATING) state [5] ; condition self
[5];
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 () ;
}
}
initialization_code() {
for (int i = 0; i < 5; i++)
state[i] = THINKING;
}
}
```

UNIT- III DEADLOCKS

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

3.1 System Model

Definition:

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

A process must request a resource before using it, and must release resource after usir

1. **Request:** If the request cannot be granted immediately then the requesting process wait until it can acquire the resource.
2. **Use:** The process can operate on the resource
3. **Release:** The process releases the resource.

3.2 Deadlock Characterization

Four Necessary conditions for a deadlock

1. **Mutual exclusion:** At least one resource must be held in a non sharable mode.Tha only one process at a time can use the resource. If another process requests that resour the requesting process must be delayed until the resource has been released.
2. **Hold and wait:** A process must be holding at least one resource and waiting to acc additional resources that are currently being held by other processes.
3. **No preemption:** Resources cannot be preempted.
4. **Circular wait:** P_0 is waiting for a resource that is held by P_1, P_1 is waiting for a res that is held by P_2...P_{n-1}.

Resource-Allocation Graph

•It is a Directed Graph with a set of vertices V and set of edges E.

• V is partitioned into two types:

1. nodes P = {p1, p2,..pn}
2. Resource type R ={R1,R2,...Rm}

• Pi -->Rj - request => request edge

• Rj-->Pi - allocated => assignment edge.

• Pi is denoted as a circle and Rj as a square.

• Rj may have more than one instance represented as a dot with in the square.

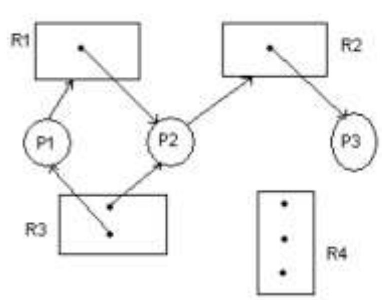
Sets P,R and E.

P = { P1,P2,P3}

R = {R1,R2,R3,R4}

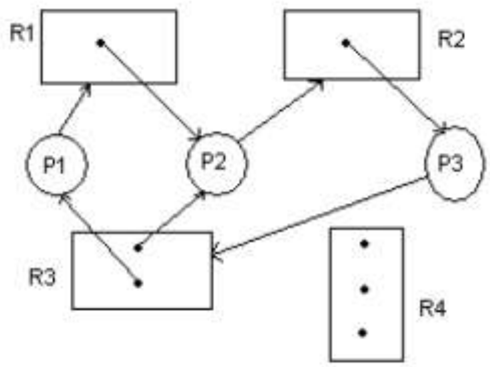
E= {P1->R1, P2->R3, R1->P2, R2->P1, R3->P3 } • Resource instances

One instance of resource type R1,Two instance of resource type R2,One instance of resource type R3,Three instances of resource type R4.



Process states

Process P1 is holding an instance of resource type R2, and is waiting for an insta
of resource type R1.**Resource Allocation Graph with a deadlock**



Process P2 is holding an instance of R1 and R2 and is waiting for an instance of resou
type R3.Process P3 is holding an instance of R3.
P1->R1->P2->R3->P3->R2->P1
P2->R3->P3->R2->P2

3.3 Methods for handling Deadlocks
1. Deadlock Prevention
2. Deadlock Avoidance
3. Deadlock Detection and Recovery

3.4 Deadlock Prevention:
- This ensures that the system never enters the deadlock state.
- Deadlock prevention is a set of methods for ensuring that at least one of the necessary conditions cannot hold.
- By ensuring that at least one of these conditions cannot hold, we can prevent the occurre a deadlock.

1. Denying Mutual exclusion
- Mutual exclusion condition must hold for non-sharable resources.
- Printer cannot be shared simultaneously shared by prevent processes. • sharable resou example Read-only files.
- If several processes attempt to open a read-only file at the same time, they can be gr simultaneous access to the file.
- A process never needs to wait for a sharable resource.

2. Denying Hold and wait
- Whenever a process requests a resource, it does not hold any other resource.
- One technique that can be used requires each process to request and be allocated resources before it begins execution.
- Another technique is before it can request any additional resources, it must release al resources that it is currently allocated.
- These techniques have two main **disadvantages** :

o First, resource utilization may be low, since many of the resources may be alloca but unused for a long time.

o We must request all resources at the beginning for both protocols. starvation is possible.

3. Denying No preemption

• If a Process is holding some resources and requests another resource that cannot be immediately allocated to it. (that is the process must wait), then all resources currently being held are preempted.(**ALLOW PREEMPTION**)

• These resources are implicitly released.

• The process will be restarted only when it can regain its old resources.

4. Denying Circular wait

• Impose a total ordering of all resource types and allow each process to request f resources in an increasing order of enumeration.

• Let R = {R1,R2,...Rm} be the set of resource types.

• Assign to each resource type a unique integer number.

• If the set of resource types R includes tapedrives, disk drives and printers.

F(tapedrive)=1,

F(diskdrive)=5,

F(Printer)=12.

• Each process can request resources only in an increasing order of enumerat

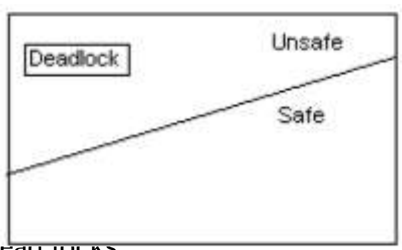
3.5 Deadlock Avoidance:

• Deadlock avoidance request that the OS be given in advance additional informat concerning which resources a process will request and use during its life time. With information it can be decided for each request whether or not the process should wait.

• To decide whether the current request can be satisfied or must be delayed, a system r consider the resources currently available, the resources currently allocated to eacl process and future requests and releases of each process.

• **Safe State**

A state is safe if the system can allocate resources to each process in some order and st avoid a dead lock.



- A deadlock is an unsafe s
- Not all unsafe states are dead locks
- An unsafe state may lead to a dead lock

• Two algorithms are used for deadlock avoidance namely;

1. Resource Allocation Graph Algorithm - single instance of a resource type.
2. Banker's Algorithm - several instances of a resource type.

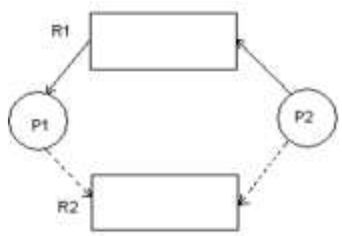
Resource allocation graph algorithm

• **Claim edge** - Claim edge P_i---> R_j indicates that process Pi may request resource some time, represented by a dashed directed edge.

• When process Pi request resource R_j, the claim edge P_i -> R_j is converted to a request

• Similarly, when a resource R_j is released by P_i the assignment edge R_j -> P_i is reconv to a claim edge P_i -> R_j

• The request can be granted only if converting the request edge P_i -> R_j to an assignme edge R_j -> P_i does not form a cycle.



• If no cycle exists, then the allocation of the resource will leave the system in a safe stat

• If a cycle is found, then the allocation will put the system in an unsafe state.

Banker's algorithm

•**Available:** indicates the number of available resources of each type.

•**Max:** Max[i, j]=k then process P_i may request at most k instances ofresource ty

•**Allocation :** Allocation[i. j]=k, then process P_i is currently allocated Kinstances of resource type R_j

•**Need :** if Need[i, j]=k then process P_i may need K more instances ofresource typ
Need [i, j]=Max[i, j]-Allocation[i, j]

Safety algorithm

1. Initialize work := available and 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, goto step 4
3. work :=work+ $allocation_i$;
Finish[i]:=true
goto step 2
4. If finish[i]=true for all i, then the system is in a safe state

Resource Request Algorithm

Let $Request_i$ be the request from process P_i for resources.
1. If $Request_i$<= $Need_i$ goto step2, otherwise raise an error condition, since the pro has exceeded its maximum claim.
2. If $Request_i$ <= Available, goto step3, otherwise P_i must wait, since the resources are available.
3. Available := Availabe-$Request_i$;
$Allocation_i$:= $Allocation_i$ + $Request_i$
$Need_i$:= $Need_i$ - $Request_i$;
• Now apply the safety algorithm to check whether this new state is safe or not. • If it is then the request from process P_i can be granted.

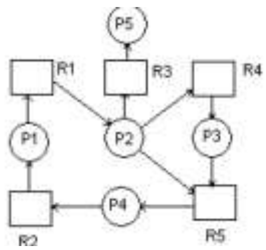
3.6 Deadlock detection

(i) Single instance of each resource type

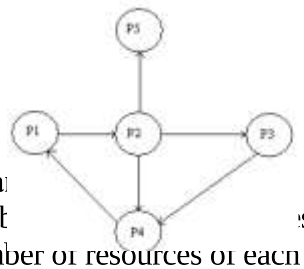
• If all resources have only a single instance, then we can define a deadlock detection algorithm that use a variant of resource-allocation graph called a wait for graph.

Resource Allocation Graph

`



Wait for Graph



(ii) Several Insta

Available : Numb ... s of each type

Allocation : number of resources of each type currently allocated to each process **Requ**

Current request of each process

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

1. Initialize work := available

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 step4.

3. Work:=work+$allocation_i$

Finish[i]:=true

goto step2

4. If finish[i]=false

then process P_i is deadlocked

<u>3.7 Deadlock Recovery</u>

1. Process Termination

1. Abort all deadlocked processes.
2. Abort one deadlocked process at a time until the deadlock cycle is eliminated.

After each process is aborted , a deadlock detection algorithm must be in to determine where any process is still dead locked.

2. Resource Preemption

Preemptive some resources from process and give these resources to other pro until the deadlock cycle is broken.

i. **Selecting a victim:** which resources and which process are to be preempted.

ii. **Rollback:** if we preempt a resource from a process it cannot continue w normal execution. It is missing some needed resource. we must rollback the process some safe state, and restart it from that state.

iii. **Starvation :** How can we guarantee that resources will not always be preempted from the same process. ‘

3.8 Storage Management: Background

• In general, to rum a program, it must be brought into memory.

• **Input queue** - collection of processes on the disk that are waiting to be brought i memory to run the program.

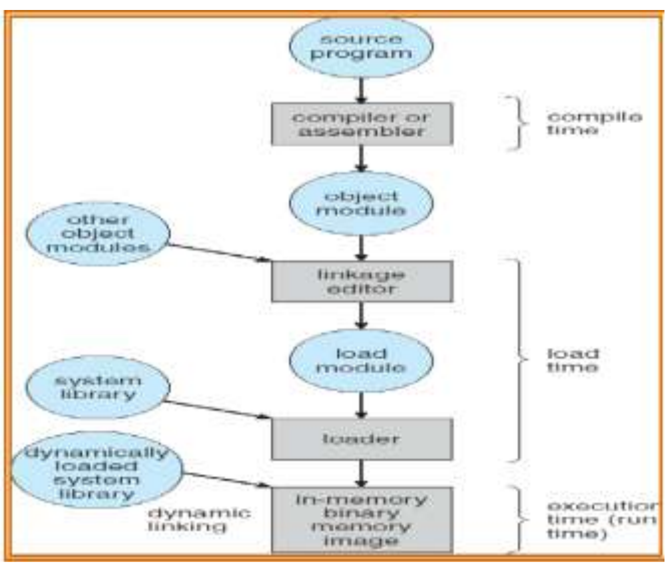
• User programs go through several steps before being run

• **Address binding**: Mapping of instructions and data from one address to another address in memory.

Three different stages of binding:

1. **Compile time**: Must generate absolute code if memory location is known in prior.
2. **Load time**:   Must generate relocatable code if memory location is not known compile time
3. **Execution time**: Need hardware support for address maps (e.g., base and limit registers).

Multistep Processing of a User Program



Logical vs. Physical Address Space

• **Logical address** - generated by the CPU; also referred to as **“virtual address**

• **Physical address** - address seen by the memory unit.

• Logical and physical addresses are the **same** in —compile-time and load-time address-binding schemes

• Logical (virtual) and physical addresses **differ** in —execution-time address-binding sch

Memory-Management Unit (MMU)

• It is a hardware device that maps virtual / Logical address to physical address

• In this scheme, the relocation register's value is added to Logical address generated user process.

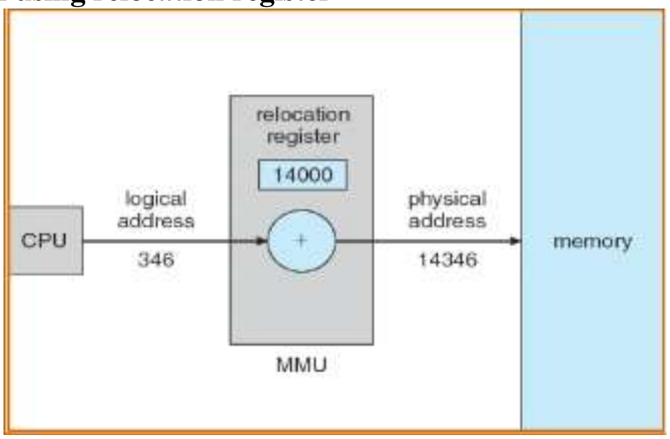
• The user program deals with *logical* addresses; it never sees the *real* physical addresses

• Logical address range: 0 to max

• Physical address range: R+0 to R+max, where R—value in relocation register

Note: relocation register is a base register.

Dynamic relocation using relocation register



Dynamic Loading

• Through this, the routine is not loaded until it is called.

o Better memory-space utilization; unused routine is never loaded

o Useful when large amounts of code are needed to handle infrequently occurring cases

o No special support from the operating system is required implemented through progra design

Dynamic Linking

• Linking postponed until execution time & is particularly useful for libraries • Small p of code

called stub, used to locate the appropriate memory-resident library routine or functi

• Stub replaces itself with the address of the routine, and executes the routine

• Operating system needed to check if routine is in processes' memory address

• **Shared libraries**: Programs linked before the new library was installed will continue the older library

Overlays:

• Enable a process larger than the amount of memory allocated to it.

• At a given time, the needed instructions & data are to be kept within a memory.

3.9 Swapping

• A process can be swapped temporarily out of memory to a backing store (SWAP OUT)and then brought back into memory for continued execution (SWAP IN).

• **Backing store** - fast disk large enough to accommodate copies of all memory images for all users & it must provide direct access to these memory images

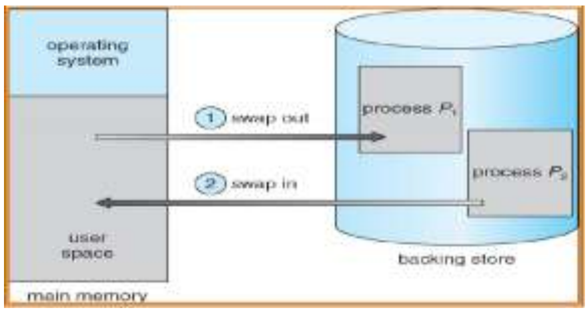
• **Roll out, roll in** - swapping variant used for priority-based scheduling algorithn lower-priority process is swapped out so higher-priority process can be loaded and execu

• **Transfer time** :

✓ Major part of swap time is transfer time

✓ Total transfer time is directly proportional to the amount of memory swapped.

✓ **Example**: Let us assume the user process is of size 1MB & the backing store is a stan hard disk with a transfer rate of 5MBPS. Transfer time= 1000KB/5000KB per second= sec = 200ms



(i) Memory Protection:

○ It should consider;

a) Protecting the OS from user process.

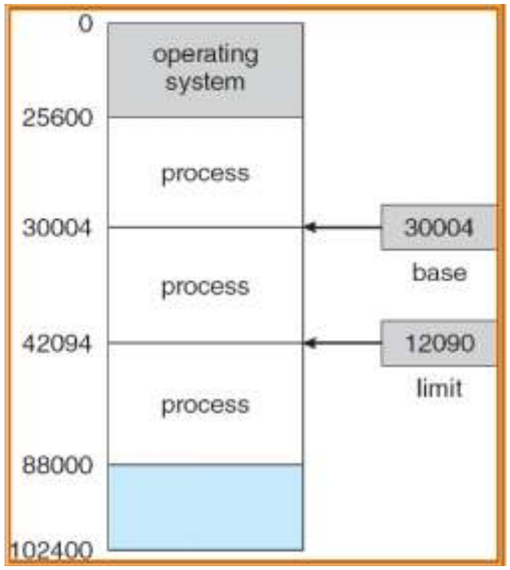
b) Protecting user processes from one another.

○ The above protection is done by **"Relocation-register & Limit-register scheme—**

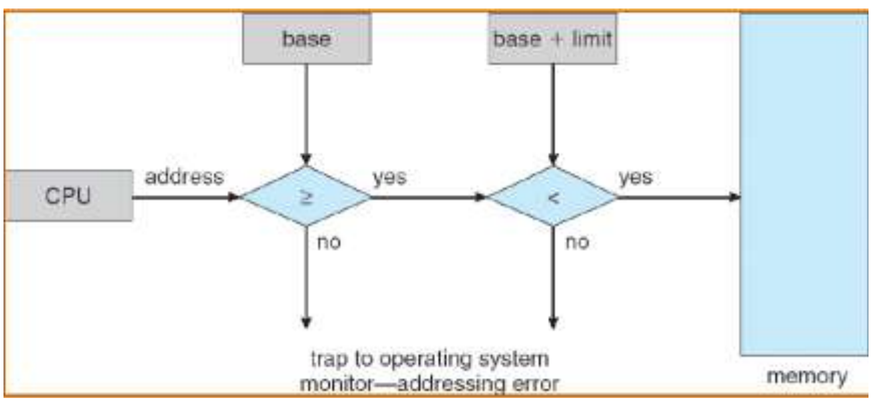
○ Relocation register contains value of smallest physical address i.e base value.

○ Limit register contains range of logical addresses - each logical address must be than the limit register

A base and a limit register define a logical address space



HW address protection with base and limit registers



3.10 Contiguous Memory Allocation

•Each process is contained in a single contiguous section of memory.

•There are two methods namely :

- Fixed - Partition Method
- Variable - Partition Method

• **Fixed - Partition Method** :

o Divide memory into fixed size partitions, where each partition has exactly one prc

o The drawback is memory space unused within a partition is wasted.(eg.whe process size < partition size)

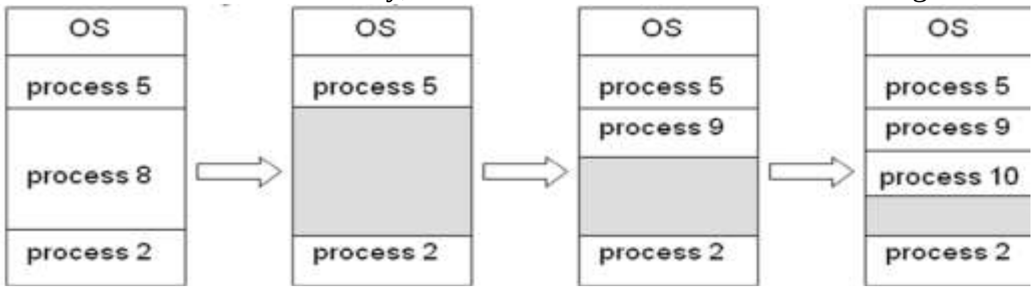
• **Variable-partition method:**

o Divide memory into variable size partitions, depending upon the size of the incomin process.

o When a process terminates, the partition becomes available for another proces

o As processes complete and leave they create holes in the main memory.

o ***Hole*** - block of available memory; holes of various size are scattered throughout me



Solution:

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

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

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

NOTE: First-fit and best-fit are better than worst-fit in terms of speed and storage util

•Fragmentation:

o **External Fragmentation** - This takes place when enough total memory space e to satisfy a request, but it is not contiguous i.e, storage is fragmented into a large n of small holes scattered throughout the main memory.

o**Internal Fragmentation** - Allocated memory may be slightly larger thanrequested memory.

Example: hole = 184 bytes Process size = 182 bytes.
We are left with a hole of 2 bytes.

o**Solutions:**

1. **Coalescing :** Merge the adjacent holes together.
2. **Compaction:** Move all processes towards one end of memory, hole towards oth of memory, producing one large hole of available memory. This scheme is expensi it can be done if relocation is dynamic and done at execution time.
3. Permit the logical address space of a process to be **non-contiguous**. This is achi through two memory management schemes namely **paging** and **segmentation**.

3.11 Paging

• It is a memory management scheme that permits the physical address space of a proce be noncontiguous.

• It avoids the considerable problem of fitting the varying size memory chunks on to the backing store.

(i) Basic Method:

○ Divide logical memory into blocks of same size called **"pages"**.

○ Divide physical memory into fixed-sized blocks called **"frames"**

○ Page size is a power of 2, between 512 bytes and 16MB.

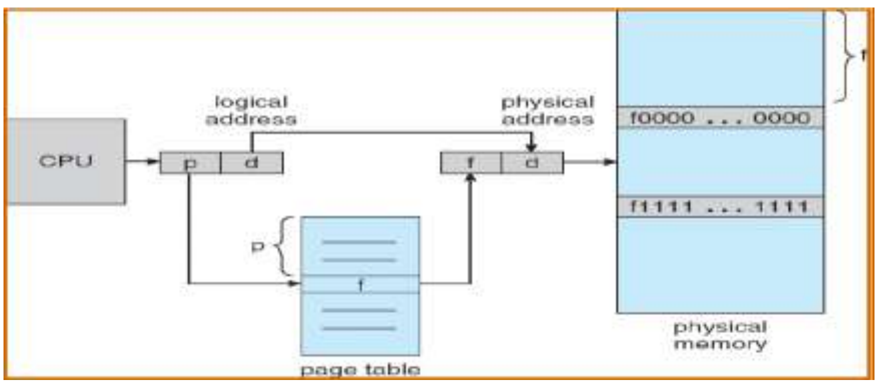
Address Translation Scheme

○ Address generated by CPU(logical address) is divided into:

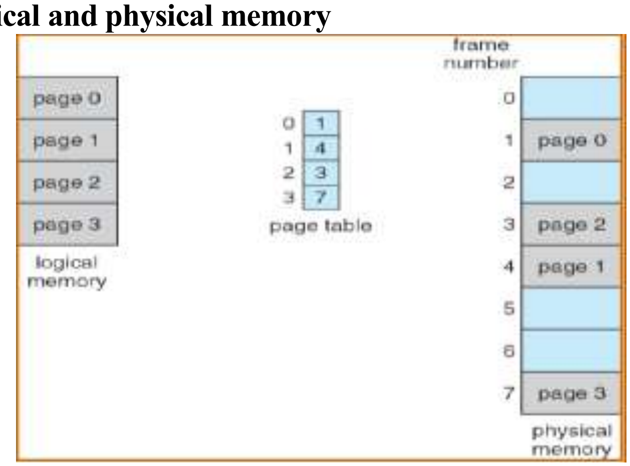
✓ **Page number *(p)*** - used as an index into a page table which contains base address of each page in physical memory

✓ **Page offset *(d)*** - combined with base address to define the physical address i.e., Physical address = base address + offset

Paging Hardware



Paging model of logical and physical memory



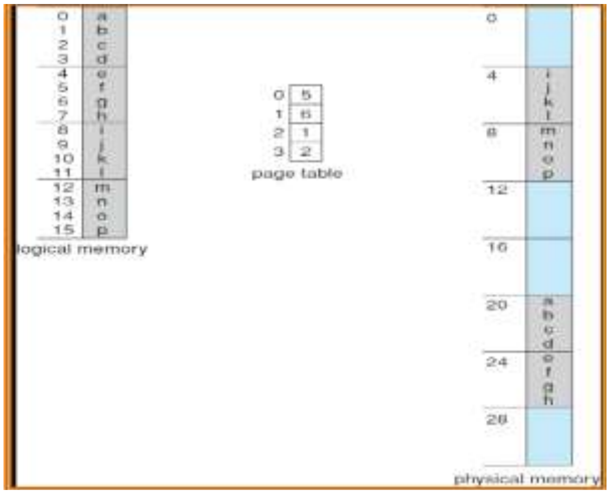
Paging example for a 32-byte memory with 4-byte pages

Page size = 4 bytes

Physical memory size = 32 bytes i.e (4 X 8 = 32 so, 8 pages)

Logical address _0' maps to physical address 20 i.e ((5 X 4) +0)

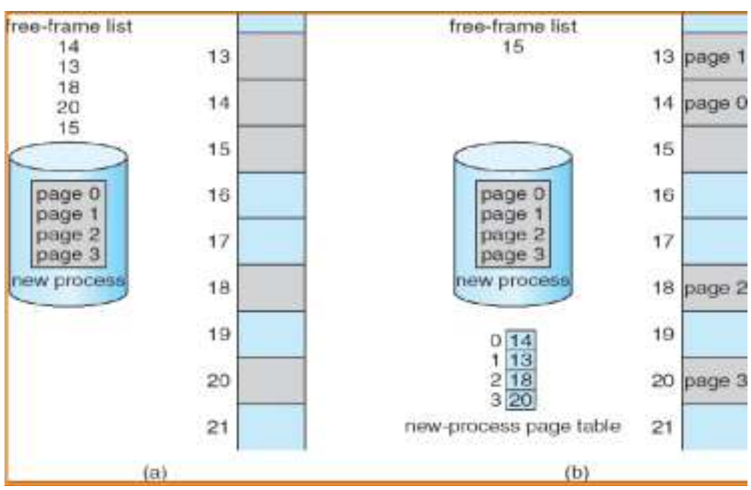
Where Frame no = 5, Page size = 4,Offset= 0



Allocation

- When a process arrives into the system, its size (expressed in pages) is examined.
- Each page of process needs one frame. Thus if the process requires _n' pages, at least frames must be available in memory.
- If _n' frames are available, they are allocated to this arriving process.
- The 1st page of the process is loaded into one of the allocated frames & the frame numb put into the page table.

Repeat the above step for the next pages & so on.



(a) Before Allocation (b) After Allocation

Frame table: It is used to determine which frames are allocated, which frames are avai how many total frames are there, and so on.(ie) It contains all the information about the in the physical memory.

(ii) Hardware implementation of Page Table

○ This can be done in several ways :

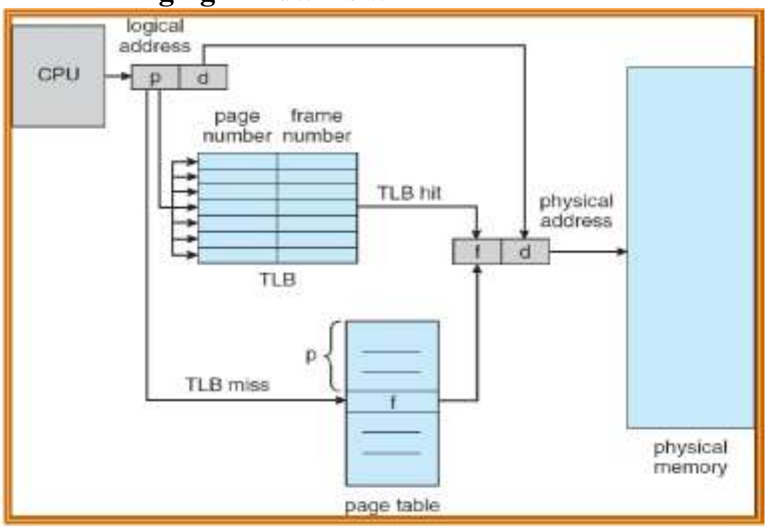
1. Using PTBR
2. TLB

○ The simplest case is **Page-table base register (PTBR)**, is an index to point the pag table.

○ **TLB (Translation Look-aside Buffer)**

- It is a fast lookup hardware cache.
- It contains the recently or frequently used page table entries
- It has two parts: Key (tag) & Value.
- More expensive.

Paging Hardware with TLB



○When a logical address is generated by CPU, its page number is presented toTLB.

○**TLB hit**: If the page number is found, its frame number is immediatelyavailable & is u access memory

○**TLB miss**: If the page number is not in the TLB, a memory reference to thepage table be made.

○**Hit ratio:** Percentage of times that a particular page is found in the TLB.

- For example hit ratio is 80% means that the desired page number in the TLB is 80% of time.

○ **Effective Access Time:**

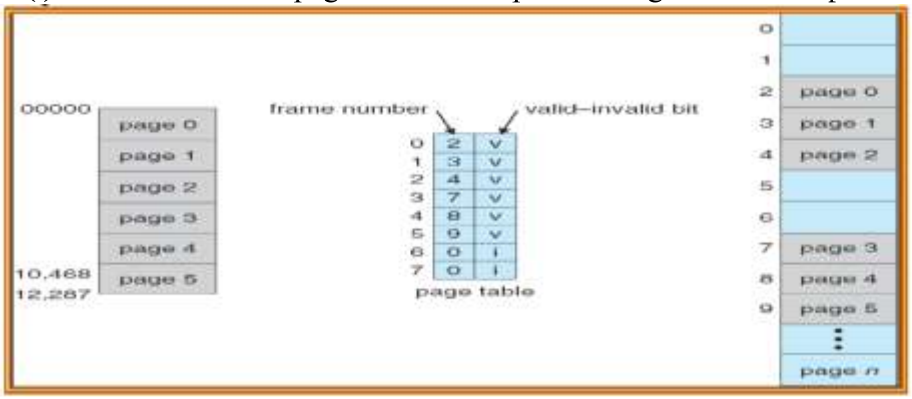
- Assume hit ratio is 80%.
- If it takes 20ns to search TLB & 100ns to access memory, then the memory access ta 120ns(TLB hit)
- If we fail to find page no. in TLB (20ns), then we must 1^{st} access memory for page ta (100ns) & then access the desired byte in memory (100ns).

Therefore Total = 20 + 100 + 100 = 220 ns(TLB miss).

Then Effective Access Time (EAT) = 0.80 X (120 + 0.20) X 220. = 140 ns.

(iii) Memory Protection

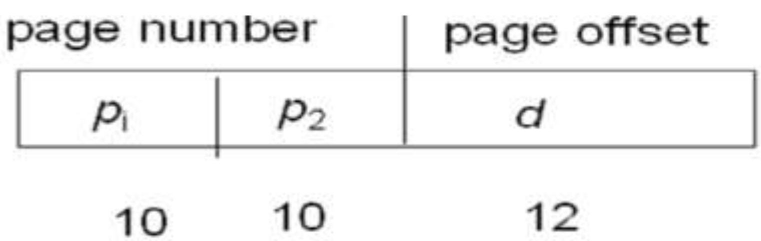
○ Memory protection implemented by associating protection bit with each frame
○ **Valid-invalid** bit attached to each entry in the page table:
✓ **"valid (v)"** indicates that the associated page is in the process' logical address space, thus a legal page
✓ **"invalid (i)"** indicates that the page is not in the process' logical address space

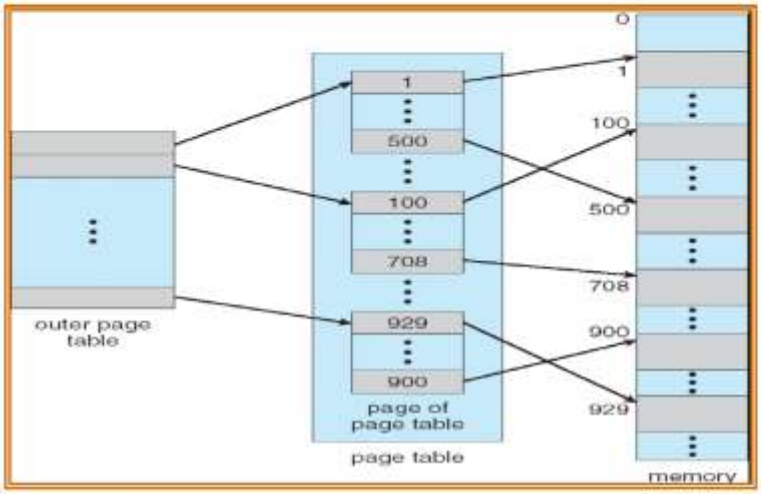


(iv) Structures of the Page Table

a) Hierarchical Paging
b) Hashed Page Tables
c) Inverted Page Tables

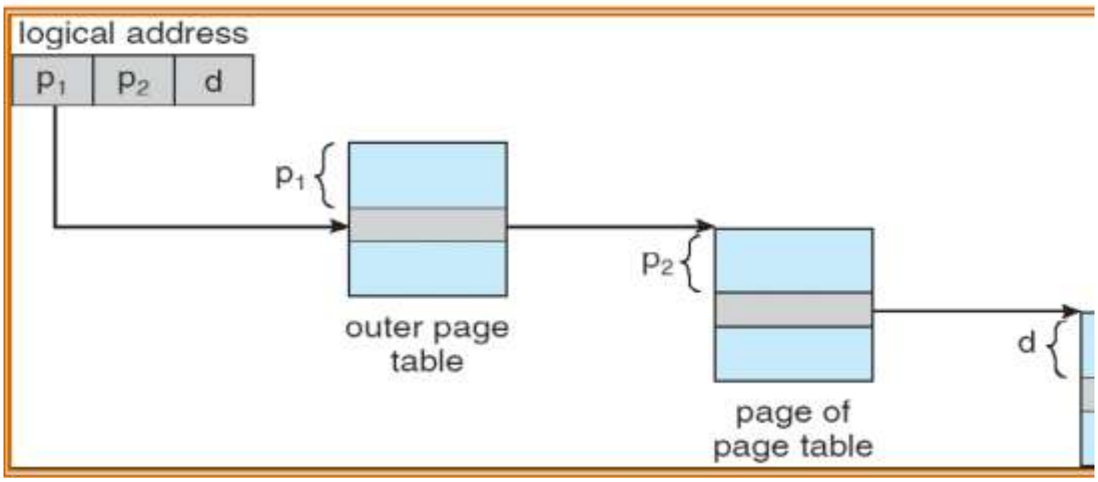
a) Hierarchical Paging
○ Break up the Page table into smaller pieces. Because if the page table is too large the quit difficult to search the page number.
Example: "Two-Level Paging "





Address-Translation Scheme

Address-translation scheme for a two-level 32-bit paging architecture



It requires more number of memory accesses, when the number of levels is increased.

(b) Hashed Page Tables

- Each entry in hash table contains a linked list of elements that hash to the same locat
- Each entry consists of;

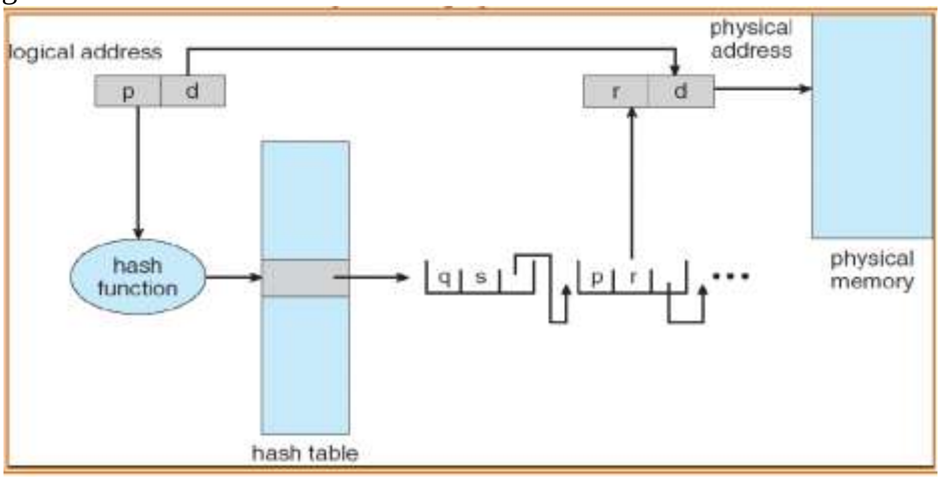
(a) Virtual page numbers

(b) Value of mapped page frame.

(c) Pointer to the next element in the linked list. o Working Procedure:

- The virtual page number in the virtual address is hashed into the hash table.
- Virtual page number is compared to field (a) in the 1st element in the linked list.
- If there is a match, the corresponding page frame (field (b)) is used to form the desir physical address.

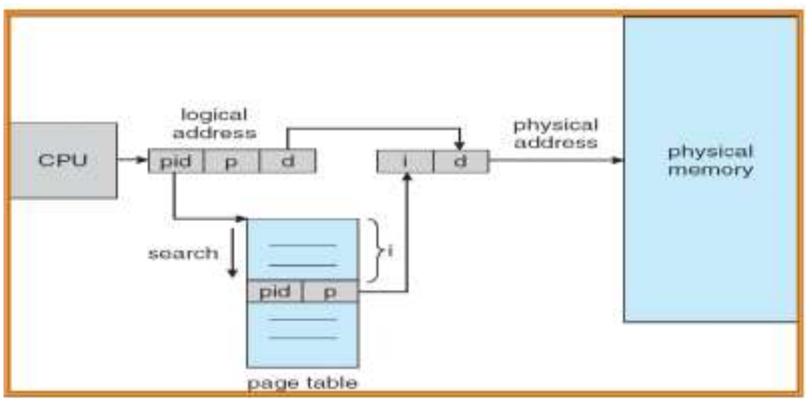
➢ If there is no match, subsequent entries in the linked list are searched for a matching virtual page number.



Clustered page table: It is a variation of hashed page table & is similar to hashed page except that each entry in the hash table refers to several pages rather than a single page.

(c)Inverted Page Table

○ It has one entry for each real page (frame) of memory & each entry consists of the virtual address of the page stored in that real memory location, with information about the process that owns that page. So, only one page tabl the system.



○ When a memory reference occurs, part of the virtual address ,consisting of <Process Page-no> is presented to the memory sub-system.

○ Then the inverted page table is searched for match:
(i)If a match is found, then the physical address is generated.
(ii)If no match is found, then an illegal address access has beenattempted.
○ **Merit:** Reduce the amount of memory needed.

○ **Demerit:** Improve the amount of time needed to search the table when a page referenc oocurs.

(v) Shared Pages

○ One advantage of paging is the possibility of sharing common code. ○ **Shared code**
✓ One copy of read-only (reentrant) code shared among processes (i.e., text editors, compilers, window systems).
✓ Shared code must appear in same location in the logical address space of all process
○ **Reentrant code (Pure code):** Non-self modifying code. If the code is reentrant, then never changes during execution. Thus two or more processes can execute the same code same time.
○ **Private code and data**
✓ Each process keeps a separate copy of the code and data

✓ The pages for the private code and data can appear anywhere in the logical address sp

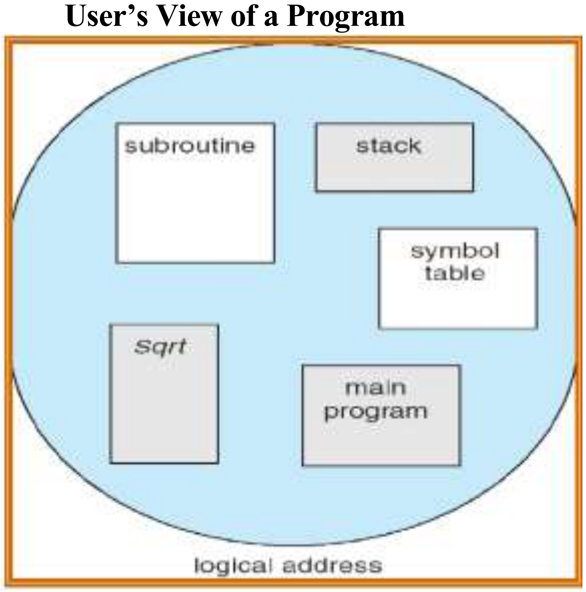
Drawback of Paging - Internal fragmentation

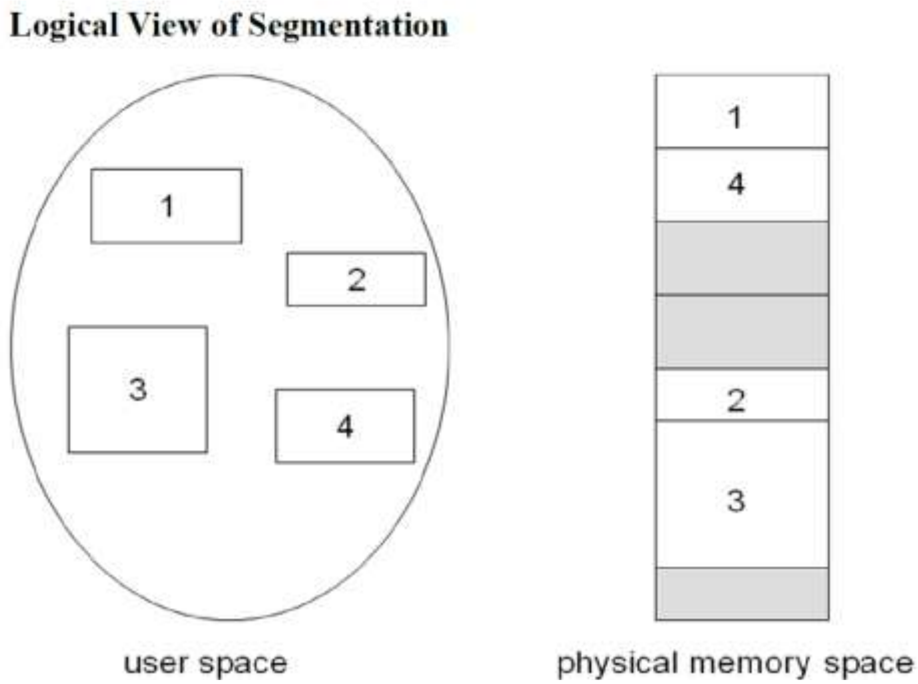
○ In the worst case a process would need n pages plus one byte.It would be allocate frames resulting in an **internal fragmentation** of almost an entire frame.

<u>3.12 Segmentation</u>

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

User's View of a Program





Segmentation Hardware

o Logical address consists of a two tuple :
<Segment-number, offset>

o**Segment table** - maps two-dimensional physical addresses; each table entry has:

✓ **Base** - contains the starting physical address where the segmentsreside in memory
✓ **Limit** - specifies the length of the segment

o***Segment-table base register (STBR)*** points to the segment table's locationin memo

o***Segment-table length register (STLR)*** indicates number of segments usedby a prog

Segment number_s' is legal, if *s* < STLR

o**Relocation**.

✓ dynamic
✓ by segment table

o**Sharing**.

✓ shared segments
✓ same segment number

o**Allocation**.

✓ first fit/best fit
✓ external fragmentation

o **Protection:** With each entry in segment table associate:

✓ validation bit = 0 □ illegal segment

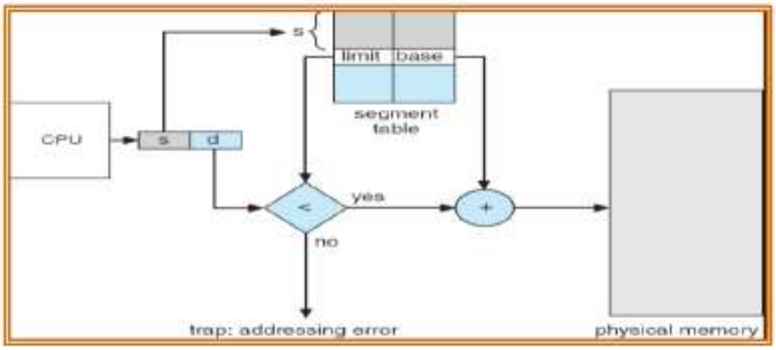
✓ read/write/execute privileges

○Protection bits associated with segments; code sharing occurs at segmentlevel

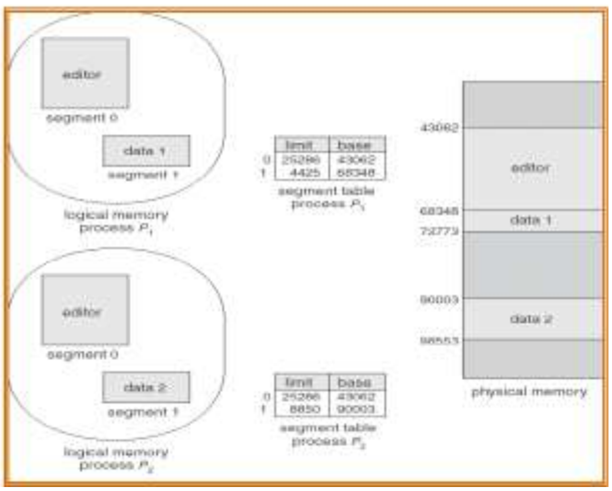
○Since segments vary in length, memory allocation is a dynamic storage allocation problem

○A segmentation example is shown in the following diagram

Address Translation scheme



Sharing of Segments



○ Another advantage of segmentation involves the sharing of code or data.

○ Each process has a segment table associated with it, which the dispatcher uses to c the hardware segment table when this process is given the CPU.

○ Segments are shared when entries in the segment tables of two different pro point to the same physical location.

3.13 Segmentation with paging

○ The IBM OS/ 2.32 bit version is an operating system running on top of the Int 386 architecture. The 386 uses segmentation with paging for memory managemen maximum number of segments per process is 16 KB, and each segment can be as larg gigabytes.

○ The local-address space of a process is divided into two partitions.

▪ The first partition consists of up to 8 KB segments that are private to that process.

s	g	p
13	1	2

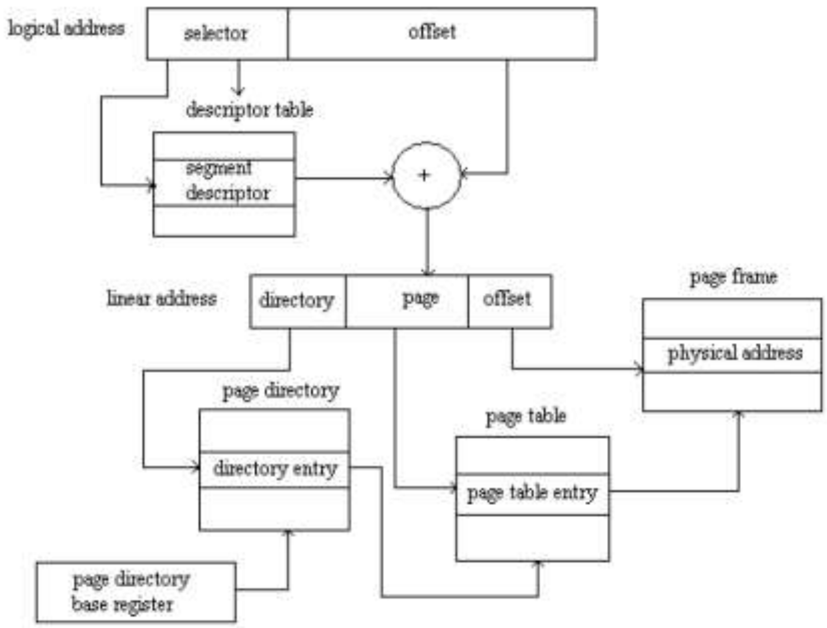
Where s designates the segment number, g indicates whether the seg is in the GDT or LDT, and p deals with protection. The offset is a 32-bit number specif the location of the byte within the segment in question.

- The base and limit information about the segment in question are used to gene linear-address.
- First, the limit is used to check for address validity. If the address is not valid, a memo fault is generated, resulting in a trap to the operating system. If it is valid, then the val the offset is added to the value of the base, resulting in a 32-bit linear address. This addr then translated into a physical address.
- The linear address is divided into a page number consisting of 20 bits, and a page offset consisting of 12 bits. Since we page the page table, the page number is fur divided into a 10-bit page directory pointer and a 10-bit page table pointer. The logi address is as follows.

▪ The second partition consists of up to 8KB segments that are shared among all the processes.

- Information about the first partition is kept in the **local descriptor table (LD** information about the second partition is kept in the **global descriptor table (GDT)**.
- Each entry in the LDT and GDT consist of 8 bytes, with detailed information abou particular segment including the base location and length of the segment. The logi address is a pair (selector, offset) where the selector is a16-bit number:

p_1	p_2	d
10	10	12



○ To improve the efficiency of physical memory use. Intel 386 page tables can be swapped to disk. In this case, an invalid bit is used in the page directory entry to in whether the table to which the entry is pointing is in memory or on disk.

○ If the table is on disk, the operating system can use the other 31 bits to specify the d location of the table; the table then can be brought into memory on demand.

UNIT- IV VIRTUAL MEMORY

Virtual Memory – Demand Paging – Process creation – Page Replacement – Allocat frames – Thrashing - File Concept – Access Methods – Directory Structure – File Sys Mounting – File Sharing – Protection

4.1 Virtual Memory

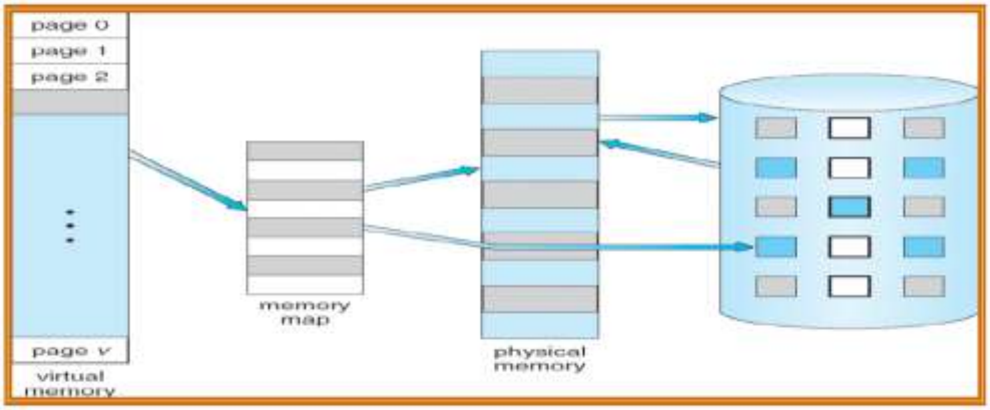
○ It is a technique that allows the execution of processes that may not be compl in main memory.

○ **Advantages:**

✓ Allows the program that can be larger than the physical memory. ✓ Separation of us logical memory from physical memory

✓ Allows processes to easily share files & address space.
✓ Allows for more efficient process creation.
o Virtual memory can be implemented using
✓ Demand paging
✓ Demand segmentation

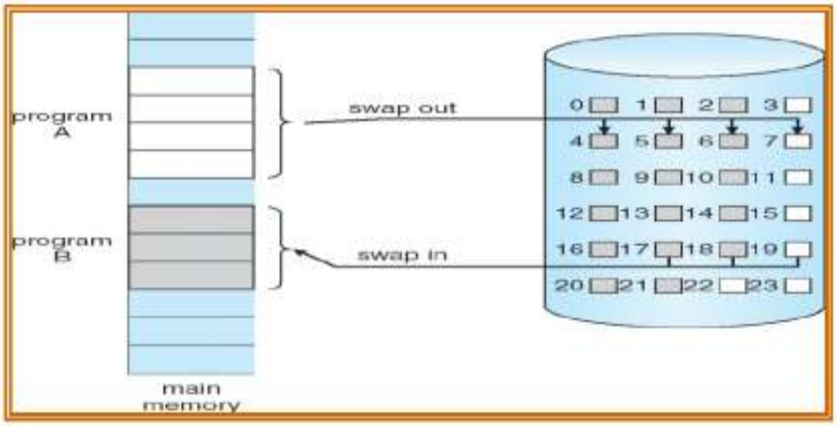
Virtual Memory That is Larger than Physical Memory



4.2 Demand Paging

o It is similar to a paging system with swapping.
o Demand Paging - Bring a page into memory only when it is needed
o To execute a process, swap that entire process into memory. Rather than swappin entire process into memory however, we use ―Lazy Swapper‖
o **Lazy Swapper** - Never swaps a page into memory unless that page will be needed.
o **Advantages**
✓ Less I/O needed
✓ Less memory needed
✓ Faster response
✓ More users

Transfer of a paged memory to contiguous disk space



Basic Concepts:

○ Instead of swapping in the whole processes, the pager brings only those necess pages into memory. Thus,

1. It avoids reading into memory pages that will not be used anyway.
2. Reduce the swap time.
3. Reduce the amount of physical memory needed.

○ To differentiate between those pages that are in memory & those that are on the disl use the **Valid-Invalid bit**

Valid-Invalid bit

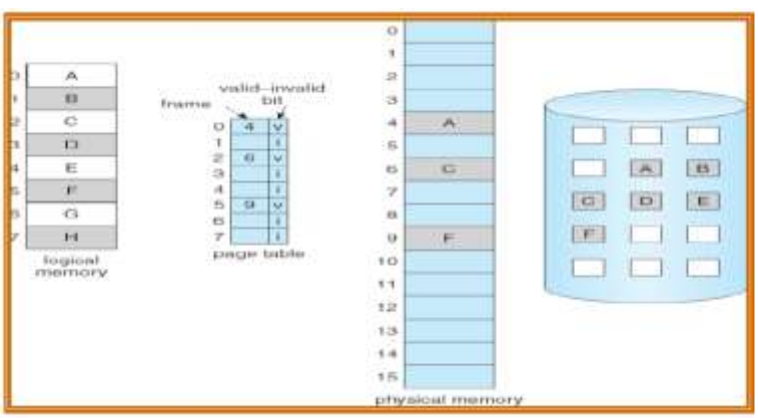
○ A valid - invalid bit is associated with each page table entry. ○ Valid → associatec is in memory.

In-Valid →

✓ invalid page

✓ valid page but is currently on the disk

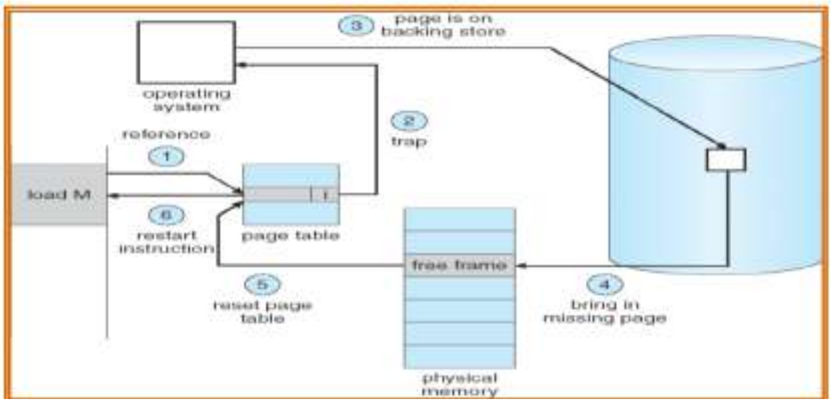
Page table when some pages are not in main memory



Page Fault

Access to a page marked invalid causes a page fault trap**.**

Steps in Handling a Page Fault



1. Determine whether the reference is a valid or invalid memory access
2. a) If the reference is invalid then terminate the process.

   b) If the reference is valid then the page has not been yet brought into main memc
3. Find a free frame.
4. Read the desired page into the newly allocated frame.
5. Reset the page table to indicate that the page is now in memory.
6. Restart the instruction that was interrupted .

Pure demand paging

○ Never bring a page into memory until it is required.

○ We could start a process with no pages in memory**.**

○ When the OS sets the instruction pointer to the 1st instruction of the process, which the non-memory resident page, then the process immediately faults for the page.

○ After this page is bought into the memory, the process continue to execute, faultin; necessary until every page that it needs is in memory.

Performance of demand paging

○ Let p be the probability of a page fault 0 □ *p* □ 1○ Effective Access Time (EAT)

EAT = (1 - *p*) x ma + *p* x page fault time. Where ma → memory access, p → Probabilit page fault (0≤ p ≤ 1)

○The memory access time denoted ma is in the range 10 to 200 ns.

○If there are no page faults then EAT = ma.

○To compute effective access time, we must know how much time is neededto service ε fault.

○A page fault causes the following sequence to occur:

1. Trap to the OS

2. Save the user registers and process state.
3. Determine that the interrupt was a page fault.
4. Check whether the reference was legal and find the location of page on disk.
5. Read the page from disk to free frame.
a. Wait in a queue until read request is serviced.
b. Wait for seek time and latency time.
c. Transfer the page from disk to free frame.
6. While waiting ,allocate CPU to some other user.
7. Interrupt from disk.
8. Save registers and process state for other users.
9. Determine that the interrupt was from disk.
7. Reset the page table to indicate that the page is now in memory.
8. Wait for CPU to be allocated to this process again.
9. Restart the instruction that was interrupted .

4.3 Process Creation

○ Virtual memory enhances the performance of creating and running processes: - Cop
Write

- Memory-Mapped Files

a) Copy-on-Write

○ **fork()** creates a child process as a duplicate of the parent process & it worked by creating copy of the parent address space for child, duplicating the pages belonging parent.

○ **Copy-on-Write (COW**) allows both parent and child processes to initially *shar* same pages in memory. These shared pages are marked as Copy-on-Write pages, mear that if either process modifies a shared page, a copy of the shared page is created.

○ **vfork():**

▪ With this the parent process is suspended & the child process uses the addres space of the parent.

▪ Because vfork() does not use Copy-on-Write, if the child process changes any page the parent's address space, the altered pages will be visible to the parent once it resume

▪ Therefore, vfork() must be used with caution, ensuring that the child process does n modify the address space of the parent.

(b)Memory - mapped files:

○ Sequential read of a file on disk uses open() , read() and write()
○ Every time a file is accessed it requires a system call and disk access. ○ Alternative me
"Memory - mapped files"

• Allowing a part of virtual address space to be logically associated with file

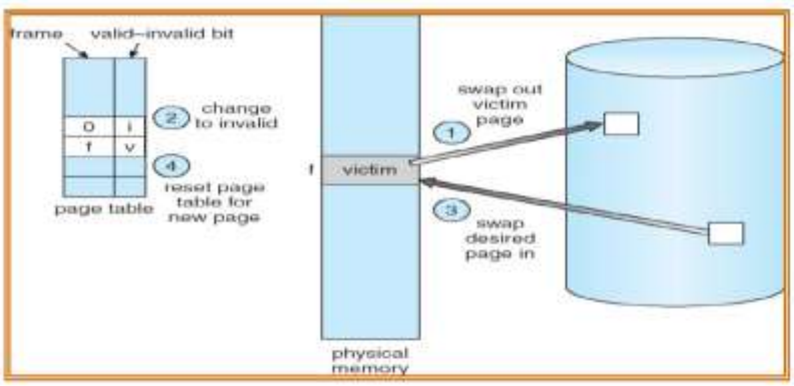
• Mapping a disk block to a page in memory.

4.4. Page Replacement

o If no frames are free, we could find one that is not currently being used & free it.
o We can free a frame by writing its contents to swap space & changing the page tab indicate that the page is no longer in memory.
o Then we can use that freed frame to hold the page for which the process faulted.

Basic Page Replacement

1. Find the location of the desired page on disk
2. Find a free frame - If there is a free frame , then use it. - If there is no free frame, us page replacement algorithm to select a **victim** frame - Write the victim page to tl disk, change the page & frame tables accordingly.
3. Read the desired page into the (new) free frame. Update the page and frame tables.
4. Restart the process



Note: If no frames are free, two page transfers are required & this situation effectively doubles the page- fault service time.

Modify (dirty) bit:

o It indicates that any word or byte in the page is modified.
o When we select a page for replacement, we examine its modify bit.
▪ If the bit is set, we know that the page has been modified & in this case we must v that page to the disk.
▪ If the bit is not set, then if the copy of the page on the disk has not been overwritt then we can avoid writing the memory page on the disk as it is already there.

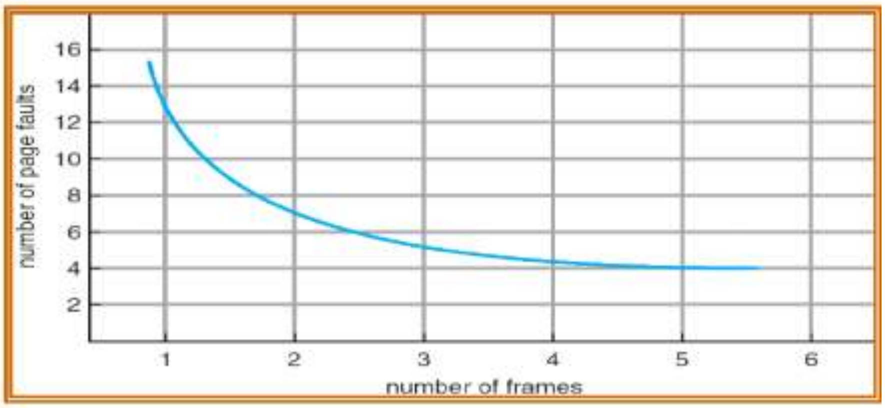
Page Replacement Algorithms

1. FIFO Page Replacement
2. Optimal Page Replacement

3. LRU Page Replacement
4. LRU Approximation Page Replacement
5. Counting-Based Page Replacement

○ We evaluate an algorithm by running it on a particular string of memory referen computing the number of page faults. The string of memory reference is called a —reference string||.The algorithm that provides less number of page faults is termed to b good one.

○ As the number of available frames increases , the number of page faults decrea This is shown in the following graph:

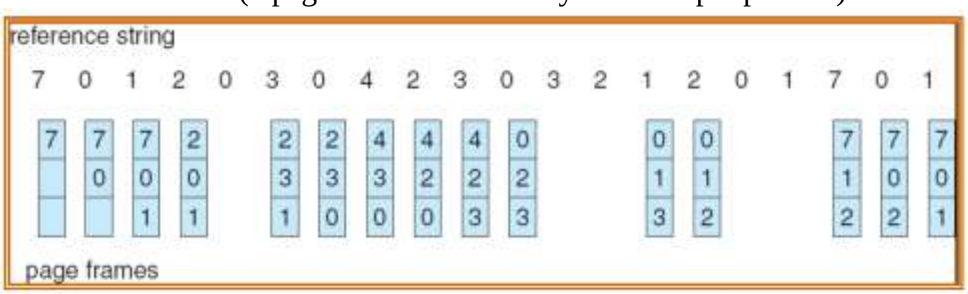


(a) FIFO page replacement algorithm

○ **Replace the oldest page.**
○ This algorithm associates with each page ,the time when that page was brougl
Example:

Reference string: 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

No.of available frames = 3 (3 pages can be in memory at a time per process)



No. of page faults = 15

Drawback:

○ FIFO page replacement algorithm _s performance is not always good. ○ To illustrate

consider the following example:

Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

○ If No.of available frames -= 3 then the no.of page faults =9
○ If No.of available frames =4 then the no.of page faults =10
○ Here the no. of page faults increases when the no.of frames increases .This is called as **Belady's Anomaly.**

Drawback:

○ It is difficult to implement as it requires future knowledge of the reference string.

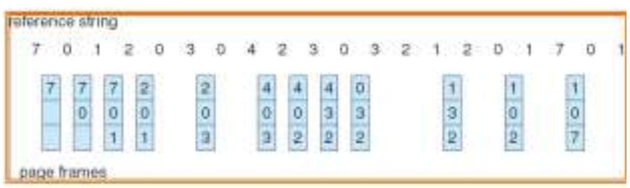
(b) Optimal page replacement algorithm

○ **Replace the page that will not be used for the longest period of time. Example:**

Reference string: 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1 No.of available frames =3

(c) LRU(Least Recently Used) page replacement algorithm

○ **Replace the page that has not been used for the longest period of time.**
Example: Reference string: 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1 No.of available frames

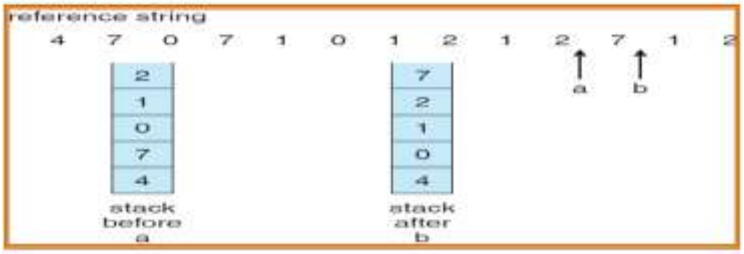


No. of page faults = 12

○ LRU page replacement can be implemented using
1. Counters
✓ Every page table entry has a time-of-use field and a clock or counter is associa
with the CPU.
✓ The counter or clock is incremented for every memory reference. ✓ Each time a pa
referenced , copy the counter into the time-of-use field.
✓ When a page needs to be replaced, replace the page with the smallest counter
2. **Stack**
✓ Keep a stack of page numbers
✓ Whenever a page is referenced, remove the page from the stack and put it on top
stack.
✓ When a page needs to be replaced, replace the page that is at the bottom of the

stack.(LRU page)

Use of A Stack to Record The Most Recent Page References



(d) LRU Approximation Page Replacement

○ Reference bit

✓ With each page associate a reference bit, initially set to 0 ✓ When page is referenced bit is set to 1

○ When a page needs to be replaced, replace the page whose reference bit is 0 ○ The of use is not known , but we know which pages were used and which were not used

(i) Additional Reference Bits Algorithm

○ Keep an 8-bit byte for each page in a table in memory.

○ At regular intervals , a timer interrupt transfers control to OS.

○ The OS shifts reference bit for each page into higher- order bit shifting the other b right 1 bit and discarding the lower-order bit.

Example:

○ If reference bit is 00000000 then the page has not been used for 8 time periods.

○ If reference bit is 11111111 then the page has been used atleast once each time peri

○ If the reference bit of page 1 is 11000100 and page 2 is 01110111 then page 2 is the I page.

(ii) Second Chance Algorithm

○ Basic algorithm is FIFO

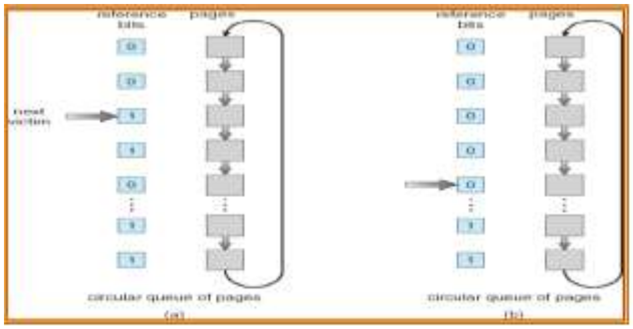
○ When a page has been selected , check its reference bit.

✓ If 0 proceed to replace the page

✓ If 1 give the page a second chance and move on to the next FIFO page.

✓ When a page gets a second chance, its reference bit is cleared and arrival time reset to current time.

✓ Hence a second chance page will not be replaced until all other pages are replace



(iii) Enhanced Second Chance Algorithm

o Consider both reference bit and modify bit

o There are four possible classes

1. (0,0) - neither recently used nor modified → Best page to replace
2. (0,1) - not recently used but modified → page has to be written out before replacement.
3. (1,0) - recently used but not modified → page may be used again
4. (1,1) - recently used and modified → page may be used again and page has to b written to disk

(e) Counting-Based Page Replacement

o Keep a counter of the number of references that have been made to each page

1.**Least Frequently Used (LFU)Algorithm**: replaces page withsmallest count

2.**Most Frequently Used (MFU)Algorithm**: replaces page withlargest count

✓ It is based on the argument that the page with the smallest count was proba just brought in and has yet to be used

Page Buffering Algorithm

o These are used along with page replacement algorithms to improve their performance

Technique 1:

o A pool of free frames is kept.

o When a page fault occurs, choose a victim frame as before. o Read the desired page free frame from the pool

o The victim frame is written onto the disk and then returned to the pool of free fr

Technique 2:

o Maintain a list of modified pages.

o Whenever the paging device is idles, a modified is selected and written to disk a modify bit is reset.

Technique 3:

o A pool of free frames is kept.

o Remember which page was in each frame.

o If frame contents are not modified then the old page can be reused directly from t free frame pool when needed

4.5 Allocation of Frames

There are two major allocation schemes

✓ Equal Allocation

✓ Proportional Allocation

o **Equal allocation**

✓ If there are n processes and m frames then allocate m/n frames to each process.

✓ **Example:** If there are 5 processes and 100 frames, give each process 20 frames.

o **Proportional allocation**

✓ Allocate according to the size of process Let s_i be the size of process i.

Let m be the total no. of frames Then $S = \sum s_i a_i = s_i / S * m$ where a_i is the no.of frames allocated to process i.

Global vs. Local Replacement

o **Global replacement** - each process selects a replacement frame from the set of all fi one process can take a frame from another.

o **Local replacement** - each process selects from only its own set of allocated frames

4.6 Thrashing

o High paging activity is called **thrashing**.

o If a process does not have —enough|| pages, the page-fault rate is very high. This leads

✓ low CPU utilization

✓ operating system thinks that it needs to increase the degree of multiprogram

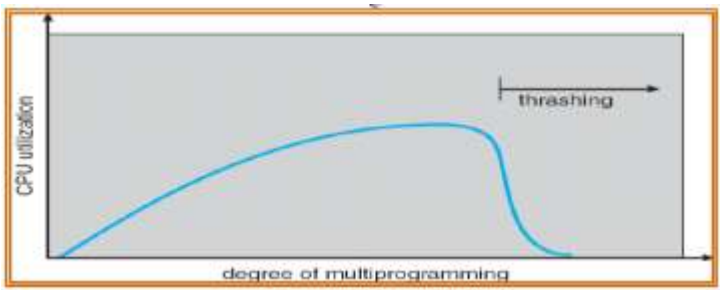
✓ another process is added to the system

o When the CPU utilization is low, the OS increases the degree of multiprogramming.

o If global replacement is used then as processes enter the main memory they tend to frames belonging to other processes.

o Eventually all processes will not have enough frames and hence the page fault r becomes very high.

o Thus swapping in and swapping out of pages only takes place. o This is the cause of thrashing.



○ To **limit thrashing**, we can use a **local replacement** algorithm. ○ To prevent thrash there are two methods namely ,

✓ Working Set Strategy

✓ Page Fault Frequency

1. Working-Set Strategy

○ It is based on the assumption of the model of locality.

○ Locality is defined as the set of pages actively used together.

○ Working set is the set of pages in the most recent □ page references ○ □ is the work window.

✓ if □ too small , it will not encompass entire locality

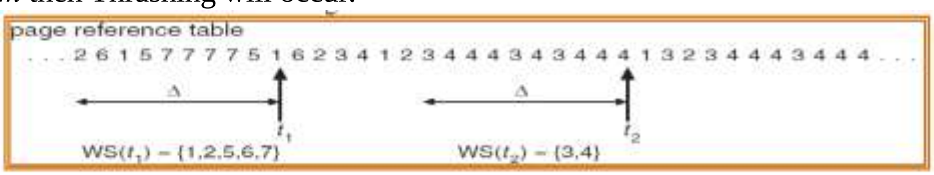
✓ if □ too large ,it will encompass several localities

✓ if □ = □ □ it will encompass entire program

○ $D = \square\ WSS_i$

✓ Where WSS_i is the working set size for process i. ✓ D is the total demand of frames

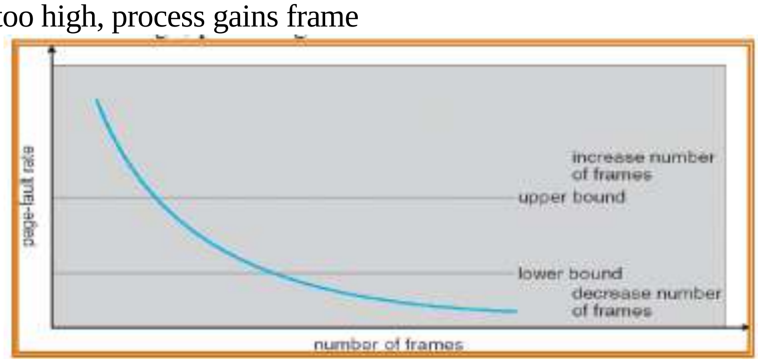
○ if $D > m$ then Thrashing will occur.



2. Page-Fault Frequency Scheme

○ If actual rate too low, process loses frame

○ If actual rate too high, process gains frame



Other Issues

○ **Prepaging**

✓ To reduce the large number of page faults that occurs at process startup

✓ Prepage all or some of the pages a process will need, before they are referenced

✓ But if prepaged pages are unused, I/O and memory are wasted

○**Page Size**

Page size selection must take into consideration:

○ fragmentation

○ table size

○ I/O overhead

○ locality

○**TLB Reach**

✓ TLB Reach - The amount of memory accessible from the TLB ✓ TLB Reach = (TLI X (Page Size)

✓ Ideally, the working set of each process is stored in the TLB. Otherwise th a high degree of page faults.

✓ Increase the Page Size. This may lead to an increase in fragmentation as not all applications require a large page size

✓ Provide Multiple Page Sizes. This allows applications that require larger page si the opportunity to use them without an increase in fragmentation.

○**I/O interlock**

✓ Pages must sometimes be locked into memory

✓ Consider I/O. Pages that are used for copying a file from a device must be l from being selected for eviction by a page replacement algorithm.

4.7 File Concept

A file is a named collection of related information that is recorded on secondary storage.

• From a user's perspective, a file is the smallest allotment of logical secondary storage is, data cannot be written to secondary storage unless they are within a file.

Examples of files:

• A text file is a sequence of characters organized into lines (and possibly pages). A sour file is a sequence of subroutines and functions, each of which is further organized a declarations followed by executable statements. An object file is a sequence of bytes org into blocks understandable by the system's linker. An executable file is a series of code sections that the loader can bring into memory and execute.

File Attributes

• **Name:** The symbolic file name is the only information kept in human readable form.

• **Identifier:** This unique tag, usually a number identifies the file within the file syste is the non-human readable name for the file.

• **Type:** This information is needed for those systems that support different types.
• **Location:** This information is a pointer to a device and to the location of the file on tl device.
• **Size:** The current size of the file (in bytes, words or blocks)and possibly the maximun allowed size are included in this attribute.
• **Protection:** Access-control information determines who can do reading, writing, executing and so on.
• **Time, date and user identification:** This information may be kept for creation, last modification and last use. These data can be useful for protection, security and usage monitoring.

File Operations

- Creating a file
- Writing a file
- Reading a file
- Repositioning within a file
- Deleting a file
- Truncating a file

File types

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 languages
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, so, dll, mpeg, mov, rm	Libraries of routines for programmers
print or view	arc, zip, tar	ASCII or binary file in a format for printing or viewing
Archive	arc, zip, tar	Related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm	Binary file containing audio or A/V information

File Structure

- All disk I/O is performed in units of one block (physical record) size which will exa match the length of the desired logical record.

- Logical records may even vary in length. **Packing** a number of logical records physical blocks is a common solution to this problem.

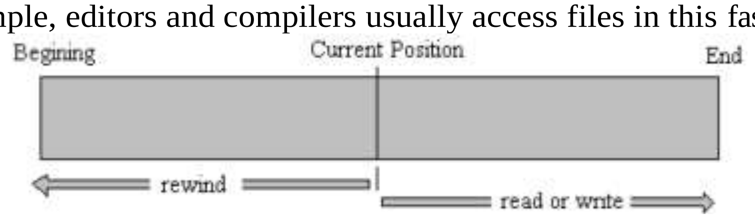
- For example, the UNIX operating system defines all files to be simply a stream bytes. Each byte is individually addressable by its offset from the beginning (or end) o file. In this case, the logical records are 1 byte. The file system automatically packs and unpacks bytes into physical disk blocks -say, 512 bytes per block - as necessary.

- The logical record size, physical block size, and packing technique determine how ma logical records are in each physical block. The packing can be done either by the use application program or by the operating system.

4.8 Access Methods

1. Sequential Access

The simplest access method is sequential access. Information in the file is processed in order, one record after the other. This mode of access is by far the most common; for example, editors and compilers usually access files in this fashion.



The bulk of the operations on a file is reads and writes. A read operation rea next portion of the file and automatically advances a file pointer, which tracks the I/O location. Similarly, a write appends to the end of the file and advances to the end newly written material (the new end of file). Such a file can be reset to the beginning and some systems, a program may be able to skip forward or back ward n records, for s integer n-perhaps only for n=1. Sequential access is based on a tape model of a and works as well on sequential-access devices as it does on random - access ones.

2. Direct Access

Another method is direct access (or relative access). A file is made up of fix length logical records that allow programs to read and write records rapidly in no partic order. The direct- access methods is based on a disk model of a file, since disks allow random access to any file block.

For direct access, the file is viewed as a numbered sequence of blocks or rec A direct-access file allows arbitrary blocks to be read or written. Thus, we may read blo 14, then read block 53, and then write block7. There are no restrictions on the orde reading or writing for a direct-access file.

Direct - access files are of great use for immediate access to large amounts of information. Database is often of this type. When a query concerning a particula subject arrives, we compute which block contains the answer, and then read that block directly to provide the desired information.

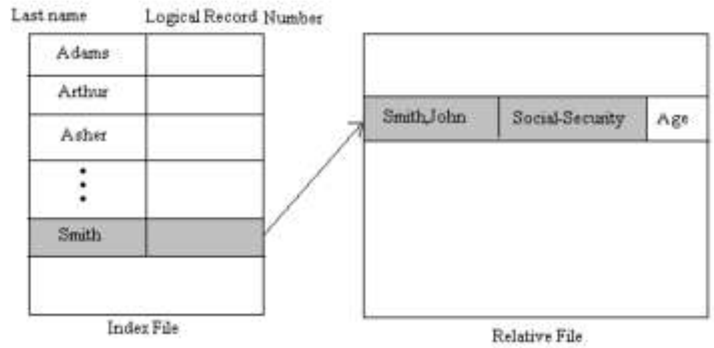
As a simple example, on an air line - reservation system, we might store all th information about a particular flight (for example, flight 713) in the block identif the flight number.

Thus, the number of available seats for flight 713 is stored in block 713 of the reservation file. To store information about a larger set, such as people, we might compute a hash function on the people's names, or search a small inmemory inde determine a block to read and search.

Sequential access	Implementation for direct access
Reset	Cp=0;
Read next	Read cp; Cp=cp+1;
Write next	Write cp; Cp=cp+1;

3. Other Access methods

Other access methods can be built on top of a direct - access method these me generally involve the construction of an index for the file. The index like an index in th back of a book contains pointers to the various blocks in find a record in the file. W search the index, and then use the pointer to access the file directly and the find the desi record.



With large files, the index file itself may become too large to be kept in memory. One solution is to create an index for the index file. The primary index fil would contain pointers to secondary index tiles, which would point to the actual items.

4.9 Directory Structure

There are five directory structures. They are

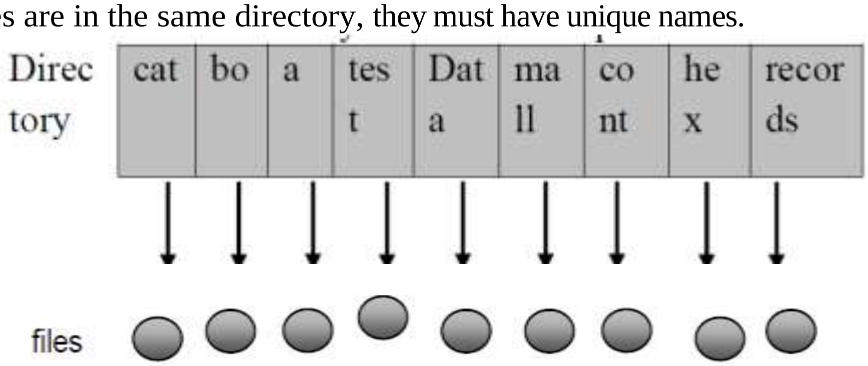
1. Single-level directory
2. Two-level directory
3. Tree-Structured directory
4. Acyclic Graph directory
5. General Graph directory

1. Single - Level Directory

• The simplest directory structure is the single- level directory. • All files are contained i same directory.

• **Disadvantage:**

➢ When the number of files increases or when the system has more than one use since all files are in the same directory, they must have unique names.



2. Two - Level Directory

• In the two level directory structures, each user has her own user file directory (UI

• When a user job starts or a user logs in, the system's master file directory (M is searched. The MFD is indexed by user name or account number, and each entry po the UFD for that user.

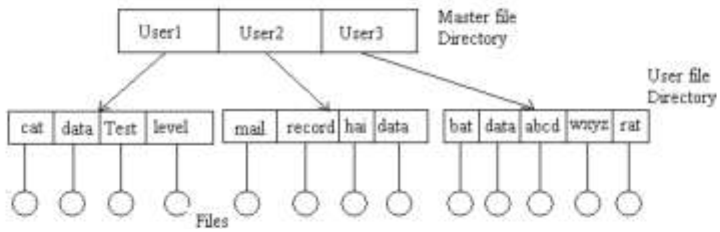
• When a user refers to a particular file, only his own UFD is searched.

• Thus, different users may have files with the same name.

• Although the two - level directory structure solves the name-collision problem

• **Disadvantage:**

➢ Users cannot create their own sub-directories.



3. Tree - Structured Directory

• A tree is the most common directory structure.

• The tree has a root directory. Every file in the system has a unique path name.

• A **path name** is the path from the root, through all the subdirectories to a specified file.

• A directory (or sub directory) contains a set of files or sub directories. • A directory simply another file. But it is treated in a special way. • All directories have the same int format.

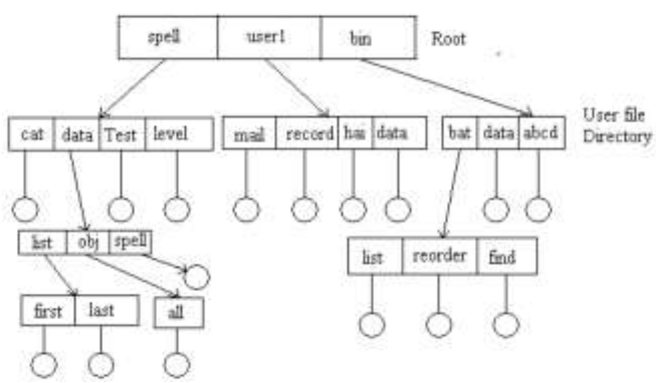
• One bit in each directory entry defines the entry as a file (0) or as a subdirectory (

• Special system calls are used to create and delete directories.

• Path names can be of two types: absolute path names or relative path names.

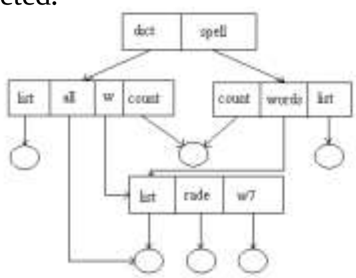
• An absolute path name begins at the root and follows a path down to the specified file giving the directory names on the path.

• A relative path name defines a path from the current directory.



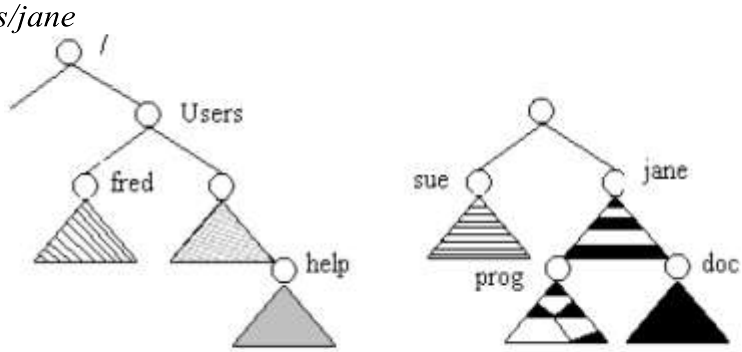
4. Acyclic Graph Directory.

- An acyclic graph is a graph with no cycles.
- To implement shared files and subdirectories this directory structure is used.
- An acyclic - graph directory structure is more flexible than is a simple tree structure it is also more complex. In a system where sharing is implemented by symbolic link, situation is somewhat easier to handle. The deletion of a link does not need to affec original file; only the link is removed.
- Another approach to deletion is to preserve the file until all references to it are deleted. implement this approach, we must have some mechanism for determining that the l reference to the file has been deleted.



<u>4.10 File System Mounting</u>

- Just as a file must be opened before it is used, a file system must be mounted before i be available to processes on the system.
- The mount procedure is straightforward. The operating system is given the name of t device, and the location within the file structure at which to attach the File system (o mount point).
- A mount point is an empty directory at which the mounted file system will be attache
- For instance, on a UNIX system, a file system containing user's home directorie might be mounted as /home; then to access the directory structure within that file system could precede the directory names with /*home, as in /home/jane.*
- Mounting that file system under/*user* would result in the pathname/*users/jane*



File System (a) Existing (b) Unmounted Partation

• The operating system verifies that the devices contain a valid file system.

• It does so by asking the device driver to read the device directory and verifying the directory was the expected format.

• Finally, the operating system notes in its directory structure that a file system is mounted at the specified mount point.

4.11 File Sharing

1. Multiple Users:

• When an operating system accommodates multiple users, the issues of file sharing, f naming and file protection become preeminent.

• The system either can allow user to access the file of other users by default, or it may require that a user specifically grant access to the files.

• These are the issues of access control and protection.

• To implementing sharing and protection, the system must maintain more file and directo attributes than a on a single-user system.

• The owner is the user who may change attributes, grand access, and has the most con over the file or directory.

• The group attribute of a file is used to define a subset of users who may share acces the file.

• Most systems implement owner attributes by managing a list of user names and assoc user identifiers (user Ids).

• When a user logs in to the system, the authentication stage determines the appropriate u ID for the user. That user ID is associated with all

2. Remote File System:

• Networks allowed communications between remote computers.

• Networking allows the sharing or resource spread within a campus or even around th world.

• User manually transfer files between machines via programs like **ftp**.

• A **distributed file system** (DFS) in which remote directories is visible from the local machine.

• The **World Wide Web**: A browser is needed to gain access to the remote file and separate operations (essentially a wrapper for ftp) are used to transfer files.

a) The client-server Model:

• Remote file systems allow a computer to a mount one or more file systems from one more remote machines.

• A server can serve multiple clients, and a client can use multiple servers, dependi the implementation details of a given client -server facility. • Client identification is m difficult. Clients can be specified by their network name or other identifier, such as I address, but these can be spoofed (or imitate). An unauthorized client can spoof the serv deciding that it is authorized, and the unauthorized client could be allowed access.

b) Distributed Information systems:

- Distributed information systems, also known as distributed naming service, have be devised to provide a unified access to the information needed for remote computing.
- Domain name system (DNS) provides host-name-to-network address translations for their entire Internet (including the World Wide Web).
- Before DNS was invented and became widespread, files containing the same informati were sent via e-mail of ftp between all networked hosts.

c) Failure Modes:

- **Redundant arrays of inexpensive disks** (**RAID**) can prevent the loss of a disk from resulting in the loss of data.
- Remote file system has more failure modes. By nature of the complexity of networkir system and the required interactions between remote machines, many more problems c interfere with the proper operation of remote file systems.

d) Consistency Semantics:

- It is characterization of the system that specifies the semantics of multiple users acc a shared file simultaneously.
- These semantics should specify when modifications of data by one user are observab other users.
- The semantics are typically implemented as code with the file system.
- A series of file accesses (that is reads and writes) attempted by a user to the same file always enclosed between the open and close operations.
- The series of access between the open and close operations is a **file session**.

(i) UNIX Semantics:

The UNIX file system uses the following consistency semantics:

1. Writes to an open file by a user are visible immediately to other users that have this f open at the same time.
2. One mode of sharing allows users to share the pointer of current location into tl Thus, the advancing of the pointer by one user affects all sharing users.

(ii) Session Semantics:

The Andrew file system (AFS) uses the following consistency semantics:

1. Writes to an open file by a user are not visible immediately to other users that have same file open simultaneously.
2. Once a file is closed, the changes made to it are visible only in sessions startin later. Already open instances of the file do not reflect this change.

(iii) Immutable -shared File Semantics:

- Once a file is declared as shared by its creator, it cannot be modified. • An immutabl

has two key properties:
✓ Its name may not be reused and its contents may not be altered.

4.12 File Protection

(i) Need for file protection.

• When information is kept in a computer system, we want to keep it safe from **phy damage** (reliability) and **improper access** (protection).

• Reliability is generally provided by duplicate copies of files. Many compute have systems programs that automatically (or though computer-operator intervention) disk files to tape at regular intervals (once per day or week or month) to maintain copy should a file system be accidentally destroyed.

• File systems can be damaged by hardware problems (such as errors in reading o writing), power surges or failures, head crashes, dirt, temperature extremes, and vandalis Files may be deleted accidentally. Bugs in the file-system software can also cause file con to be lost.

• Protection can be provided in many ways. For a small single-user system, we migh provide protection by physically removing the floppy disks and locking them in a de drawer or file cabinet. In a multi-user system, however, other mechanisms are need

(ii) Types of Access

• Complete protection is provided by prohibiting access. • Free access is provided with protection.

• Both approaches are too extreme for general use.

• What is needed is **controlled access**.

• Protection mechanisms provide controlled access by limiting the types of file acces can be made. Access is permitted or denied depending on several factors, one of wh is the type of access requested. Several different types of operations may be controlle

1. **Read:** Read from the file.
2. **Write:** Write or rewrite the file.
3. **Execute:** Load the file into memory and execute it.
4. **Append:** Write new information at the end of the file.
5. **Delete:** Delete the file and free its space for possible reuse.
6. **List:** List the name and attributes of the file.

(iii) Access Control

• Associate with each file and directory an access-control list (ACL) specifying user name and the types of access allowed for each user.

• When a user requests access to a particular file, the operating system checks the access associated with that file. If that user is listed for the requested access, the access is allov

Otherwise, a protection violation occurs and the user job is denied access to the file.

- This technique has two undesirable consequences:

➢ Constructing such a list may be a tedious and unrewarding task, especially if v not know in advance the list of users in the system.

➢ The directory entry, previously of fixed size, now needs to be of variable size, resulting in more complicated space management.

- 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 group.

➢ **Universe:** All other users in the system constitute the universe.

UNIT- V FILE SYSTEMS

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

5.1 File System Structure

• **Disk** provide the bulk of secondary storage on which a file system is maintained.

• **Characteristics of a disk:**

1. They can be rewritten in place, it is possible to read a block from the disk, to moc the block and to write it back into the same place.
2. They can access directly any given block of information to the disk.

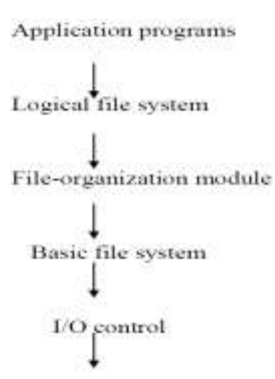
• To produce an efficient and convenient access to the disk, the operating system impos one or more file system to allow the data to be stored, located and retrieved easily.

• The file system itself is generally composed of many different levels. Each level in the design uses the features of lower level to create new features for use by higher levels.

Layered File System

• The **I/O control** consists of device drivers and interrupt handlers to transfer inform between the main memory and the disk system .

• The **basic file system** needs only to issue generic commands to the appropriate devic driver to read and write physical blocks on the disk. Each physical block is identified b numeric disk address (for example, drive -1, cylinder 73, track 2, sector 10)



• The **file-organization module** knows about file and their logical blocks, as well as ph blocks. By knowing the type of file allocation used and the location of the file, the file organization module can translate logical block address to physical block addresses for basic fie system to transfer . The file-organization module also includes the free-space manager, which tracks unallocated blocks and provides these blocks to the file-organizati module when requested.

• The **logical file system** manages metadata information. Metadata includes all of the system structure, excluding the actual *data* (or contents of the files). The logical file sys manages the directory structure to provide the file-organization module with the information the latter needs, given a symbolic file name. It maintains file structure , v control blocks. A **file control block** (FCB) contains information about the file , inclu ownership, permissions, and location of the file contents. The logical file system is als responsible for protection and security.

5.2 File System Implementation

•Several-on-disk and in-memory structures are used to implement a filesystem

•The on-disk structures include:

1. A **boot control block** can contain information needed by the system to boot an operating from that partition. If the disk does not contain an operating System, this can be empty. It is typically the first block of a partition. In **UFS**, this is called the **b block**; In **NTFS**, it is **partition boot sector**.

2. A **partition control block** contains partition details such as the number of blocks in partition, size of the blocks, free-block count and free block pointers and free FCB and FCB pointers. In **UFS** this is called a **super block**; in **NTFS**, it is the **Master File Table**.

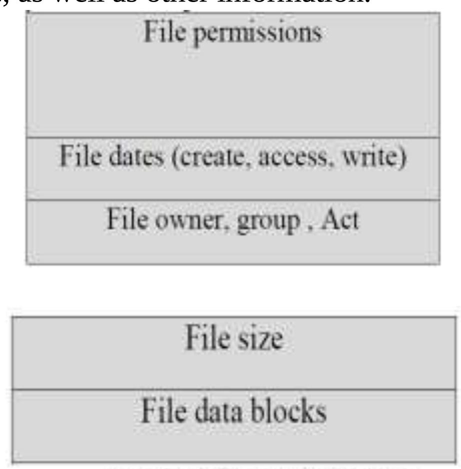
3. A **directory structure** is used to organize the files.

4. An **FCB** contains many of the files details, including file permissions , ownership, and location of the data blocks. IN **UFS** this called the **inode**. In NTFS, this informatic

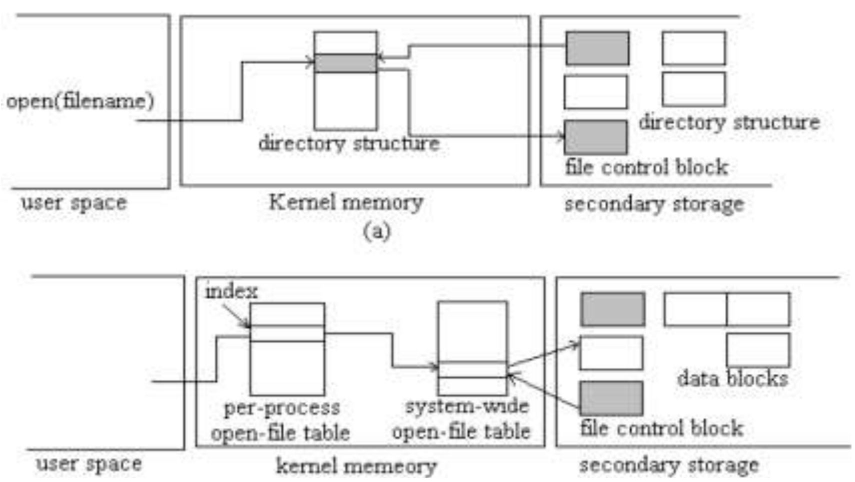
actually stored within the Master File Table, which uses a relational database structure, w
row per file.

• The in-memory structures include:

1. An **in-memory partition table** containing , information about each mounted partition.
2. An **in-memory directory structure** that hold s the directory information of recently accessed directories.
3. The **system-wide open-file table** contains a copy of the FCB of each open files, a
as other information.
4. The **per-process open-file table** contains a pointer tot eh appropriate entry in the
systems-wide open file table, as well as other information.



A typical file control block



5.3 Directory Implementation

1. Linear List

• The simplest method of implementing a directory is to use a linear list of file names w pointer to the data blocks.

• A linear list of directory entries requires a linear search to find a particular entry.

• This method is simple to program but time- consuming to execute. To create a new file must first search the but time - consuming to execute.

• The real disadvantage of a linear list of directory entries is the linear search to find a

2. Hash Table

• In this method, a linear list stores the directory entries, but a hash data structure used.

• The hash table takes a value computed from the file name and returns a pointer to file name in the linear list.

• Therefore, it can greatly decrease the directory search time.

• Insertion and deleting are also fairly straight forward, although some provision be made for collisions - situation where two file names hash to the same location.

• The major difficulties with a hash table are its generally fixed size and the dependen the hash function on that size.

5.4 Allocation Methods

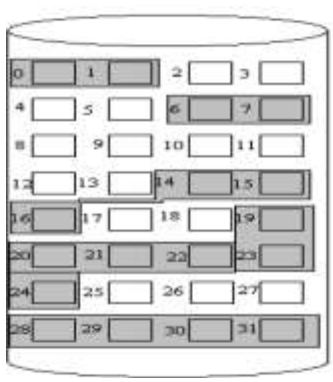
• The main problem is how to allocate space to these files so that disk space is utilize effectively and files can be accessed quickly .

• There are there major methods of allocating disk space:

1. Contiguous Allocation
2. Linked Allocation
3. Indexed Allocation

1. Contiguous Allocation

• The contiguous - allocation method requires each file to occupy a set of contiguo blocks on the disk.



Directory

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

• Contiguous allocation of a file is defined by the disk address and length (in block un

of the first block. If the file is n blocks long and starts at location b, then it occupi
blocks b,. b+1, b+2,….,b+n-1.

• The directory entry for each file indicates the address of the starting block and the le of the area allocated for this file.

Disadvantages:

1. Finding space for a new file.

• The contiguous disk space-allocation problem suffer from the problem of external fragmentation. As file are allocated and deleted, the free disk space is broken into chun becomes a problem when the largest contiguous chunk is insufficient for a request; s is fragmented into a number of holes, no one of which is large enough to store the data.

2. Determining how much space is needed for a file.

• When the file is created, the total amount of space it will need must be found an alloc how does the creator know the size of the file to be created?

• If we allocate too little space to a file, we may find that file cannot be extended other possibility is to find a larger hole, copy the contents of the file to the new space release the previous space. This series of actions may be repeated as long as space exists, although it can be time consuming. However, in this case, the user never ne be informed explicitly about what is happening ; the system continues despite the problem, although more and more slowly.

• Even if the total amount of space needed for a file is known in advance pre-allocation be inefficient.

• A file that grows slowly over a long period (months or years) must be allocate enough space for its final size, even though much of that space may be unused for a time the file, therefore has a large amount of internal fragmentation.

To overcome these disadvantages:

• Use a modified contiguous allocation scheme, in which a contiguous chunk of space c as an **extent** is allocated initially and then, when that amount is not large enough another chunk of contiguous space an extent is added to the initial allocation.

• Internal fragmentation can still be a problem if the extents are too large, and external fragmentation can be a problem as extents of varying sizes are allocated and deallo

2. Linked Allocation

• Linked allocation solves all problems of contiguous allocation.

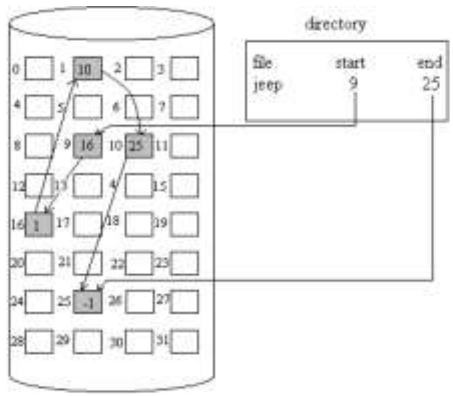
• With linked allocation, each file is a linked list of disk blocks, the disk blocks m scattered any where on the disk.

• The directory contains a pointer to the first and last blocks of the file. For example, of five blocks might start at block 9, continue at block 16, then block 1, block 10, and finally bock 25.

• Each block contains a pointer to the next block. These pointers are not made available user.

• There is no external fragmentation with linked allocation, and any free block on the free list can be used to satisfy a request.

• The size of a file does not need to the declared when that file is created. A file can continue to grow as long as free blocks are available consequently, it is never necessary compacts disk space.



Disadvantages:

1. 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 th pointers until we get to the ith block. Each aces to a pointer requires a disk read, ar sometimes a disk seek consequently, it is inefficient to support a direct- access capability linked allocation files.

2. Space required for the pointers

• If a pointer requires 4 bytes out of a 512-byte block, then 0.78 percent of the disk is bei used for pointers, rather than for information.

• Solution to this problem is to collect blocks into multiples, called **clusters**, and to al the clusters rather than blocks. For instance, the file system may define a clusters as 4 blocks, and operate on the disk in only cluster units.

3. Reliability

• Since the files are linked together by pointers scattered all over the disk hardware failure might result in picking up the wrong pointer. This error cou result in linking into the free- space list or into another file. Partial solution are to u doubly linked lists or to store the file names in a relative block number in each block; however, these schemes require even more over head for each file.

File Allocation Table(FAT)

•An important variation on the linked allocation method is the use of a fileallocation table(FAT).

•This simple but efficient method of disk- space allocation is used by the MS-DOS and operating systems.

•A section of disk at beginning of each partition is set aside to contain thetable.

•The table has entry for each disk block, and is indexed by block number.

•The FAT is much as is a linked list.

•The directory entry contains the block number the first block of the file.

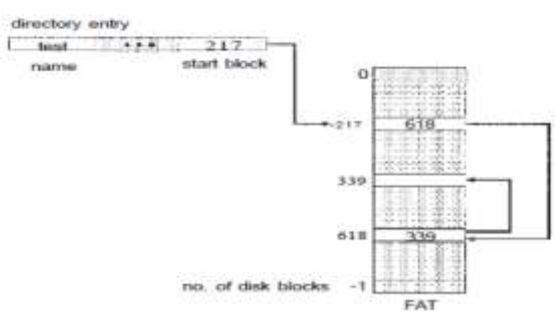
•The table entry indexed by that block number contains the block number ofthe next bl the file.

•This chain continues until the last block which has a special end - of - filevalue as the entry.

•Unused blocks are indicated by a 0 table value.

• Allocating a new block file is a simple matter of finding the first 0 - valued table entry replacing the previous end of file value with the address of the new block.

• The 0 is replaced with the end - of - file value, an illustrative example is the FAT structure for a file consisting of disk blocks 217,618, and 339.



3. Indexed Allocation

• Linked allocation solves the external - fragmentation and size- declaration problems contiguous allocation.

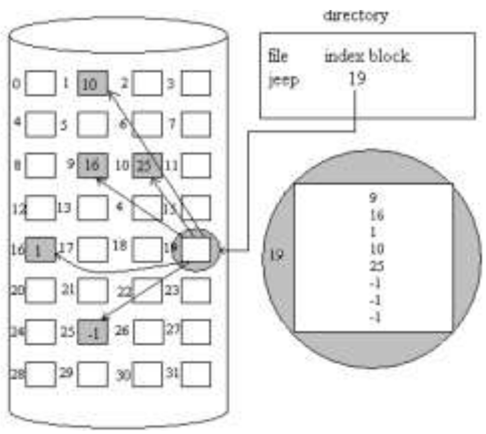
• Linked allocation cannot support efficient direct access, since the pointers to the block scattered with the blocks themselves all over the disk and need to be retrieved in ord

• Indexed allocation solves this problem by bringing all the pointers together into one location: the **index block**.

• Each file has its own index block, which is an array of disk- blockaddresses.

• The ith entry in the index block points to the ith block of the file. • The directory cont the address of the index block .

• To read the ith block, we use the pointer in the ith index - block entry to find and read t desired block this scheme is similar to the paging scheme .



• When the file is created, all pointers in the pointers in the index block are set to nil. w the ith block is first written, a block is obtained from the free space manager, and its ac is put in the ith index - block entry.

• Indexed allocation supports direct access, without suffering from external fragment because any free block on the disk may satisfy a request for more space.

Disadvantages

1.Pointer Overhead

• Indexed allocation does suffer from wasted space. The pointer over head of the index is generally greater than the pointer over head of linked allocation.

2. Size of Index block

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

• **Linked Scheme:** An index block is normally one disk block. Thus, it can be read and written directly by itself. To allow for large files, we may link together several inde blocks.

•**Multilevel index:** A variant of the linked representation is to use a first levelindex bloc point to a set of second - level index blocks.

•**Combined scheme:**

o Another alternative, used in the UFS, is to keep the first, say, 15 pointers of the block in the file's inode.

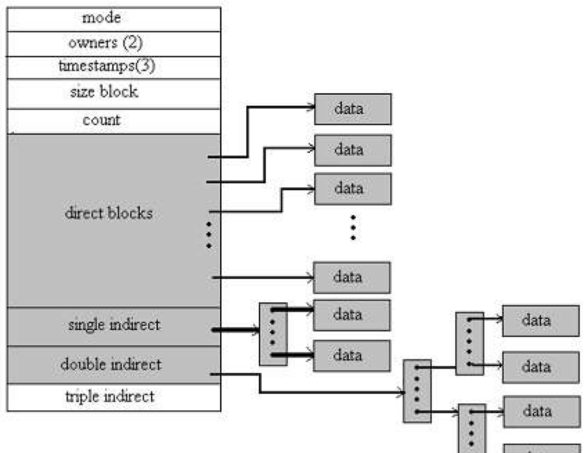
o The first 12 of these pointers point to direct blocks; that is for small (no more than blocks) files do not need a separate index block

o The next pointer is the address of a single indirect block.

✓ The single indirect block is an index block, containing not data, but rather the addres
blocks that do contain data.

o Then there is a double indirect block pointer, which contains the address of a
that contain pointers to the actual data blocks. The last pointer would contain pointers t
actual data blocks.

o The last pointer would contain the address of a triple indirect block.



The UNIX inode

5.5 Free-space Management

- Since disk space is limited, we need to reuse the space from deleted files for new files,
possible.
- To keep track of free disk space, the system maintains a free-space list.
- The free-space list records all free disk blocks - those not allocated to some file or directory.
- To create a file, we search the free-space list for the required amount of space, an
allocate that space to the new file.
- This space is then removed from the free-space list.
- When a file is deleted, its disk space is added to the free-space list.

1. Bit Vector

- The free-space list is implemented as a bit map or bit vector.

- Each block is represented by 1 bit. If the block is free, the bit is 1; if the block is allocated, the bit is 0.

- For example, consider a disk where block 2,3,4,5,8,9,10,11,12,13,17,18,25,26 and 27 a free, and the rest of the block are allocated. The free space bit map would be001111001111110001100000011100000 …

• The main **advantage** of this approach is its relatively simplicity and efficiency finding the first free block, or n consecutive free blocks on the disk.

2. Linked List

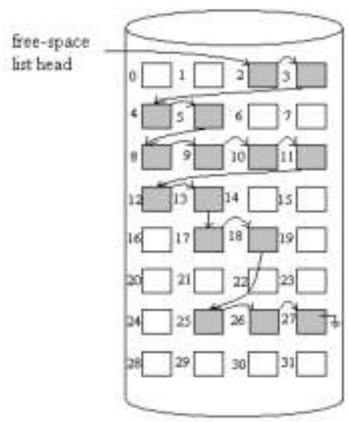
• Another approach to free-space management is to link together all the free disk bloc keeping a pointer to the first free block in a special location on the disk and caching it i memory.

• This first block contains a pointer to the next free disk block, and so on.

• In our example, we would keep a pointer to block 2, as the first free block. Block 2 contain a pointer to block 3, which would point to block 4, which would point to block which would point to block 8, and so on.

• However, this scheme is not efficient; to traverse the list, we must read each block, wh requires substantial I/O time.

• The FAT method incorporates free-block accounting data structure. No separate method is needed.



3. Grouping

• A modification of the free-list approach is to store the addresses of n free blocks in first free block.

• The first n-1 of these blocks are actually free.

• The last block contains the addresses of another n free blocks, and so on.

• The importance of this implementation is that the addresses of a large number of blocks can be found quickly.

4. Counting

• We can keep the address of the first free block and the number n of free contiguo 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 overal will be shorter, as long as the count is generally greater than

Recovery

- Files and directories are kept both in main memory and on disk, and care must be ta to ensure that system failure does not result in loss of data or in data inconsistency.

1. Consistency Checking

- The directory information in main memory is generally more up to date than is the corresponding information on the disk, because cached directory information is no necessarily written to disk as soon as the update takes place.
- Frequently, a special program is run at reboot time to check for and correct disk inconsistencies.
- The consistency checker—a systems program such as
- chkdsk in MS-DOS—compares the data in the directory structure with the data blo on disk and tries to fix any inconsistencies it finds. The allocation and free-spac management algorithms dictate what types of problems the checker can find and how successful it will be in fixing them.

2. Backup and Restore

- Magnetic disks sometimes fail, and care must be taken to ensure that the data lost such a failure are not lost forever. To this end, system programs can be used to **back** up from disk to another storage device, such as a floppy disk, magnetic tape, optical disk, other hard disk.
- Recovery from the loss of an individual file, or of an entire disk, may then be a matt **restoring** the data from backup.

A **typical backup schedule** may then be as follows:

Day 1: Copy to a backup medium all files from the disk. This is called a **full bacl**

Day 2: Copy to another medium all files changed since day 1. This is an **incremental backup.**

Day 3: Copy to another medium all files changed since day 2.

Day N: Copy to another medium all files changed since day N— 1. Then go back to Da

Log-Structured File Systems

- Computer scientists often find that algorithms and technologies originally used in one area are equally useful in other areas.
- These logging algorithms have been applied successfully to the problem ofconsistency checking.

•The resulting implementations are known as **log-based transaction-oriented** (or **journaling)** file systems.

•Fundamentally, all metadata changes are written sequentially to a log.

•Each set of operations for performing a specific task is a transaction.

•Once the changes are written to this log, they are considered to becommitted, a system call can return to the user process, allowing it tocontinue execution.

•As the changes are made, a pointer is updated to indicate which actions havecomplete which are still incomplete.

•When an entire committed transaction is completed, it is removed from thelog file, wh actually a circular buffer.

• A circular buffer writes to the end of its space and then continues at the beginning overwriting older values as it goes. If the system crashes, the log file will contain zero o transactions.

5.6 Kernel I/O Subsystem

Kernels provide many services related to I/O.

✓ One way that the I/O subsystem improves the efficiency of the computer is by sche I/O operations.

✓ Another way is by using storage space in main memory or on disk, via techniques c buffering, caching, and spooling.

Services include;

I/O Scheduling:

To determine a good order in which to execute the set of I/O requests. Uses:

a) It can improve overall system performance,

b) It can share device access fairly among processes, and

c) It can reduce the average waiting time for 1/0 to complete. Implementation: OS developers implement scheduling by maintaining a ―queue of requests‖ for each device

1. When an application issues a blocking I/O system call,
2. The request is placed on the queue for that device.
3. The I/O scheduler rearranges the order of the queue to improve the overall system efficiency and the average response time experienced by applications.

Buffering:

Buffer: A memory area that stores data while they are transferred between two devices or between a device and an application.

Reasons for buffering:

a) To cope with a speed mismatch between the producer and consumer of a data stream.

b) To adapt between devices that have different data-transfer sizes.

c) To support copy semantics for application I/O.

Copy semantics: Suppose that an application has a buffer of data that it wishes to write disk. It calls the write () system call, providing a pointer to the buffer and an integer spec the number of bytes to write.

After the system call returns, what happens if the application changes the contents of t buffer? With copy semantics, the version of the data written to disk is guaranteed to be version at the time of the application system call, independent of any subsequent chang the application's buffer. A simple way that the operating system can guarantee copy sem is for the write() system call to copy the application data into a kernel buffer before retu control to the application. The disk write is performed from the kernel buffer, so that subsequent changes to the application buffer have no effect.

5.3.3. Caching

A cache is a region of fast memory that holds copies of data. Access to the cached copy more efficient than access to the original Cache vs buffer: A buffer may hold the only existing copy of a data item, whereas a cache just holds a copy on faster storage of an it that resides elsewhere.

When the kernel receives a file I/O request,

1. The kernel first accesses the buffer cache to see whether that region of the file is alrea available in main memory.
2. If so, a physical disk I/O can be avoided or deferred. Also, disk writes are accumulate the buffer cache for several seconds, so that large transfers are gathered to allow efficie write schedules.

5.3.4. Spooling and Device Reservation:

Spool: A buffer that holds output for a device, such as a printer, that cannot accept interleaved data streams. A printer can serve only one job at a time, several application wish to print their output concurrently, without having their output mixed together

The os provides a control interface that enables users and system administrators ;

a) To display the queue,

b) To remove unwanted jobs before those jobs print,

c) To suspend printing while the printer is serviced, and so on. Device reservation - pro exclusive access to a device

✓ System calls for allocation and de-allocation

✓ Watch out for deadlock

Error Handling:

• An operating system that uses protected memory can guard against many kinds of hardware and application errors.

• OS can recover from disk read, device unavailable, transient write failures • Most r an error number or code when I/O request fails

• System error logs hold problem reports

5.7 Disk Structure

- Disk drives are addressed as large 1-dimensional arrays of *logical blocks*, where tl logical block is the smallest unit of transfer.
- The 1-dimensional array of logical blocks is mapped into the sectors of the disk sequentially.
- Sector 0 is the first sector of the first track on the outermost cylinder.
- Mapping proceeds in order through that track, then the rest of the tracks in that cyl and then through the rest of the cylinders from outermost to innermost.

5.8 Disk scheduling:

One of the responsibilities of the operating system is to use the hardware efficiently. Fc disk drives,

1. A fast access time and
2. High disk bandwidth.

• The **access time** has two major components;

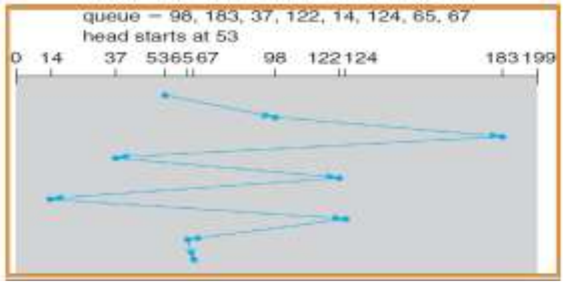
✓ The **seek time** is the time for the disk arm to move the heads to the cylinder contain the desired sector.

✓ The **rotational latency** is the additional time waiting for the disk to rotate the desi sector to the disk head.

• The disk **bandwidth** is the total number of bytes transferred, divided by the total tim between the first request for service and the completion of the last transfer. We can imp both the access time and the bandwidth by disk scheduling.

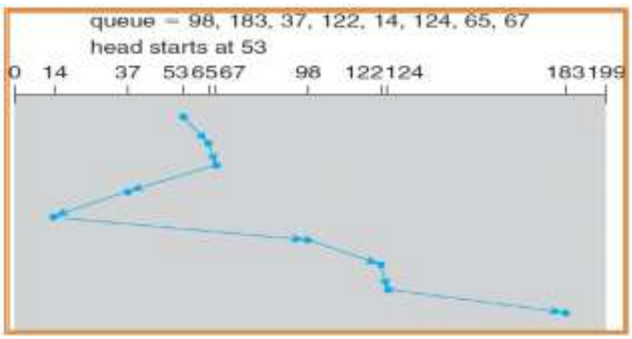
Disk scheduling: Servicing of disk I/O requests in a good order.

FCFS Scheduling: The simplest & fastest form of disk scheduling.



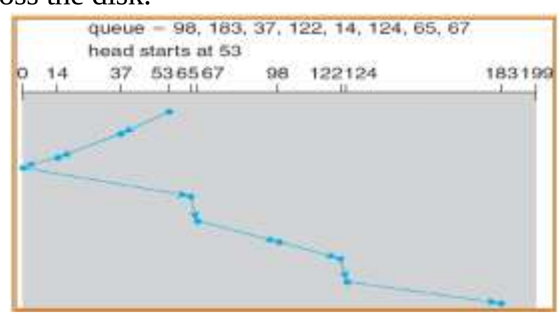
SSTF (shortest-seek-time-first)Scheduling

Service all the requests close to the current head position, before moving the head f away to service other requests. That is selects the request with the minimum seek time the current head position.



SCAN Scheduling

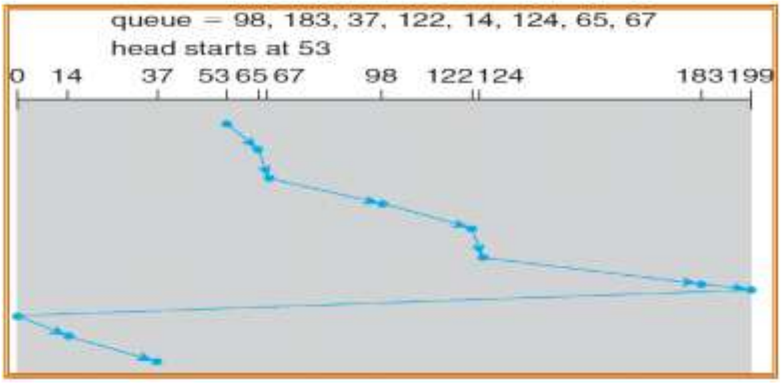
The disk head starts at one end of the disk, and moves toward the other end, servicing requests as it reaches each cylinder, until it gets to the other end of the disk. At the othe the direction of head movement is reversed, and servicing continues. The head continu scans back and forth across the disk.



Elevator algorithm: Sometimes the SCAN algorithm is called as the elevator algorith since the disk arm behaves just like an elevator in a building, first servicing all the requ going up, and then reversing to service requests the other way.

C-SCAN Scheduling

Variant of SCAN designed to provide a more uniform wait time. It moves the head one end of the disk to the other, servicing requests along the way. When the head reach other end, however, it immediately returns to the beginning of the disk, without servici any requests on the return trip.



5.9 Disk Management:

Disk Formatting:

Low-level formatting or physical formatting:

Before a disk can store data, the sector is divided into various partitions. This process is called low-level formatting or physical formatting. It fills the disk with a special data structure for each sector.

The data structure for a sector consists of

- ✓ Header,
- ✓ Data area (usually 512 bytes in size), and
- ✓ Trailer.

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

This formatting enables the manufacturer to

1. Test the disk and
2. To initialize the mapping from logical block numbers

To use a disk to hold files, the operating system still needs to record its own data struct on the disk. It does so in two steps.

(a) The first step is **Partition** the disk into one or more groups of cylinders. Among the partitions, one partition can hold a copy of the OS's executable code, while another hol user files.

(b) The second step is **logical formatting** .The operating system stores the initial file-sys data structures onto the disk. These data structures may include maps of free and alloca space and an initial empty directory.

Boot Block:

For a computer to start running-for instance, when it is powered up or rebooted-it to have an initial program to run. This initial program is called bootstrap program & it s be simple. It initializes all aspects of the system, from CPU registers to device controlle and the contents of main memory, and then starts the operating system.

To do its job, the bootstrap program

1. Finds the operating system kernel on disk,
2. Loads that kernel into memory, and
3. Jumps to an initial address to begin the operating-system execution. The bootstrap is in read-only memory **(ROM).**

Advantages:

1. ROM needs no initialization.
2. It is at a fixed location that the processor can start executing when powered up or r
3. It cannot be infected by a computer virus. Since, ROM is read only.

The full bootstrap program is stored in a partition called the **boot blocks**, at a fixed location on the disk. A disk that has a boot partition is called a **boot disk or system dis** The code in the boot ROM instructs the disk controller to read the boot blocks into mer and then starts executing that code.

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

Bad Blocks:

The disk with defected sector is called as bad block. Depending on the disk and controll use, these blocks are handled in a variety of ways; **Method 1: Handled manually**

If blocks go bad during normal operation, a **special program** must be run manu to search for the bad blocks and to lock them away as before. Data that resided on the b blocks usually are lost.

Method 2: "sector sparing or forwarding"

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

A typical bad-sector transaction might be as follows:

1. The operating system tries to read logical block 87.

2. The controller calculates the ECC and finds that the sector is bad.

3. It reports this finding to the operating system.

4. The next time that the system is rebooted, a special command is run to tell the cont to replace the bad sector with a spare.

5. After that, whenever the system requests logical block 87, the request is translated in replacement sector's address by the controller.

Method 3: "sector slipping"

For an example, suppose that logical block 17 becomes defective, and the first available

spare follows sector 202. Then, sector slipping would remap all the sectors from 17 to 2

moving them all down one spot. That is, sector 202 would be copied into the spare, the sector 201 into 202, and then 200 into 201, and so on, until sector 18 is copied into sect Slipping the sectors in this way frees up the space of sector 18, so sector 17 can be map to it.

5.10 Swap-Space Management

Swap-space — Virtual memory uses disk space as an extension of main memo The main goal for the design and implementation of swap space is —to provide the best throughput for the virtual-memory system

Swap-Space Use

Swap space is used in various ways by different operating systems, depending on th implemented memory-management algorithms.

1. The systems that implement swapping may use swap space to hold the entire process image, including the code and data segments.

2. Paging systems may simply store pages that have been pushed out of main memory. T amount of swap space needed on a system can therefore vary depending on

(a) The amount of physical memory,

(b) The amount of virtual memory it is backing and

(c) The way in which the virtual memory is used.

It can range from a few megabytes of disk space to gigabytes.

Some operating systems, such as UNIX, allow the use of multiple swap spaces.

Estimation of swap space: Note that it is safer to overestimate than to underestimate s space, because if a system runs out of swap space it may be forced to abort processes o crash entirely. Overestimation wastes disk space that could otherwise be used for files, does no other harm.

Swap-Space Location

A swap space can reside in two places:

1. Swap space can be carved out of the normal file system, or
2. It can be in a separate disk partition.

(I) Normal file system:

If the swap space is simply a large file within the file system, normal file-system routines be used to create it, name it, and allocate its space.

This approach, though easy to implement, is also inefficient.

(-) Finding the directory structure and the disk-allocation data structures takes time and disk accesses.

(-) External fragmentation can greatly increase swapping times by forcing multiple seel during reading or writing of a process image.

We can improve performance

(a) By **caching** the block location information in physical memory and

(b) By using **special tools** to allocate physically contiguous blocks for the swap file **(II**

Separate disk partition:

In this a separate swap-space storage manager is used to allocate and de-allocat blocks. This manager uses algorithms optimized for speed, rather than for storage effici

(-) Internal fragmentation may increase, but this tradeoff is acceptable.

(+) Data in the swap space generally live for much shorter amounts of time than do file the file system

(+) The swap area may be accessed much more frequently.

This approach creates a fixed amount of swap space during disk partitioning. A more swap space can be done only via

1. Repartitioning of the disk or
2. Adding another swap space elsewhere.

Swap-Space Management: An Example

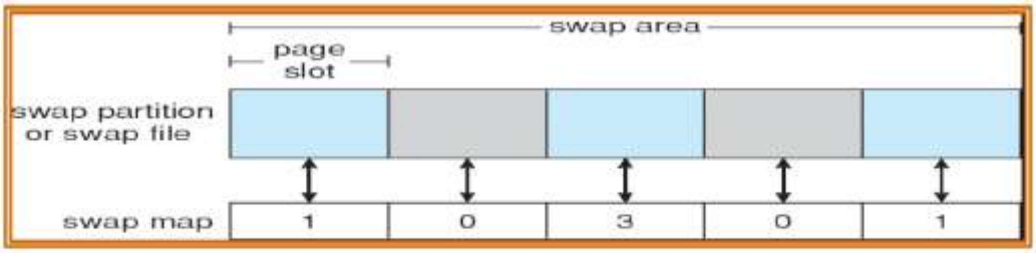
• BSD allocates swap space when process starts; holds text segment (the program) andc segment.

- Kernel uses swap maps to track swap-space use.
- Solaris 2 allocates swap space only when a page is forced out of physical memory, r

when the virtual

memory page is first created.

Data Structures for Swapping on Linux Systems



QUESTION BANK

UNIT – I
TWO MARKS

1. What is an Operating System?

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 & so on. 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?

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?

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, and 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 Graceful Degradation?

In multiprocessor systems, failure of one processor will not halt the system, but only slow it down. If there is ten processors & if any one fails then 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.

11. What 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 in order to indicate the current mode. Mode bit is '0' for monitor mode and '1' for user mode.

12. 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.

13. 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

14. 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

15. 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

16. 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.

17. What is a Process State and mention the various States of a Process?

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

18. What is Process Control Block (PCB)?

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

19. What is 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.

20. What is meant by Context Switch?

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.

21. What is Spooling?

Spooling means **S**imultaneous **P**eripheral **O**perations **O**n **L**ine. It is a high-speed device like a disk is interposed between a running program and a low –speed device involved with the program in input/output. It disassociates a running program from the slow operation of devices like printers.

22. What are System Calls?

System calls provide the interface between a process and the Operating system. System Calls are also called as Monitor call or Operating-system function call. When a system call is executed, it is treated as by the hardware as software interrupt. Control passes through the interrupt vector to a service routine in the operating system, and the mode bit is set to monitor mode.

23. List the services provided by an Operating System?

- Program execution

- I/O Operation
- File-System manipulation
- Communications
- Error detection

24. What are the two types of Real Time Systems?

- Hard real time system
- Soft real time system

25. What is the difference between Hard Real Time System and Soft Real Time System?

A hard real time system guarantees that critical tasks complete on time. In a soft real time system, a critical real-time task gets priority over the other tasks, and retains that priority until it completes. Soft real time systems have more limited utility than do hard real-time systems.

26. Write the difference between Multiprogramming and Non - Multiprogramming?

The operating system picks and begins to execute one of the jobs in the memory. Eventually, the job may have to wait for some task, such as a tape to be mounted, or an I/O operation to complete. In a non-multiprogrammed system, the CPU would sit idle. In a multiprogramming system, the operating system simply switches to and executes another job. When that job needs to wait, the CPU is switched to another job, and so on. Eventually, the first job finishes waiting and gets the CPU back. As long as there is always some job to execute, the CPU will never be idle.

27. What are the design goals of an Operating System?

The requirements can be divided into two basic groups: User goals and System goals. Users desire that the system should be convenient and easy to use, easy to learn, reliable, safe and fast. The Operating system should be easy to design, implement, and maintain. Also it should be flexible, reliable, error free and efficient. These are some of the requirements, which are vague and have no general solution.

28. What are the five major categories of System Calls?

- Process Control
- File-management
- Device-management
- Information maintenance
- Communications

29. What is the use of Fork and Execve System Calls?

Fork is a System calls by which a new process is created. Execve 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.

30. Define Elapsed CPU time and Maximum CPU time?

Elapsed CPU Time: Total CPU time used by a process to date.

Maximum CPU Time: Maximum amount of CPU time a process may use.

<u>14 MARKS</u>

1. What are the system components of an Operating System and explain them?

Common System Components,

- Process Management
- Main Memory Management

- File Management
- I/O System Management
- Secondary Management
- Networking
- Protection System
- Command-Interpreter System

2. Define System Calls. Write about the various System Calls.

Introduction

Types of System Calls

- Process control
- File management
- Device management
- Information maintenance
- Communications

3. What is a Process? Explain the Process Control Block and the various Process States.

Introduction

- An operating system executes a variety of programs:
- Batch system– jobs
- Time-shared systems – user programs or tasks
- Textbook uses the terms job and process almost interchangeably.
- Process– a program in execution; process execution must progress in sequential fashion.
- A process includes:
- Program counter
- Stack
- Data section
- Process 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.

4. Explain Process Creation and Process Termination

Process Creation

Parent process creates 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 OS to decide it (exit).

- Output the 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.

5. Explain about Inter Process Communication.

- Definition
- Message Passing System
- Naming
- Direct Communication
- Indirect Communication
- Synchronization
- Buffering

UNIT – II
TWO MARKS

1. What is a Thread?

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?

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

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

3. Compare User Threads and Kernel Threads.

4. 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.

User threads Kernel threads

User threads are supported above the kernel and are implemented by a thread library at the user level Kernel threads are supported directly by the operating system Thread creation & scheduling are done in the user space, without kernel intervention. Therefore they are fast to create and manage Thread creation, scheduling and management are done by the operating system. Therefore they are slower to create & manage compared to user threads Blocking system call will

cause the entire process to block
If the thread performs a blocking system call, the
kernel can schedule another thread in the
application for execution

5. 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.

6. Define CPU Scheduling.

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.

7. What is Preemptive and Non - Preemptive scheduling?

Under non - preemptive 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.

8. What is a Dispatcher?

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 into the user program to restart that program.

9. What is Dispatch Latency?

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

10. What are the various scheduling criteria for CPU Scheduling?

The various scheduling criteria are,

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

11. Define Throughput?

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.

12. 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.

13. Define Race Condition.

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.

14. What is Critical Section problem?

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 executing in its critical section.

15. What are the requirements that a solution to the Critical Section Problem must satisfy?

The three requirements are,

- Mutual exclusion
- Progress
- Bounded waiting

16. 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.

17. Give two hardware instructions and their definitions which can be used for implementing Mutual Exclusion.

Test And Set

```
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;
}
```

18. What is a Semaphore?

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++;
}
```

19. 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.

20. How can we say the First Come First Served scheduling algorithm is Non Preemptive?

Once the CPU has been allocated to the process, that process keeps the CPU until it releases, either by terminating or by requesting I/O. So we can say the First Come First Served scheduling algorithm is non preemptive.

21. What is Waiting Time in CPU scheduling?

Waiting time is the sum of periods spent waiting in the ready queue. CPU scheduling algorithm affects only the amount of time that a process spends waiting in the ready queue.

22. What is Response Time in CPU scheduling?

Response time is the measure of the time from the submission of a request until the first response is produced. Response time is amount of time it takes to start responding, but not the time that it takes to output that response.

23. Differentiate Long Term Scheduler and Short Term Scheduler

The long-term scheduler or job scheduler selects processes from the job pool and loads them into memory for execution. The short-term scheduler or CPU scheduler selects from among the process that are ready to execute, and allocates the CPU to one of them.

24. Write some classical problems of Synchronization?

- The Bounded-Buffer Problem
- The Readers-Writers Problem
- The Dining Philosophers Problem

25. When the error will occur when we use the Semaphore?

- When the process interchanges the order in which the wait and signal operations on the semaphore mutex.
- When a process replaces a signal (mutex) with wait (mutex).
- When a process omits the wait (mutex), or the signal (mutex), or both.

26. What is Mutual Exclusion?

A way of making sure that if one process is using a shared modifiable data, the other processes will be excluded from doing the same thing. Each process executing the shared data variables excludes all others from doing so simultaneously. This is called mutual exclusion.

27. Define the term Critical Regions?

Critical regions are small and infrequent so that system through put is largely unaffected by their existence. Critical region is a control structure for implementing mutual exclusion over a shared variable.

28. What are the drawbacks of Monitors?

- Monitor concept is its lack of implementation most commonly used

programming languages.
 There is the possibility of deadlocks in the case of nested monitor's calls.
29. What are the two levels in Threads?
Thread is implemented in two ways.
 User leveland Kernel level
30. What is a Gantt Chart?
A two dimensional chart that plots the activity of a unit on the Y-axis and the time on the X-axis. The chart quickly represents how the activities of the units are serialized.
31. Define Deadlock.
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.
32. 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.
33. What are conditions under which a deadlock situation may arise?
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
34. 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.
35. Define Request Edge and Assignment Edge.
A directed edge from process Pi to resource type Rj is denoted by PiàRj; 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 RjàPi, it signifies that an instance of resource type has been allocated to a process Pi. A directed edge PiàRj is called a request edge. A directed edge RjàPi is called an assignment edge.
36. 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.

37. Define Deadlock Prevention.

Deadlock prevention is a set of methods for ensure that at least any one of the four necessary conditions like mutual exclusion, hold and wait, no pre-emption 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.

38. 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.

39. 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 avoid 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.

40. 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-allocation is 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.

41. 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.

42. What are 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.

43. 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).

44. What are the methods for dealing the Deadlock Problem?

- Use a protocol to ensure that the system will never enter a deadlock state.
- Allow the system to enter the deadlock state and then recover.
- Ignore the problem all together, and pretend that deadlocks never occur in the system.

45. Differentiate Deadlock and Starvation.

A set of processes is in deadlock state when every process in the set is waiting for an event that can be caused only by the other process in the set. Starvation or indefinite blocking is a situation where processes wait indefinitely within the semaphore.

14 MARKS

1. Write about the various CPU Scheduling Algorithms.

- Optimization Criteria
- First-Come, First-Served (FCFS) Scheduling
- Shortest-Job-First (SJF) Scheduling
- Priority Scheduling
- Round Robin (RR)
- Multilevel Queue
- Multilevel Feedback Queue

2. Explain the classical problem on Synchronization.

Classical Problems are,

- Bounded-Buffer Problem
- Readers and Writers Problem
- Dining-Philosophers Problem

3. Explain about Monitors.

Introduction

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

monitor monitor-name
{s
hared variable declarations
procedure body P1 (…) { . . . }
procedure body P2 (…) { . . . }
procedure body Pn (…) { . . . }
{i
nitialization code
}}

4. 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;
…i
f (next-count > 0)
signal(next)
else
signal(mutex);

- Mutual exclusion within a monitor is ensured.
- 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 implementedas:

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

5. Give a detailed description about Deadlocks and its Characterization

- Deadlock Characterization
- Necessary Conditions
- Mutual exclusion: only one process at a time can use a resource.
- Hold and wait: a process 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 {P0, P1, …, P0} of waiting processes such that P0 is waiting
for a resource that is held by P1, P1 is waiting for a resource that is held by P2, …, Pn–1 is waiting for a resource that is held by Pn, and P0 is waiting for a resource that is held by P0.

6. Explain about the methods used to Prevent Deadlocks

- Deadlock Prevention
- 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.
- No Preemption
- Circular Wait – impose a total ordering of all resource types, and require that each process
  requests resources in an increasing order of enumeration.

7. Write in detail about Deadlock Avoidance.

- Multiple instances.
- Each process must a priori claim maximum use.
- When a process requests a resource it may have to wait.
- When a process gets all its resources it must return them in a finite amount of time.
- Data Structures for the Banker’s Algorithm, Safety Algorithm
- Resource-Request Algorithm for Process Pi
- Example of Banker’s Algorithm

UNIT –III
TWO MARKS

1. 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 relocatable 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 relocatable linking loader is called to load the desired program into memory.

2. 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.

3. 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.

4. Define Swapping.

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

5. What do you mean by Best Fit?

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.

6. What do you mean by First Fit?

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.

7. How is memory protected in a paged environment?

Protection bits that are associated with each frame accomplish memory protection in a paged environment. The protection bits can be checked to verify that no writes are being made to a read-only page.

8. What is External Fragmentation?

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

9. What is Internal Fragmentation?

When the allocated memory may be slightly larger than the requested memory, the difference between these two numbers is internal fragmentation.

10. What do you mean by Compaction?

Compaction is a solution to external fragmentation. The memory contents are shuffled to place all free memory together in one large block. It is possible only if relocation is dynamic, and is done at execution time.

11. What are Pages and Frames?

Paging is a memory management scheme that permits the physical-address space of a process to be non-contiguous. In the case of paging, physical memory is broken into fixed-sized blocks called frames and logical memory is broken into blocks of the same size called pages.

12. What is the use of Valid-Invalid Bits in Paging?

When the bit is set to valid, this value indicates that the associated page is in the process's logical address space, and is thus a legal page. If the bit is said to invalid, this value indicates that the page is not in the process's logical address space. Using the valid-invalid bit traps illegal addresses.

13. What is the basic method of Segmentation?

Segmentation is a memory management scheme that supports the user view of memory. A logical address space is a collection of segments. The logical address consists of segment number and offset. If the offset is legal, it is added to the segment base to produce the address in physical memory of the desired byte.

14. A Program containing relocatable code was created, assuming it would be loaded at address 0. In its code, the program refers to the following addresses: 50,78,150,152,154. If the program is loaded into memory starting at location 250, how do those addresses have to be adjusted?

All addresses need to be adjusted upward by 250.So the adjusted addresses would be 300, 328, 400, 402, and 404.

15. What is Virtual Memory?

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.

16. 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.

17. 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.

18. 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.

19. 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

20. 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.

21. 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.

22. 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.

23. What are the major problems to implement Demand Paging?

The two major problems to implement demand paging is developing,

- Frame allocation algorithm
- Page replacement algorithm

24. 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.

14 MARKS

1. Explain Dynamic Storage-Allocation Problem

- ***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 leftover hole.
- ***Worst-fit:*** Allocate the largest hole; must also search entire list. Produces the largest leftover hole. First-fit and best-fit is better than worst-fit in terms of the speed and storage utilization.

2. Explain about Fragmentation

Fragmentation

- External Fragmentation– total memory space exists to satisfy a request, but it is not
 contiguous.
- Internal Fragmentation– allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition, but not being used.
- Reduce external fragmentation by compaction
- Shuffle memory contents to place all free memory together in one large block.
- Compaction is possible only if relocation is dynamic, and is done at execution time.
- I/O problem

3. Explain the concept of Paging

Basic method

- Logical address space of a process can be noncontiguous; process is allocated physical memory whenever the latter is available.

- Divide physical memory into fixed-sized blocks called frames (size is power of 2, between 512 bytes and 8192 bytes).
- Divide logical memory into blocks of same size called pages.
- Keep track of all free frames.
- To run a program of size n pages, need to find n free frames and load program.
- Set up a page table to translate logical to physical addresses.
- Internal fragmentation.

Address Translation Scheme

Address generated by CPU is divided into:

- Page number (p) – used as an index into a page table which contains base address of each page in physical memory.
- Page offset (d) – combined with base address to define the physical memory address that is sent to the memory unit.

4. Explain the types of Page Table Structure

- Hierarchical Paging
- Hashed Page Tables
- Inverted Page Tables

5. Explain about Segmentation in detail.

Basic method

- Memory-management scheme that supports user view of memory Segmentation

Architecture

- Logical address
- Segment table
- Base
- Limit
- Segment-table base register (STBR)
- Segment-table length register (STLR)
- Relocation.

Sharing

- Shared segments
- Same segment number

Allocation

- First fit/best fit
- external fragmentation

Protection With each entry in segment table associate:

- Validation bit = 0 illegal segment
- Read/write/execute privileges

Protection bits associated with segments; code sharing occurs at segment level. Since segments vary in length, memory allocation is a dynamic storage-allocation problem. A segmentation example is shown in the following diagram.

UNIT – IV
TWO MARKS

1. 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.

2. List the various File Attributes.

A file has certain other attributes, which vary from one operating system to another, but typically consist of these: Name, identifier, type, location, size, protection, time, date and user identification.

3. What are the various File Operations?

The basic file operations are,

- Creating a file
- Writing a file
- Reading a file
- Repositioning within a file
- Deleting a file
- Truncating a file

4. What is 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

5. What are the different Accessing Methods of a File?

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.

6. 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.

7. 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

8. 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

9. Define UFD and MFD.

In the two-level directory structure, each user has 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.

10. 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.

11. What is Access Control List (ACL)?

The most general scheme to implement identity-dependent access is to associate with each file and directory an access control unit.

12. Define Equal Allocation.

The way to split '*m'* frames among '*n'* processes is to give everyone an equal share, *m/n* frames. For instance, if there are 93 frames and 5 processes, each process will get 18 frames. The leftover 3 frames could be used as a free-frame buffer pool. This scheme is called equal allocation.

13. What is the cause of Thrashing? How does the system detect thrashing? Once it
detects thrashing, what can the system do to eliminate this problem?

Thrashing is caused by under allocation of the minimum number of pages required by a process, forcing it to continuously page fault. The system can detect thrashing by evaluating the level of CPU utilization as compared to the level of multiprogramming. It can be eliminated by reducing the level of multiprogramming.

14. If the average page faults service time of 25 ms and a memory access time of 100ns.Calculate the effective access time.

Effective access time = (1-p)*ma + p*page fault time = (1-p)*100+p*25000000
= 100-100p+25000000*p = 100 + 24999900p

15. What is Belady's Anomaly?

For some page replacement algorithms, the page fault rate may increase as the number of allocated frames increases.

16. What are the different types of Access?

Different types of operations may be controlled in access type. These are,

 Read
 Write
 Execute
 Append
 Delete
 List

17. What are the types of Path Names?

Path names can be of two types.

 Absolute path name: Begins at the root and follows a path down to the specified file, giving the directory names on the path.
 Relative path name: Defines a path from the current directory.

18. What is meant by Locality of Reference?

The locality model states that, as a process executes, it moves from locality to

locality. Locality is of two types.
- Spatial locality
- Temporal locality.

19. What are the various layers of a File System?

The file system is composed of many different levels. Each level in the design uses the feature of the lower levels to create new features for use by higher levels.
- Application programs
- Logical file system
- File-organization module
- Basic file system
- I/O control
- Devices

20. What are the Structures used in File-System Implementation?

Several on-disk and in-memory structures are used to implement a file system
- On-disk structure include
- Boot control block
- Partition block
- Directory structure used to organize the files in File control block (FCB)
- In-memory structure include
- In-memory partition table
- In-memory directory structure
- System-wide open file table
- Per-process open table

21. What are the Functions of Virtual File System (VFS)?

It has two functions,
- It separates file-system-generic operations from their implementation defining a clean VFS interface. It allows transparent access to different types of file systems mounted locally.
- VFS is based on a file representation structure, called a vnode. It contains a numerical value for a network-wide unique file .The kernel maintains one vnode structure for each active file or directory.

22. Define Seek Time and Latency Time.

The time taken by the head to move to the appropriate cylinder or track is called seek time. Once the head is at right track, it must wait until the desired block rotates under the read- write head. This delay is latency time.

23. What are the Allocation Methods of a Disk Space?

Three major methods of allocating disk space which are widely in use are
- Contiguous allocation
- Linked allocation
- Indexed allocation

24. What are the advantages of Contiguous Allocation?

The advantages are,
- Supports direct access
- Supports sequential access

- Number of disk seeks is minimal.

25. What are the drawbacks of Contiguous Allocation of Disk Space?

The disadvantages are,

- Suffers from external fragmentation
- Suffers from internal fragmentation
- Difficulty in finding space for a new file
- File cannot be extended
- Size of the file is to be declared in advance

26. What are the advantages of Linked Allocation?

The advantages are,

- No external fragmentation
- Size of the file does not need to be declared

27. What are the disadvantages of Linked Allocation?

The disadvantages are,

- Used only for sequential accessof files.
- Direct access is not supported
- Memory space required for the pointers.
- Reliability is compromised if the pointers are lost or damaged

28. What are the advantages of Indexed Allocation?

The advantages are,

- No external-fragmentation problem
- Solves the size-declaration problems
- Supports direct access

29. How can the index blocks be implemented in the Indexed Allocation Scheme?

The index block can be implemented as follows,

- Linked scheme
- Multilevel scheme
- Combined scheme

30. Define Rotational Latency and Disk Bandwidth.

Rotational latency is the additional time waiting for the disk to rotate the desired sector to the disk head. The disk bandwidth is the total number of bytes transferred, divided by the time between the first request for service and the completion of the last transfer.

31. How free-space is managed using Bit Vector Implementation?

The free-space list is implemented as a bit map or bit vector. Each block is represented by 1 bit. If the block is free, the bit is 1; if the block is allocated, the bit is 0.

32. Define Buffering.

A buffer is a memory area that stores data while they are transferred between two devices or between a device and an application. Buffering is done for three reasons,

- To cope with a speed mismatch between the producer and consumer of a data stream
- To adapt between devices that have different data-transfer sizes
- To support copy semantics for application I/O

14 MARKS

1. Explain the File System Structure in detail

- None- sequence of words, bytes
- Simple record structure
- Lines
- Fixed length
- Variable length
- Complex Structures
- Formatted document
- Relocatable load file
- Can simulate the last two with the first method by inserting appropriate control characters.
- Who decides?
- Operating system
- Program

2. Discuss the File System Organization and File System Mounting.

A file system must be mounted before it can be accessed. An unmounted file system is mounted at a mount point.

- Existing
- Unmounted Partition
- Mount Point

3. Explain about File Sharing.

- Introduction
- File Sharing– Remote File Systems
- File Sharing– Failure Modes
- File Sharing– Consistency Semantics

4. Explain about the File System Implementation.

- File System Structure
- File System Implementation
- Directory Implementation
- Allocation Methods
- Free-Space Management
- Efficiency and Performance
- Recovery and Log-Structured File Systems
- NFS

5. Explain about various Allocation Methods.

An allocation method refers to how disk blocks are allocated for files:

- Contiguous allocation
- Linked allocation
- Indexed allocation

UNIT – V
TWO MARKS

1. Define 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. Caching and buffering are distinct functions, but sometimes a region of memory can be used for both purposes.

2. Define Spooling.

A spool is a buffer that holds output for a device, such as printer, that cannot accept interleaved data streams. 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.

3. What are the various Disk-Scheduling Algorithms?

The various disk-scheduling algorithms are,

- First Come First Served Scheduling
- Shortest Seek Time First Scheduling
- SCAN Scheduling
- C-SCAN Scheduling
- LOOK scheduling

4. What is Low-Level Formatting?

Before a disk can store data, it must be divided into sectors that the disk controller can read and write. This process is called low-level formatting or physical formatting. Low-level formatting fills the disk with a special data structure for each sector. The data structure for a sector consists of a header, a data area, and a trailer.

5. What is the use of Boot Block?

For a computer to start running when powered up or rebooted it needs to have an initial program to run. This bootstrap program tends to be simple. It finds the operating system on the disk loads that kernel into memory and jumps to an initial address to begin the operating system execution. The full bootstrap program is stored in a partition called the boot blocks, at fixed location on the disk. A disk that has boot partition is called boot disk or system disk.

6. What is Sector Sparing?

Low-level formatting also sets aside spare sectors not visible to the operating system. The controller can be told to replace each bad sector logically with one of the spare sectors. This scheme is known as sector sparing or forwarding.

7. What are the techniques used for performing I/O.

- Programmed I/O
- Interrupt driven I/O
- Direct Memory Access (DMA).

8. Give an example of an application in which data in a file should be accessed in the following order:

***Sequentially* -** Print the content of the file.

Randomly - Print the content of record *i*. This record can be found using hashing or index techniques

9. What problems could occur if a system allowed a file system to be mounted simultaneously at more than one location?

There would be multiple paths to the same file, which could confuse users or encourage mistakes. (Deleting a file with one path deletes the file in all the other paths.)

10. Why must the bit map for file allocation be kept on mass storage rather than in main memory?

In case of system crash (memory failure), the free-space list would not be lost as it would be if the bit map had been stored in main memory.

11. What criteria should be used in deciding which strategy is best utilized for a particular file?

- **Contiguous -** File is usually accessed sequentially, if file is relatively small.
- **Linked** - File is usually accessed sequentially, if the file is large.
- **Indexed** - File is usually accessed randomly, if file is large.

12. What is meant by RAID?

"RAID" is now used as an umbrella term for computer data storage schemes that can divide and replicate data among multiple hard disk drives. The different schemes architectures are named by the word RAID followed by a number, as in RAID 0, RAID 1, etc. RAID's various designs involve two key design goals: increase data reliability and/or increase output performance. When multiple physical disks are set up to use RAID technology, they are said to be *in* a *RAID* array.

13. What is meant by Stable Storage?

Stable storage is a classification of computer data storage technology that guarantees atomicity for any given write operation and allows software to be written that is robust against some hardware and power failures. To be considered atomic, upon reading back a just written-to portion of the disk, the storage subsystem must return either the write data or the data that was on that portion of the disk before the write operation.

14. What is meant by Tertiary Storage?

Tertiary storage or **tertiary memory** provides a third level of storage. Typically it involves a robotic mechanism which will *mount* (insert) and *dismount* removable mass storage media into a storage device according to the system's demands; this data is often copied to secondary storage before use.

15. Write a note on Descriptor?

UNIX processes use *descriptors* to reference I/O streams. Descriptors are small unsigned integers obtained from the *open* and *socket* system calls.. A *read* or *write* system call can be applied to a descriptor to transfer data. The *close* system call can be used to deallocate any descriptor. Descriptors represent underlying objects supported by the kernel, and are created by system calls specific to the type of object. In 4.4BSD, three kinds of objects can be represented by descriptors: files, pipes, and sockets.

16. Write short notes on Pipes?

A *pipe* is a linear array of bytes, as is a file, but it is used solely as an I/O stream, and it is unidirectional. It also has no name, and thus cannot be opened with *open*. Instead, it is created by the *pipe* system call, which returns two descriptors, one of which accepts input that is sent to the other descriptor reliably, without duplication, and in order. The system also supports a named pipe or FIFO. A FIFO has properties identical to a pipe, except that it appears in the file system; thus, it can be opened using the *open* system call. Two processes that wish to communicate each open the FIFO: One opens it for reading, the other for writing.

14 MARKS

1. Explain the allocation methods for disk space?

2. What are the various methods for free space management?

3. Write about the kernel I/O subsystem.

4. Explain the various disk scheduling techniques

- FCFS

- SSTF
- SCAN
- C-SCAN
- C-LOOK

5. Write notes about disk management and swap-space management.
6. Explain in detail the allocation and freeing the file storage space.
7. Explain the backup and recovery of files.
8. Discuss with diagrams the following three disk scheduling: FCFS, SSTF, CSCAN.
9. Compare and contrast the FREE SPACE and SWAP SPACE management.
10. Explain the disk scheduling algorithms
11. Describe the most common schemes for defining the logical structure of a Directory.
12. Explain the life cycle of an I/O request with flowchart.
13. Discuss about the UNIX file system in detail.
14. Discuss briefly about Memory Management in UNIX.
15. Explain the process management under LINUX OS.
16. In what ways the directory is implemented?
17. Explain linked allocation in detail.
18. Write the indexed allocation with its performance.
19. Explain the I/O hardware.
20. Explain in detail about Raid

- RAID 1
- RAID 2
- RAID 3
- RAID 4
- RAID 5

ONLINE QUESTIONS

UNIT-I

Questions	opt1	opt2	opt3	opt4	opt5	opt6	answer
Operating system is referred as	Control program	Resource allocator	Resource manager	All of these			All of these
Systems have more than one processor in close communication are called	Tightly coupled system	Loosely coupled systems	Co-operative system	All of these			Tightly coupled system
The system, which takes task's priority over other tasks is	Soft real system	Co operating system	Multiprocessor System	Hard real time			Soft real system
The system which provide a file-system interface where clients can create, update, read, and delete files.	Compute-Server system	File server system	Client server system	All of these			File server system
The system which has a small amount of memory include slow processors and feature small display screens is referred as	Mainframe	Desktop	Multiprocessor	Hand held			Hand held
Which of the following is a not Symmetric Multiprocessing system	Windows NT	OS/2	UNIX	Sun OS version 4			Sun OS version 4
Privileged instructions can be executed by	User	kernel	Both kernel & user	None of the above			kernel
How is the protection for memory provided	Using Physical & Logical address	Using index register	Using base & limit register	Using program counter			Using base & limit register
Mechanism used for processor allocation is called	Disk scheduling	CPU scheduling	Job scheduling	None of these			CPU scheduling
PCB holds the information about	I/O	Memory	Process	All of these			All of these
The process which spend more time in processor is	Bound process	CPU Bound process	I/O bound process	None of these			CPU Bound process

called							
A process does not affect or affected by the other processes executing in the system is called	Sharing system	cooperating system	Independent process	None of these			Independent process
Fork return 0 to create	Parent process	Child process	Separate new process	All of these			Child process
The scheduler selects processes from the job pool and loads them into memory for execution is called	Short-term scheduler	Long-term scheduler	Medium term scheduler	All of these			Long-term scheduler
Which of the following is NOT an operation on process	Copy the process	Change a process priority	Block a process	Wake up a process			Copy the process
The state that the process is waiting to be assigned to a processor is called as	New	Running	Waiting	Ready			Ready
Sender never blocks in --------------- Buffering method	Zero capacity	Bounded Capacity	Unbounded capacity	All of these			Unbounded capacity
The module that gives control of the CPU to the process selected by the short-term scheduler.	Long-term scheduler	Medium term scheduler	Dispatcher	All of these			Dispatcher
The user who read information from buffer is called as	Producer	Consumer	Reader	None of these			Consumer
Execlp system call is used to_________	Replace the process memory space	Execute the command	Invoke the specified file	All of these			Replace the process memory space
Which system is a collection of loosely coupled processors interconnected	Clustered system	Distributed system	Mainframe system	Real time system			Distributed system

by a communication network?							
A fault-tolerant system should continue to function, perhaps in a degrade form, when faced with failures such as ____________	Communication faults	Storage-device crashes	Machine failures	All of these			All of these
The capability of a system to adapt to increased service load is its _____________	Scalability	Reliability	Flexibility	Atomicity			Scalability
A __________ consists of a set of machines under a dedicated cluster server.	Cross cluster	Networking	Cluster	None of these			Cluster
A _________ is a software entity running on one or more machines and providing a particular type of function to a priori unknown clients.	Server	Interface	Client	Service			Client
A ___________ DFS facilitates user mobility by bringing the user's environment to wherever a user logs in.	Trparent	Conventional	Dependent	Independent			Trparent
Which problem is the major drawback of caching?	Cache update	Cache-consistency	Buffer cache	Page cache			Cache-consistency
In ___________, the name of a file does not reveal any hint of the file's physical storage location.	Location independence	Location trparency	Location dependence	None of these			Location trparency
Which is the smallest set of	Physical unit	Logical unit	FCB	Component unit			Component unit

files that can be stored in a single machine, independently from other units?							
A _______ is a file service system whose clients, servers, and storage devices are dispersed among the sites of a distributed system.	AFS	NFS	DFS	RPC			DFS
In event ordering, if two events A and B, are not related by the ? relation, then they will be executed _______.	Sequentially	Independently	Monotonically	Concurrently			Concurrently
In fully distributed approach of mutual exclusion, a number of messages per critical section entry is ________.	2 * (n-1)	4 * (n-1)	2 * (n-2)	4 * (n-2)			2 * (n-1)
A _______ is a special type of message that is passed around the system.	File	Request	Token	Release			Token
Which one of the following is an advantage of Single-Coordinator approach?	Bottleneck	Simple implementation	Vulnerability	None of these			Simple implementation
The __________ includes a multitude of components, some written from scratch, others borrowed from other development	Linux kernel	Linux distribution	Linux system	Linux licensing			Linux system

projects.							
The _________ standard document is maintained by the Linux community as a me of keeping compatibility across the various system components.	Slackware	File trfer protocol	Public domain	File system hierarch y			File system hierarchy
One of the difficulty faced with deadlock prevention scheme is the possibility of __________	Semaphor e	Starvation	Mutual exclusion	All of these			Starvation
Which of the following is/are the components of Linux system?	Kernel	System libraries	System utilities	All of these			All of these
The algorithms that determine where a new copy of the coordinator should be restarted are called__________	Election algorithms	Stack algorithm	Local replacem ent algorithm s	Elevator algorith m			Election algorithm s
________ represent separate, concurrent execution contexts within a single process running a single program.	Fork	Kernel	Threads	Exec			Threads
____________ manages the execution of user programs to prevent errors and improper use of the computer	control program	CPU	process	thread			control program
The user view of the computer varies by the _________obei ng used	system	interface	terminal	None of the above			interface
The primary	efficient	increased	convenie	None of			convenie

goal of operating system is __________for the user		throughput	nce	the above			nce
To speed up the processing, operators __________together the jobs with similar needs	delete	time share	transfer	batched			batched
______ were the first computers used to tackle many commercial & scientific applications.	Mainframe systems	Desktop systems	Real time systems	Distributed system			Mainframe systems
__________ increases CPU utilization by organizing jobs so that the CPU always has one to execute.	Mainframe systems	Desktop systems	Real time systems	Multi programmed systems			Multi programmed systems
A process is a program in _________	Compilation	execution	Memory	Stack			execution
_________ is one of the advantage of multiprocessor systems	Self replicating	Decreased overhead	Increased reliability	Worm protection			Increased reliability
A _____ network exists within a room a, floor or a building	WAN	MAN	LAN	SAN			LAN
Compute receiver systems provide an ______to which clients can send requests to perform an action.	cable	interface	server	client			interface
Distributed systems is also	Tightly coupled	Loosely coupled	None of the	Both a & b			Loosely coupled

known as _____________.	systems	systems	above				systems
____________ gather together multiple CPUs to accomplish computational work.	Distributed systems	real time systems	clustered systems	None of the above			clustered systems
______________ _ systems has well defined ,fixed time constraints	Distributed systems	real time systems	clustered systems	None of the above			real time systems
In __________ real time system ,a critical real time tasks gets priority over other tasks.	Hard real time systems	Soft real time systems	Both	None of the above			Soft real time systems
________ denotes the current activity of a process	state	stack	program	registers			state
PCB is expanded as __________	program control block	process control block	producer consumer block	None of the above			process control block
When a process enters a system ,it is put into _____queue	ready queue	job queue	device queue	None of the above			job queue
When a process is ready and waiting to execute is kept in _________ queue.	ready queue	job queue	device queue	None of the above			ready queue
Each device has its own ______queue.	ready queue	job queue	device queue	None of the above			device queue
_______scheduler selects from among the processes that are ready to execute & allocate CPU to them.	Long time scheduler	Short time scheduler	None of the above	Both a & b			Short time scheduler

UNIT-II

Questions	opt1	opt2	opt3	opt4	opt 5	opt 6	answer
The process is also known as	Program section	Code section	Text section	None of these			Text section
The temporary data in a process is stored in the	List	Stack	Queue	Memory			Stack
The global variables of a process is stored in the	Program section	Text section	Data section	None of these			Data section
The program is also known as	Active entity	Process entity	Code entity	Passive entity			Passive entity
The process is also known as	Active entity	Process entity	Code entity	Passive entity			Active entity
The process shifts from the running to ready state when	Exit	I/O event occurs	Interrupt occurs	None of these			Interrupt occurs
The process shifts from waiting state to running state after the	I/O event completion	Exit	I/O event occurs	Interrupt occurs			I/O event completion
The process control block is also known as	Code control block	Task control block	Program control block	None of these			Task control block
The ready queue is implemented as	Queue	Stack	List	Graph			List
Which queue has its header pointing to first and the final PCB?	Job queue	I/O queue	Ready queue	None of these			Ready queue
Which of the	Device	Ready	Job	Device			Ready

following queues are found in Queuing diagram	queue	queue and device queue	queue and ready queue	and job queue			queue and device queue
The selection of a process is carried out by the	Enqueuer	Dequeuer	Selector	Scheduler			Scheduler
The long term scheduler is also known as	CPU scheduler	Job scheduler	Short term scheduler	Medium scheduler			Job scheduler
The short term scheduler is also known as	CPU scheduler	Job scheduler	Short term scheduler	Medium scheduler			CPU scheduler
The long term scheduler controls the degree of	Consistency	Processing	Multiprogramming	None of these			Multiprogramming
Which scheduler should select the good mix of I/O and CPU bound process	CPU scheduler	Medium term scheduler	Short term scheduler	Job scheduler			Job scheduler
The time sharing scheduler has	CPU scheduler	Medium term scheduler	Short term scheduler	Job scheduler			Medium term scheduler
The phenomenon of stopping the process temporarily and reintroducing it into the memory and executing it from where it left off	Shifting	Controlling	Swapping	None of these			Swapping
Switching the CPU to another process requires saving the state of the	Swapping	Shifting	Context switching	Switching			Context switching

old process and loading the saved state for the new process is called as							
Which of the following are not found in the PCB?	Context	Process counter	Register	None of these			None of these
Which system call is used to create child process?	Create process system call	Fork system call	Execlp system call	Wait system call			Fork system call
The process identifier returned by the fork system call for the new child process is	Non zero value	Zero is returned	Void	None of these			Zero is returned
The process identifier returned by the fork system call for the parent process is	Non zero value	Zero is returned	Void	None of these			Non zero value
Which system call is used to replace the process memory space with a new program?	Signal	Wait	Fork	Execlp			Execlp
The phenomenon of terminating the child process when the parent process terminates is called	Parallel termination	Process termination	Controlled termination	Cascaded termination			Cascaded termination
Which module gives	dispatcher	interrupt	scheduler	none of the mentio			dispatcher

control of the CPU to the process selected by the short-term scheduler ?				ned			
The interval from the time of submission of a process to the time of completion is termed as	waiting time	turnaround time	response time	throughput			turnaround time
Which scheduling algorithm allocates the CPU first to the process that requests the CPU first?	first-come, first-served scheduling	shortest job scheduling	priority scheduling	none of the mentioned			first-come, first-served scheduling
In priority scheduling algorithm	CPU is allocated to the process with highest priority	CPU is allocated to the process with lowest priority	equal priority processes can not be scheduled	none of the mentioned			CPU is allocated to the process with highest priority
In priority scheduling algorithm, when a process arrives at the ready	all process	currently running process	parent process	init process			currently running process

queue, its priority is compared with the priority of							
Time quantum is defined in	shortest job scheduling algorithm	round robin scheduling algorithm	priority scheduling algorithm	multilevel queue scheduling algorithm			round robin scheduling algorithm
Process are classified into different groups in	shortest job scheduling algorithm	round robin scheduling algorithm	priority scheduling algorithm	multilevel queue scheduling algorithm			multilevel queue scheduling algorithm
In multilevel feedback scheduling algorithm	a process can move to a different classified ready queue	classification of ready queue is permanent	processes are not classified into groups	none of the mentioned			a process can move to a different classified ready queue
Which one of the following can not be scheduled by the kernel?	kernel level thread	user level thread	process	none of the mentioned			user level thread
CPU scheduling is the basis of _____________	multiprocessor systems	multiprogramming operating systems	larger memory sized systems	None of these			multiprogramming operating systems
With multiprogramming, ______ is used productive	time	space	money	All of these			time

ly.							
The two steps of a process execution are : (choose two)	I/O Burst	CPU Burst	Memory Burst	OS Burst			a and b
An I/O bound program will typically have :	a few very short CPU bursts	many very short I/O bursts	many very short CPU bursts	a few very short I/O bursts			many very short CPU bursts
A process is selected from the ______ queue by the ________ scheduler, to be executed.	blocked , short term	wait, long term	ready, short term	ready, long term			ready, short term
In the following cases non – preemptive scheduling occurs : (Choose two)	When a process switches from the running state to the ready state	When a process goes from the running state to the waiting state	When a process switches from the waiting state to the ready state	When a process terminates			When a process terminates
The switching of the CPU from one process or thread to another is called	process switch	task switch	context switch	All of these			All of these
Dispatch latency is	the speed of dispatching a	the time of dispatching a process	the time to stop one process and start	None of these			the time to stop one process and start

	process from running to the ready state	from running to ready state and keeping the CPU idle	running another one				running another one
Scheduling is done so as to :	increase CPU utilization	decrease CPU utilization	keep the CPU more idle	None of these			increase CPU utilization
Scheduling is done so as to	increase the throughput	decrease the throughput	increase the duration of a specific amount of work	None of these			increase the throughput
Turnaround time is :	the total waiting time for a process to finish execution	the total time spent in the ready queue	the total time spent in the running queue	the total time from the completion till the submission of a process			the total time from the completion till the submission of a process
Scheduling is done so as to	increase the turnaround time	decrease the turnaround time	keep the turnaround time same	there is no relation between scheduling and turnaround time			decrease the turnaround time
Waiting time is	the total time in the blocked and waiting	the total time spent in the ready queue	the total time spent in the running queue	the total time from the completion till the			the total time spent in the ready queue

	queues			submission of a process			
Scheduling is done so as to	increase the waiting time	keep the waiting time the same	decrease the waiting time	None of these			decrease the waiting time
Response time is	the total time taken from the submission time till the completion time	the total time taken from the submission time till the first response is produced	the total time taken from submission time till the response is output	None of these			the total time taken from the submission time till the first response is produced
Scheduling is done so as to :	increase the response time	keep the response time the same	decrease the response time	None of these			decrease the response time
Concurrent access to shared data may result in	data consistency	data insecurity	data inconsistency	None of these			data inconsistency
A situation where several processes access and manipulate the same data concurrently and the outcome of the execution depends on the particular order in	data consistency	race condition	aging	starvation			race condition

which access takes place is called							
The segment of code in which the process may change common variables, update tables, write into files is known as	program	critical section	non – critical section	synchronizing			critical section
Mutual exclusion implies that	if a process is executing in its critical section, then no other process must be executing in their critical sections	if a process is executing in its critical section, then other processes must be executing in their critical sections	if a process is executing in its critical section, then all the resources of the system must be blocked until it finishes execution	None of these			if a process is executing in its critical section, then no other process must be executing in their critical sections
Bounded waiting implies that there exists a bound on the number of times a process is allowed to	after a process has made a request to enter its critical section and before	when another process is in its critical section	before a process has made a request to enter its critical section	None of these			after a process has made a request to enter its critical section and before the request is

enter its critical section	the request is granted						granted
A minimum of _____ variable(s) is/are required to be shared between processes to solve the critical section problem	one	two	three	four			two
In the bakery algorithm to solve the critical section problem	each process is put into a queue and picked up in an ordered manner	each process receives a number (may or may not be unique) and the one with the lowest number is served next	each process gets a unique number and the one with the highest number is served next	each process gets a unique number and the one with the lowest number is served next			each process receives a number (may or may not be unique) and the one with the lowest number is served next
A monitor is a type of	semaphore	low level synchronization construct	high level synchronization construct	None of these			high level synchronization construct
A monitor is characterized by	a set of programmer defined operators	an identifier	the number of variables in it	All of these			a set of programmer defined operators
Dispatch latency is	the speed of dispatching a	the time of dispatching a process	the time to stop one process and start	None of these			the time to stop one process and start

	process from running to the ready state	from running to ready state and keeping the CPU idle	running another one				running another one

UNIT-III

Questions	opt1	opt2	opt3	opt4	opt5	opt6	answer
In the resource allocation graph Pi->Rj is the	Assignment edge	Process edge	Request edge	None of these			Request edge
In the resource allocation graph Ri->Pj is the	Assignment edge	Process edge	Request edge	None of these			Assignment edge
The resource allocation graph is allocation is applicable for	Single user system	Multi user system	Single instance of a single resource	Multiple instance of single resource			Multiple instance of single resource
Which of these is not a dead lock prevention mechanism	Mutual exclusion	Hold and wait	Safe sequence	No preemption			Safe sequence
The resource allocation graph algorithm has additional edge called	Request edge	Resource edge	Claim edge	Assignment edge			Claim edge
The data structures like available, maximum, allocation and need are available in	Resource allocation	graph algorithm	Banker's algorithm	None of these			graph algorithm
Which of these is not a	Low resource utilization	Starvation	Unsafe state	None of these			Unsafe state

disadvantage of the deadlock prevention method?							
What is the reusable resource?	that can be used by one process at a time and is not depleted by that use	that can be used by more than one process at a time	that can be shared between various threads	none of the mentioned			that can be used by one process at a time and is not depleted by that use
Which of the following condition is required for deadlock to be possible?	mutual exclusion	a process may hold allocated resources while awaiting assignment of other resources	no resource can be forcibly removed from a process holding it	all of the mentioned			all of the mentioned

A system is in the safe state if	the system can allocate resources to each process in some order and still avoid a deadlock	there exist a safe sequence	both (a) and (b)	none of the mentioned			both (a) and (b)
The circular wait condition can be prevented by	defining a linear ordering of resource types	using thread	using pipes	all of the mentioned			defining a linear ordering of resource types
Which one of the following is the deadlock avoidance algorithm?	banker’s algorithm	round-robin algorithm	elevator algorithm	karn’s algorithm			banker’s algorithm
What is the drawback of banker’s algorithm?	in advance processes rarely know that how much resource they will need	the number of processes changes as time progresses	resource once available can disappear	all of the mentioned			all of the mentioned

For effective operating system, when to check for deadlock?	every time a resource request is made	at fixed time intervals	both (a) and (b)	none of the mentioned			both (a) and (b)
A problem encountered in multitasking when a process is perpetually denied necessary resources is called	deadlock	starvation	inversion	aging			starvation
Which one of the following is a visual (mathematical) way to determine the deadlock occurrence?	resource allocation graph	starvation graph	inversion graph	none of the mentioned			resource allocation graph
To avoid deadlock	there must be a fixed number of resources to allocate	resource allocation must be done only once	all deadlocked processes must be aborted	inversion technique can be used			there must be a fixed number of resources to allocate

The number of resources requested by a process	must always be less than the total number of resources available in the system	must always be equal to the total number of resources available in the system	must not exceed the total number of resources available in the system	must exceed the total number of resources available in the system			must not exceed the total number of resources available in the system
The request and release of resources are ____________	command line statements	interrupts	system calls	special programs			system calls
Multithreaded programs are :	lesser prone to deadlocks	more prone to deadlocks	not at all prone to deadlocks	none of the mentioned			more prone to deadlocks
For Mutual exclusion to prevail in the system	at least one resource must be held in a non sharable mode	the processor must be a uniprocessor rather than a multiprocessor	there must be at least one resource in a sharable mode	All of these			at least one resource must be held in a non sharable mode

For a Hold and wait condition to prevail	A process must be not be holding a resource, but waiting for one to be freed, and then request to acquire it	A process must be holding at least one resource and waiting to acquire additional resources that are being held by other processes	A process must hold at least one resource and not be waiting to acquire additional resources	None of these			A process must be holding at least one resource and waiting to acquire additional resources that are being held by other processes
Deadlock prevention is a set of methods	to ensure that at least one of the necessary conditions cannot hold	to ensure that all of the necessary conditions do not hold	to decide if the requested resources for a process have to be given or not	to recover from a deadlock			to ensure that at least one of the necessary conditions cannot hold
For non sharable resources like a printer, mutual exclusion	must exist	must not exist	may exist	None of these			must exist
For sharable resources, mutual	is required	is not required	None of these				is not required

exclusion							
To ensure that the hold and wait condition never occurs in the system, it must be ensured that	whenever a resource is requested by a process, it is not holding any other resources	each process must request and be allocated all its resources before it begins its execution	a process can request resources only when it has none	All of these			All of these
The disadvantage of a process being allocated all its resources before beginning its execution is	Low CPU utilization	Low resource utilization	Very high resource utilization	None of these			Low resource utilization
To ensure no preemption, if a process is holding some resources and requests another resource that cannot be immediately allocated	then the process waits for the resources be allocated to it	the process keeps sending requests until the resource is allocated to it	the process resumes execution without the resource being allocated to it	then all resources currently being held are preempted			then all resources currently being held are preempted

to it							
One way to ensure that the circular wait condition never holds is to	impose a total ordering of all resource types and to determine whether one precedes another in the ordering	to never let a process acquire resources that are held by other processes	to let a process wait for only one resource at a time	All of these			impose a total ordering of all resource types and to determine whether one precedes another in the ordering
Given a priori information about the ________ number of resources of each type that maybe requested for each process, it is possible	minimum	average	maximum	approximate			maximum

to construct an algorithm that ensures that the system will never enter a deadlock state.							
A deadlock avoidance algorithm dynamically examines the ___________, to ensure that a circular wait condition can never exist.	resource allocation state	system storage state	operating system	resources			resource allocation state
A state is safe, if	the system does not crash due to deadlock occurrence	the system can allocate resources to each process in some order and still avoid a deadlock	the state keeps the system protected and safe	All of these			the system can allocate resources to each process in some order and still avoid a deadlock

A system is in a safe state only if there exists a	safe allocation	safe resource	safe sequence	All of these			safe sequence
All unsafe states are :	deadlock	not deadlock	fatal	none of the mentioned			not deadlock
If no cycle exists in the resource allocation graph :	then the system will not be in a safe state	then the system will be in a safe state	either a or b	None of these			then the system will be in a safe state
The resource allocation graph is not applicable to a resource allocation system	with multiple instances of each resource type	with a single instance of each resource type	Both a and b	None of these			with multiple instances of each resource type
The Banker's algorithm is _______________ than the resource allocation graph algorithm	less efficient	more efficient	None of these				less efficient
The content of the matrix Need is :	Allocation – Available	Max – Available	Max – Allocation	Allocation – Max			Max – Allocation
The wait-for graph is a deadlock detection algorithm that is	all resources have a single instance	all resources have multiple instances	both a and b				all resources have a single instance

applicable when							
An edge from process Pi to Pj in a wait for graph indicates that :	Pi is waiting for Pj to release a resource that Pi needs	Pj is waiting for Pi to release a resource that Pj needs	Pi is waiting for Pj to leave the system	Pj is waiting for Pi to leave the system			Pi is waiting for Pj to release a resource that Pi needs
If the wait for graph contains a cycle :	then a deadlock does not exist	then a deadlock exists	then the system is in a safe state	either b or c			then a deadlock exists
If deadlocks occur frequently, the detection algorithm must be invoked ________.	rarely	frequently	none of the mentioned				frequently
The disadvantage of invoking the detection algorithm for every request is	overhead of the detection algorithm due to consumption of memory	excessive time consumed in the request to be allocated memory	considerable overhead in computation time	All of these			considerable overhead in computation time
A deadlock eventually cripples system	increase	drop	stay still	None of these			drop

throughput and will cause the CPU utilization to _______.							
A computer system has 6 tape drives, with ‘n’ processes competing for them. Each process may need 3 tape drives. The maximum value of ‘n’ for which the system is guaranteed to be deadlock free is :	2	3	4	1			2
A system has 3 processes sharing 4 resources. If each process needs a maximum of 2 units then, deadlock :	can never occur	may occur	has to occur	None of these			can never occur
‘m’ processes share ‘n’ resources of the same type. The	can never occur	may occur	has to occur	None of these			can never occur

maximum need of each process doesn’t exceed ‘n’ and the sum of all their maximum needs is always less than m+n. In this setup, deadlock :							
Physical memory is broken into fixed-sized blocks called ________.	frames	pages	backing store	None of these			frames
Logical memory is broken into blocks of the same size called _______	frames	pages	backing store	None of these			pages
The __________ is used as an index into the page table	frame bit	page number	page offset	frame offset			page number
The _____ table contains the base address of each page in physical memory.	process	memory	page	frame			page

The size of a page is typically :	varied	power of 2	power of 4	None of these			power of 2
If the size of logical address space is 2 to the power of m, and a page size is 2 to the power of n addressing units, then the high order _____ bits of a logical address designate the page number, and the ____ low order bits designate the page offset.	m, n	n, m	m – n, m	m – n, n			m – n, n
With paging there is no ________ fragmentation.	internal	external	either type of	None of these			external
The operating system maintains a ______ table that keeps track of how many frames have been allocated, how many	page	mapping	frame	memory			frame

are there, and how many are available							
Paging increases the _______ time.	waiting	execution	context – switch	All of these			context – switch
Smaller page tables are implemented as a set of ________.	queues	stacks	counters	registers			registers
The page table registers should be built with _______.	very low speed logic	very high speed logic	a large memory space	None of these			very high speed logic
For larger page tables, they are kept in main memory and a ___________ points to the page table.	page table base register	page table base pointer	page table register pointer	page table base			page table base register
For every process there is a __________	page table	copy of page table	pointer to page table	All of these			page table

UNIT-IV

Questions	opt1	opt2	opt3	opt4
Because of virtual memory, the memory can be shared among	processes	threads	instructions	none of t mentione

_____ is the concept in which a process is copied into main memory from the secondary memory according to the requirement.	Paging	Demand paging	Segmentation	Swapping			Demand
The pager concerns with the	individual page of a process	entire process	entire thread	first page of a process			entire pro
Swap space exists in	primary memory	secondary memory	CPU	none of the mentioned			secondar memory
When a program tries to access a page that is mapped in address space but not loaded in physical memory, then	segmentation fault occurs	fatal error occurs	page fault occurs	no error occurs			page fau occurs
Effective access time is directly proportional to	page-fault rate	hit ratio	memory access time	none of the mentioned			page-fau
In FIFO page replacement algorithm, when a page must be replaced	oldest page is chosen	newest page is chosen	random page is chosen	none of the mentioned			oldest pa chosen
Which algorithm chooses the page that has not been used for the longest period of time whenever the page required to be replaced?	first in first out algorithm	additional reference bit algorithm	least recently used algorithm	counting based page replacement algorithm			least rece used algo
A process is thrashing if	it is spending more time paging than executing	it is spending less time paging than executing	page fault occurs	swapping can not take place			it is spen more tim paging th executing
Working set model for page replacement is based on the assumption of	modularity	locality	globalization	random access			locality
Error handler codes, to handle **unusual** errors are :	almost never executed	executed very often	executed periodically	None of these			almost ne executed
In virtual memory. the programmer __________ of overlays.	has to take care	does not have to take care	None of these				does not take care
The instruction being executed, must be in :	physical memory	logical memory	None of these				physical memory
Virtual memory is normally implemented by ________.	demand paging	buses	virtualization	All of these			demand
Segment replacement algorithms are more complex than page replacement algorithms because :	Segments are better than pages	Pages are better than segments	Segments have variable sizes	Segments have fixed sizes			Segment variable s
A swapper manipulates ___________, whereas the pager is concerned with individual _______ of a process.	the entire process, parts	all the pages of a process, segments	the entire process, pages	None of these			the entire process,
Because of virtual memory, the memory can be shared among	processes	threads	instructions	none of the mentioned			processe

_____ is the concept in which a process is copied into main memory from the secondary memory according to the requirement.	Paging	Demand paging	Segmentation	Swapping			Demand
The pager concerns with the	individual page of a process	entire process	entire thread	first page of a process			individua of a proc
Swap space exists in	primary memory	secondary memory	CPU	none of the mentioned			secondar memory
When a program tries to access a page that is mapped in address space but not loaded in physical memory, then	segmentation fault occurs	fatal error occurs	page fault occurs	no error occurs			page fau occurs
Effective access time is directly proportional to	page-fault rate	hit ratio	memory access time	none of the mentioned			page-fau
In FIFO page replacement algorithm, when a page must be replaced	oldest page is chosen	newest page is chosen	random page is chosen	none of the mentioned			oldest pa chosen
Which algorithm chooses the page that has not been used for the longest period of time whenever the page required to be replaced?	first in first out algorithm	additional reference bit algorithm	least recently used algorithm	counting based page replacement algorithm			least rece used alg
A process is thrashing if	it is spending more time paging than executing	it is spending less time paging than executing	page fault occurs	swapping can not take place			it is spen more tim paging th executing
Working set model for page replacement is based on the assumption of	modularity	locality	globalization	random access			locality
When using counters to implement LRU, we replace the page with the :	smallest time value	largest time value	greatest size	None of the mentioned			smallest value
There is a set of page replacement algorithms that can never exhibit Belady's Anomaly, called :	queue algorithms	stack algorithms	string algorithms	None of the mentioned			stack alg
Increasing the RAM of a computer typically improves performance because:	Virtual memory increases	Larger RAMs are faster	Fewer page faults occur	None of the mentioned			Fewer pa faults occ
The essential content(s) in each entry of a page table is / are :	Virtual page number	Page frame number	Both virtual page number and page frame number	Access right information			Page fra number
The minimum number of page frames that must be allocated to a running process in a virtual memory environment is determined by :	the instruction set architecture	page size	physical memory size	number of processes in memory			the instru set archit

The reason for using the LFU page replacement algorithm is :	an actively used page should have a large reference count	a less used page has more chances to be used again	it is extremely efficient and optimal	All of the mentioned			an active page sho have a la reference
The reason for using the MFU page replacement algorithm is :	an actively used page should have a large reference count	a less used page has more chances to be used again	it is extremely efficient and optimal	All of the mentioned			a less us page has chances used aga
The implementation of the LFU and the MFU algorithm is very uncommon because :	they are too complicated	they are optimal	they are expensive	All of the mentioned			they are expensiv
The minimum number of frames to be allocated to a process is decided by the :	the amount of available physical memory	Operating System	instruction set architecture	None of the mentioned			instructio architectu
When a page fault occurs before an executing instruction is complete :	the instruction must be restarted	the instruction must be ignored	the instruction must be completed ignoring the page fault	None of the mentioned			the instru must be restarted
Consider a machine in which all memory reference instructions have only one memory address, for them we need atleast _____ frame(s).	one	two	three	None of the mentioned			two
The maximum number of frames per process is defined by :	the amount of available physical memory	Operating System	instruction set architecture	None of the mentioned			the amou available physical memory
The algorithm in which we split m frames among n processes, to give everyone an equal share, m/n frames is known as :	proportional allocation algorithm	equal allocation algorithm	split allocation algorithm	None of the mentioned			equal allo algorithm
The algorithm in which we allocate memory to each process according to its size is known as :	proportional allocation algorithm	equal allocation algorithm	split allocation algorithm	None of the mentioned			proportio allocation algorithm
With either equal or proportional algorithm, a high priority process is treated ___________ a low priority process.	greater than	same as	lesser than	None of the mentioned			same as
_________ replacement allows a process to select a replacement frame from the set of all frames, even if the frame is currently allocated to some other process.	Local	Universal	Global	Public			Global
_________ replacement allows each process to only	Local	Universal	Global	Public			Local

select from its own set of allocated frames.							
One problem with the global replacement algorithm is that :	it is very expensive	many frames can be allocated to a process	only a few frames can be allocated to a process	a process cannot control its own page – fault rate			a process cannot co its own p fault rate
________ replacement generally results in greater system throughput.	Local	Global	Universal	Public			Global
A process is thrashing if :	it spends a lot of time executing, rather than paging	it spends a lot of time paging, than executing	it has no memory allocated to it	None of these			it spends time pagi than exec
Thrashing _______ the CPU utilization.	increases	keeps constant	decreases	None of these			decrease
A locality is :	a set of pages that are actively used together	a space in memory	an area near a set of processes	None of these			a set of p that are a used toge
When a subroutine is called,	it defines a new locality	it is in the same locality from where it was called	it does not define a new locality	b and c			it defines locality
A program is generally composed of several different localities, which _____ overlap.	may	must	do not	must not			may
In the working set model, for : 2 6 1 5 7 7 7 7 5 1 6 2 3 4 1 2 3 4 4 4 3 4 3 4 4 4 1 3 2 3 if DELTA = 10, then the working set at time t1 (….7 5 1) is :	{1, 2, 4, 5, 6}	{2, 1, 6, 7, 3}	{1, 6, 5, 7, 2}	{1, 2, 3, 4, 5}			{1, 6, 5, 7
The accuracy of the working set depends on the selection of :	working set model	working set size	memory size	number of pages in memory			working s
If working set window is too small :	it will not encompass entire locality	it may overlap several localities	it will cause memory problems	None of these			it will not encompa entire loc
If working set window is too large :	it will not encompass entire locality	it may overlap several localities	it will cause memory problems	None of these			it may ov several lo
If the sum of the working – set sizes increases, exceeding the total number of available frames :	then the process crashes	the memory overflows	the system crashes	the operating system selects a process to suspend			the opera system s process t suspend
Which principle states that programs, users and even the systems be given just enough privileges to perform their task?	principle of operating system	principle of least privilege	principle of process scheduling	none of the mentioned			principle privilege

_______ is an approach to restricting system access to authorized users.	Role-based access control	Process-based access control	Job-based access control	none of the mentioned			Role-bas access c
For system protection, a process should access	all the resources	only those resources for which it has authorization	few resources but authorization is not required	all of the mentioned			only thos resource which it h authoriza
If the set of resources available to the process is fixed throughout the process's lifetime then its domain is	static	dynamic	neither static nor dynamic	none of the mentioned			static
Access matrix model for user authentication contains	a list of objects	a list of domains	a function which returns an object's type	all of the mentioned			all of the mentione

UNIT-V

Questions	**opt1**	**opt2**	**opt3**	**opt4**
_______ is a unique tag, usually a number, identifies the file within the file system.	File identifier	File name	File type	none of tl mentione
To create a file	allocate the space in file system	make an entry for new file in directory	both (a) and (b)	none of tl mentione
By using the specific system call, we can	open the file	read the file	write into the file	all of the mentione
File type can be represented by	file name	file extension	file identifier	none of tl mentione
Which file is a sequence of bytes organized into blocks understandable by the system's linker?	object file	source file	executable file	text file
What is the mounting of file system?	crating of a filesystem	deleting a filesystem	attaching portion of the file system into a directory structure	removing portion of file syster into a dire structure
Mapping of file is managed by	file metadata	page table	virtual memory	file syster
Mapping of network file system protocol to local file system is done by	network file system	local file system	volume manager	remote m
Which one of the following explains the sequential file access method?	random access according to the given byte	read bytes one at a time, in order	read/write sequentially by record	read/write randomly record

	number						
file system fragmentation occurs when	unused space or single file are not contiguous	used space is not contiguous	unused space is non-contiguous	multiple files are non-contiguous			unused spa or single fil are not contiguous
Management of metadata information is done by	file-organisation module	logical file system	basic file system	application programs			logical file system
A file control block contains the information about	file ownership	file permissions	location of file contents	all of the mentioned			all of the mentioned
Which table contains the information about each mounted volume?	mount table	system-wide open-file table	per-process open-file table	all of the mentioned			all of the mentioned
To create a new file application program calls	basic file system	logical file system	file-organisation module	none of the mentioned			logical file system
When a process closes the file	per-process table entry is removed	system wide entry's open count is decremented	both (a) and (b)	none of the mentioned			both (a) an (b)
What is raw disk?	disk without file system	empty disk	disk lacking logical file system	disk having file system			disk withou file system
The data structure used for file directory is called	mount table	hash table	file table	process table			hash table
In which type of allocation method each file occupy a set of contiguous block on the disk?	contiguous allocation	dynamic-storage allocation	linked allocation	indexed allocation			contiguous allocation
If the block of free-space list is free then bit will	1	0	Any of 0 or 1	none of the mentioned			1
Which protocol establishes the initial logical connection between a server and a client?	transmission control protocol	user datagram protocol	mount protocol	datagram congestion control protocol			mount prot
The directory can be viewed as a __________, that translates file names into their directory	symbol table	partition	swap space	cache			symbol tab

entries.							
In the single level directory :	All files are contained in different directories all at the same level	All files are contained in the same directory	Depends on the operating system	None of these			All files are contained i the same directory
In the single level directory :	all directories must have unique names	all files must have unique names	all files must have unique owners	All of these			all files mu have uniqu names
In the two level directory structure :	each user has his/her own user file directory	the system has its own master file directory	both a and b	None of these			both a and
The disadvantage of the two level directory structure is that :	it does not solve the name collision problem	it solves the name collision problem	it does not isolate users from one another	it isolates users from one another			it isolates users from another
In the tree structured directories,	the tree has the stem directory	the tree has the leaf directory	the tree has the root directory	All of these			the tree ha the root directory
The current directory contains, most of the files that are :	of current interest to the user	stored currently in the system	not used in the system	not of current interest to the system			of current interest to t user
An absolute path name begins at the :	leaf	stem	current directory	root			root
A relative path name begins at the :	leaf	stem	current directory	root			current directory
In tree structure, when deleting a directory that is not empty :	The contents of the directory are safe	The contents of the directory are also deleted	None of these				The conter of the direc are also deleted
When two users keep a subdirectory in their own directories, the structure being referred to is :	tree structure	cyclic graph directory structure	two level directory structure	acyclic graph directory			acyclic gra directory
A tree structure _______ the sharing of files and directories.	allows	may restrict	restricts	None of these			restricts
The operating system ________ the links when traversing directory trees, to preserve the acyclic structure of the system.	considers	ignores	deletes	None of these			ignores
The deletion of a link, _________ the original file.	deletes	affects	does not affect	None of these			does not a
When keeping a list of all the links/references to a file, and the list is empty, implies that :	the file has no copies	the file is deleted	the file is hidden	None of these			the file is deleted

When a cycle exists, the reference count maybe non zero, even when it is no longer possible to refer to a directory or file, due to _______.	the possibility of one hidden reference	the possibility of two hidden references	the possibility of self referencing	None of these			the possibi of self referencing
In contiguous allocation :	each file must occupy a set of contiguous blocks on the disk	each file is a linked list of disk blocks	all the pointers to scattered blocks are placed together in one location	None of these			each file m occupy a s of contiguo blocks on t disk
In linked allocation :	each file must occupy a set of contiguous blocks on the disk	each file is a linked list of disk blocks	all the pointers to scattered blocks are placed together in one location	None of these			each file is linked list o disk blocks
In indexed allocation :	each file must occupy a set of contiguous blocks on the disk	each file is a linked list of disk blocks	all the pointers to scattered blocks are placed together in one location	None of these			all the poin to scattere blocks are placed together in location
On systems where there are multiple operating system, the decision to load a particular one is done by :	boot loader	boot strap	process control block	file control block			boot loader
The VFS (virtual file system) activates file system specific operations to handle local requests according to their _______.	size	commands	timings	file system types			file system types
The real disadvantage of a linear list of directory entries is the :	size of the linear list in memory	linear search to find a file	it is not reliable	All of these			linear sear to find a file
One difficulty of contiguous allocation is :	finding space for a new file	inefficient	costly	time taking			finding spa for a new fi
A device driver can be thought of as a translator. Its input consists of _____ commands and output consists of _______ instructions.	high level, low level	low level, high level	complex, simple	Both a and c			high level, level
The file organization module knows about :	files	logical blocks of files	physical blocks of files	All of these			All of these
Metadata includes :	all of the file system structure	contents of files	Both a and b	None of these			Both a and
For each file their exists a ___________, that contains information about the file, including ownership, permissions and location of the file	metadata	file control block	process control block	All of these			file control block

contents.							
For processes to request access to file contents, they need to :	they need to run a seperate program	they need special interrupts	implement the open and close system calls	None of these			implement open and close syste calls
During compaction time, other normal system operations _______ be permitted.	can	cannot	is	None of these			cannot
When in contiguous allocation the space cannot be extended easily :	the contents of the file have to be copied to a new space, a larger hole	the file gets destroyed	the file will get formatted and loose all its data	None of these			the content the file hav be copied t new space larger hole
There is no __________ with linked allocation.	internal fragmentation	external fragmentation	starvation	All of these			external fragmentat
The major disadvantage with linked allocation is that :	internal fragmentation	external fragmentation	there is no sequential access	there is only sequential access			there is onl sequential access
If a pointer is lost or damaged in a linked allocation :	the entire file could get damaged	only a part of the file would be affected	there would not be any problems	None of these			the entire f could get damaged
FAT stands for :	File Attribute Transport	File Allocation Table	Fork At Time	None of these			File Allocat Table
By using FAT, random access time is __________.	the same	increased	decreased	not affected			decreased
If the extents are too large, then the problem that comes in is :	internal fragmentation	external fragmentation	starvation	All of these			internal fragmentat
The FAT is used much as a _________.	stack	linked list	data	pointer			linked list
A section of disk at the beginning of each partition is set aside to contain the table in :	FAT	linked allocation	Hashed allocation	indexed allocation			FAT
Each _______ has its own index block.	partition	address	file	All of these			file
Indexed allocation _________ direct access.	supports	does not support	is not related to	None of these			supports