

**NAAC ACCREDITED**



**तेजस्वि नावधीतमस्तु**  
ISO 9001:2008 & 14001:2004

# FAIRFIELD

## Institute of Management & Technology

'A' Grade Institute by DHE, Govt. of NCT Delhi and Approved by the Bar Council of India and NCTE

### Reference Material for Three Years

### Bachelor of Computer Application

### Code: 020

### Semester – V



FIMT Campus, Kapashera, New Delhi-110037, Phones : 011-25063208/09/10/11, 25066256/ 57/58/59/60  
Fax : 011-250 63212 Mob. : 09312352942, 09811568155 E-mail : [fimtooffice@gmail.com](mailto:fimtooffice@gmail.com) Website : [www.fimt-ggsipu.org](http://www.fimt-ggsipu.org)

**DISCLAIMER** :FIMT, ND has exercised due care and caution in collecting the data before publishing tis Reference Material. In spite of this ,if any omission,inaccuracy or any other error occurs with regards to the data contained in this reference material, FIMT, ND will not be held responsible or liable. FIMT , ND will be grateful if you could point out any such error or your suggestions which will be of great help for other readers.

**INDEX**

**Three Years**

**Bachelor of Computer Application**

**Code: 020**

**Semester – V**

<b>S.NO.</b>	<b>SUBJECTS</b>	<b>CODE</b>	<b>PG.NO.</b>
<i>1</i>	<i>OPERATING SYSTEM</i>	<i>301</i>	<i>03-125</i>
<i>2</i>	<i>COMPUETR GRAPHICS</i>	<i>303</i>	<i>126-159</i>
<i>3</i>	<i>E-COMMERCE</i>	<i>305</i>	<i>160-195</i>
<i>4</i>	<i>WEB BASED PROGRAMMING</i>	<i>313</i>	<i>196-299</i>

## OPERATING SYSTEM (301)

### UNIT -1

#### COMPUTER SYSTEM AND OPERATING SYSTEM OVERVIEW

##### OVER VIEW OF OPERATING SYSTEM

##### What is an Operating System?

A program that acts as an intermediary between a user of a computer and the computer hardware

Operating system goals:

Execute user programs and make solving user problems easier Make the computer system convenient to use

Use the computer hardware in an efficient manner

##### Computer System Structure

Computer system can be divided into four components • Hardware – provides basic computing resources

CPU, memory, I/O devices • Operating system

Controls and coordinates use of hardware among various applications and users

- Application programs – define the ways in which the system resources are used to solve the computing problems of the users

Word processors, compilers, web browsers, database systems, video games Users

- People, machines, other computers

##### Four Components of a Computer System

##### Operating System Definition

- OS is a **resource allocator**
- 
- 
- 
- 
- 
-

Manages all resources

Decides between conflicting requests for efficient and fair resource use OS is a **control program**

Controls execution of programs to prevent errors and improper use of the computer No universally accepted definition

Everything a vendor ships when you order an operating system” is good approximation But varies wildly “The one program running at all times on the computer” is the **kernel**. Everything else is either a system program (ships with the operating system) or an application program

### Computer Startup

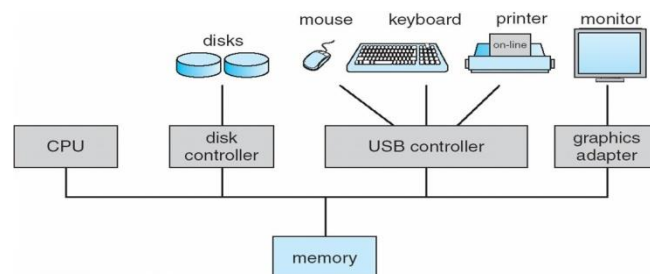
- **bootstrap program** is loaded at power-up or reboot
- 
- Typically stored in ROM or EPROM, generally known as **firmware**

Initializes all aspects of system

Loads operating system kernel and starts execution

### Computer System Organization

- Computer-system operation
- 



One or more CPUs, device controllers connect through common bus providing access to shared memory Concurrent execution of CPUs and devices competing for memory cycles

### Computer-System Operation

- I/O devices and the CPU can execute concurrently
- Each device controller is in charge of a particular device type Each device controller has a local
- buffer
- CPU moves data from/to main memory to/from local buffers I/O is from the device to local buffer of controller

Device controller informs CPU that it has finished its operation by causing An *interrupt*

### Common Functions of Interrupts

- Interrupt transfers control to the interrupt service routine generally, through the **interrupt vector**, which contains the addresses of all the service routines
- Interrupt architecture must save the address of the interrupted instruction

Incoming interrupts are *disabled* while another interrupt is being processed to prevent a *lost interrupt* A *trap* is a software-generated interrupt caused either by an error or a user request

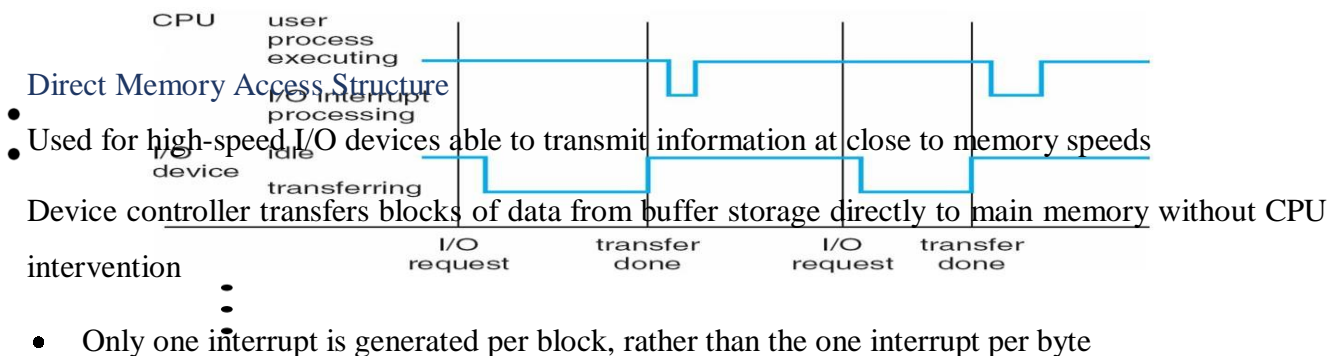
- An operating system is **interrupt driven** **Interrupt Handling**

The operating system preserves the state of the CPU by storing registers and the program counter  
Determines which type of interrupt has occurred:

### Polling

- **Vectored** interrupt system
- Separate segments of code determine what action should be taken for each type of interrupt

### Interrupt Timeline



### Storage Structure

- Main memory – only large storage media that the CPU can access directly
- Secondary storage – extension of main memory that provides large nonvolatile storage capacity
- Magnetic disks – rigid metal or glass platters covered with magnetic recording material

Disk surface is logically divided into **tracks**, which are subdivided into **sectors**

The **disk controller** determines the logical interaction between the device and the computer

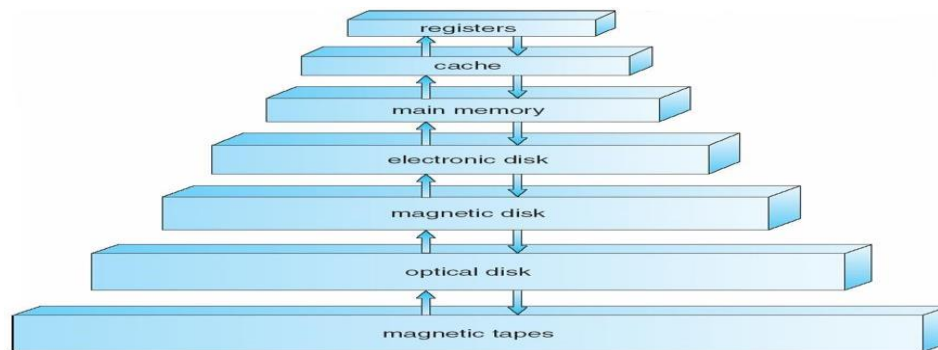
## Storage Hierarchy

- Storage systems organized in hierarchy

- Cost
- Volatility

**Caching** – copying information into faster storage system; main memory can be viewed as a last *cache* for

secondary storage



## Caching

Important principle, performed at many levels in a computer (in hardware, operating system, software) Information in use copied from slower to faster storage temporarily

Faster storage (cache) checked first to determine if information is there If it is, information used directly from the cache (fast)

If not, data copied to cache and used there Cache smaller than storage being cached Cache management important design problem Cache size and replacement policy

## Computer-System Architecture

- Most systems use a single general-purpose processor (PDAs through mainframes) Most systems have
- special-purpose processors as well

Multiprocessors systems growing in use and importance Also known as parallel systems, tightly-coupled systems

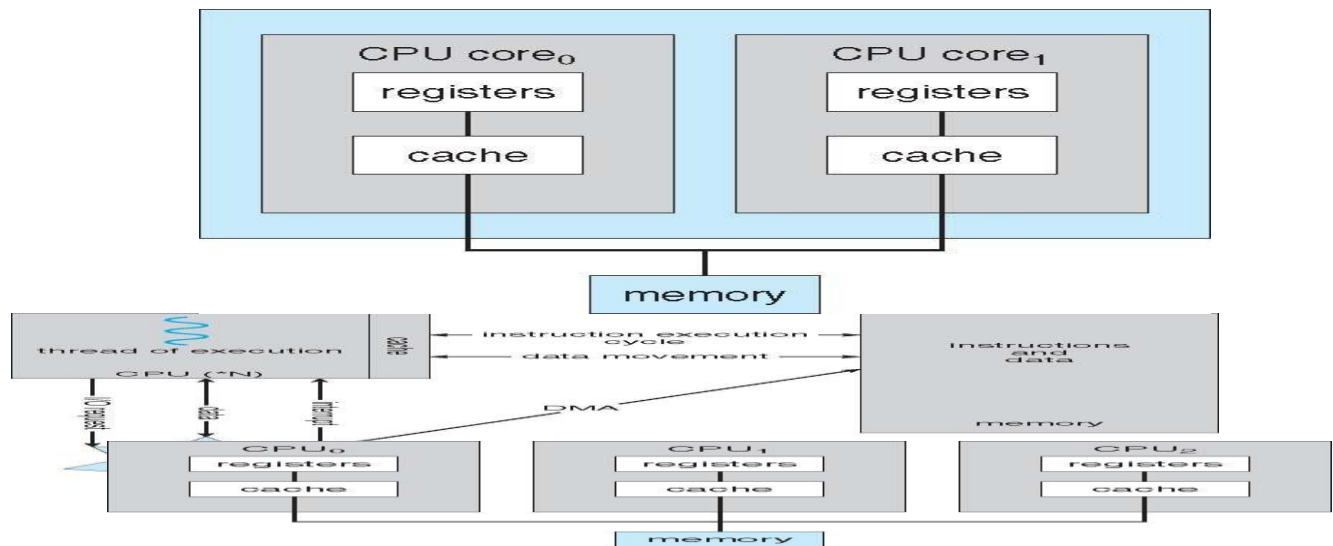
Advantages include

1. Increased throughput
2. Economy of scale

Increased reliability – graceful degradation or fault tolerance Two types

Asymmetric Multiprocessing

2. Symmetric Multiprocessing



A Dual-Core Design

**Clustered Systems**

- Like multiprocessor systems, but multiple systems working together Usually sharing storage via a storage-area network (SAN)

Provides a high-availability service which survives failures Asymmetric clustering has one machine in hot-standby mode

Symmetric clustering has multiple nodes running applications, monitoring each other • Some clusters are for high-performance computing (HPC)

Applications must be written to use parallelization

**Operating System Structure Multiprogramming** needed for efficiency

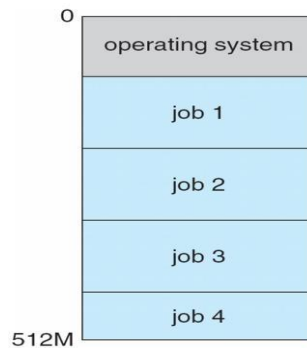
- 
- Single user cannot keep CPU and I/O devices busy at all times
- 
- Multiprogramming organizes jobs (code and data) so CPU always has one to Execute A subset of total jobs in system is kept in memory
- One job selected and run via **job scheduling**
- 
-

When it has to wait (for I/O for example), OS switches to another job

**Timesharing (multitasking)** is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing

- 
- **Response time** should be  $< 1$  second
- 
- Each user has at least one program executing in memory [**process** If several jobs ready to run at the same time [ **CPU scheduling**

If processes don't fit in memory, **swapping** moves them in and out to run **Virtual memory** allows execution of processes not completely in memory **Memory Layout for Multi programmed System.**



#### Operating-System Operations

- 
- Interrupt driven by hardware
- 
- Software error or request creates **exception** or **trap**
- 
- Division by zero, request for operating system service
- Other process problems include infinite loop, processes modifying each Other or the operating system

**Dual-mode** operation allows OS to protect itself and other system components

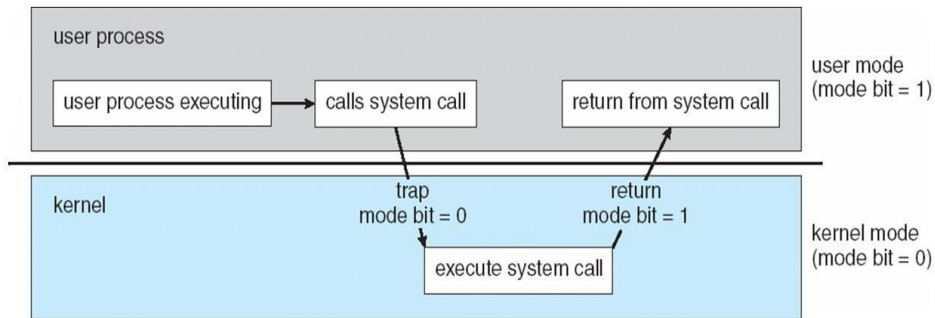
**User mode** and **kernel mode** **Mode bit** provided by hardware

Provides ability to distinguish when system is running user code or kernel code Some instructions designated as **privileged**, only executable in kernel mode System call changes mode to kernel, return from call resets it to user



## Transition from User to Kernel Mode

Timer to prevent infinite loop / process hogging resources Set interrupt after specific period, Operating system decrements counter When counter zero generate an interrupt Set up before scheduling process to regain control or terminate program that exceeds allotted time



### UNIT - 1

## OPERATING SYSTEM FUNCTIONS

### Process Management

- A process is a program in execution. It is a unit of work within the system. Program is a *passive entity*, process is an *active entity*.

Process needs resources to accomplish its task CPU, memory, I/O, files Initialization data

Process termination requires reclaim of any reusable resources

Single-threaded process has one **program counter** specifying location of next instruction to execute  
Process executes instructions sequentially, one at a time, until completion

Multi-threaded process has one program counter per thread

Typically system has many processes, some user, some operating system running concurrently on one or more CPUs

- Concurrency by multiplexing the CPUs among the processes / threads

### Process Management Activities

- 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

### Memory Management

- All data in memory before and after processing All instructions in memory in order to execute
- Memory management determines what is in memory when Optimizing CPU utilization and computer response to users
- Memory management activities
- Keeping track of which parts of memory are currently being used and by whom Deciding which processes (or parts thereof) and data to move into and out of memory Allocating and deallocating memory space as needed

### Storage Management

- OS provides uniform, logical view of information storage Abstracts physical properties to logical storage unit - **file**
- Each medium is controlled by device (i.e., disk drive, tape drive)

Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)

### File-System management

Files usually organized into directories

Access control on most systems to determine who can access what

### OS activities include

Creating and deleting files and directories Primitives to manipulate files and dirs Mapping files onto secondary storage

Backup files onto stable (non-volatile) storage media

### Mass-Storage Management

- Usually disks used to store data that does not fit in main memory or data that must be kept for a “long” period of time

Proper management is of central importance

Entire speed of computer operation hinges on disk subsystem and its algorithms

**MASS STORAGE activities** Free-space management Storage allocation

Disk scheduling

Some storage need not be fast

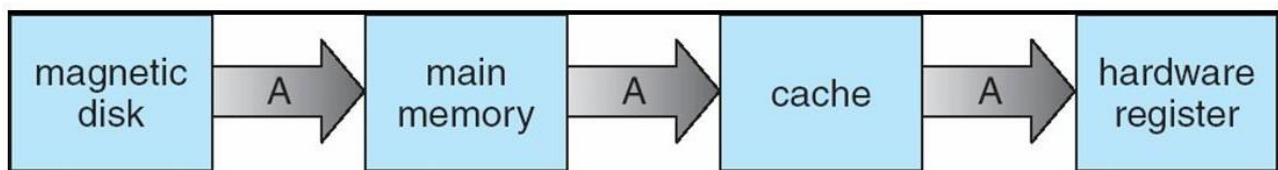
Tertiary storage includes optical storage, magnetic tape Still must be managed

Varies between WORM (write-once, read-many-times) and RW (read-write)

Performance of Various Levels of Storage

**Migration of Integer A from Disk to Register**

- Multitasking environments must be careful to use most recent value, no matter where it is stored in the storage hierarchy



- Multiprocessor environment must provide cache coherency in hardware such that all CPUs have the most recent value in their cache
- Distributed environment situation even more complex Several copies of a datum can exist
- **I/O Subsystem**
- One purpose of OS is to hide peculiarities of hardware devices from the user I/O subsystem
- responsible for

Memory management of I/O including buffering (storing data temporarily while it is being transferred), caching (storing parts of data in faster storage for performance), spooling (the overlapping of output of one job with input of other jobs)

General device-driver interface Drivers for specific hardware devices

## Protection and Security

**Protection** – any mechanism for controlling access of processes or users to resources defined by the OS

**Security** – defense of the system against internal and external attacks

- 
- Huge range, including denial-of-service, worms, viruses, identity theft, theft of service Systems
- generally first distinguish among users, to determine who can do what
- 
- User identities (**user IDs**, security IDs) include name and associated number, one per user User ID then associated with all files, processes of that user to determine access control

Group identifier (**group ID**) allows set of users to be defined and controls managed, then also associated with each process, file

- **Privilege escalation** allows user to change to effective ID with more rights

## DISTRIBUTED SYSTEMS

### Computing Environments Traditional computer

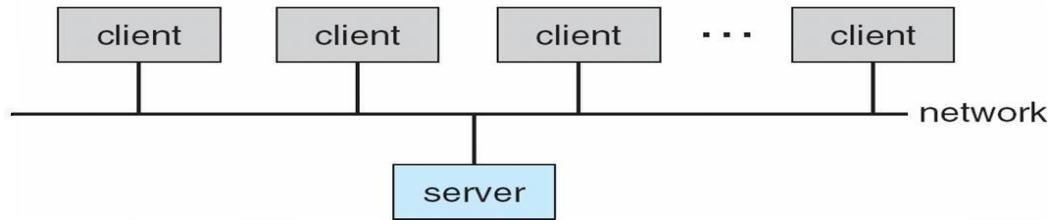
- Blurring over time Office environment
- PCs connected to a network, terminals attached to mainframe or minicomputers providing batch and timesharing

Now portals allowing networked and remote systems access to same resources • Home networks

Used to be single system, then modems Now firewalled, networked

- Client-Server Computing

Dumb terminals supplanted by smart PCs



Many systems now **servers**, responding to requests generated by **clients** **Compute-server** provides an interface to client to request services (i.e. database) **File-server** provides interface for clients to store and retrieve files

### Peer-to-Peer Computing

- Another model of distributed system
- P2P does not distinguish clients and servers Instead all nodes are considered peers
- May each act as client, server or both Node must join P2P network

Registers its service with central lookup service on network, or

Broadcast request for service and respond to requests for service via **discovery protocol**

- Examples include *Napster* and *Gnutella*

### Web-Based Computing

- Web has become ubiquitous PCs most prevalent devices
- More devices becoming networked to allow web access
- New category of devices to manage web traffic among similar servers: **load balancers**

Use of operating systems like Windows 95, client-side, have evolved into Linux and Windows XP, which can be clients and servers

### Open-Source Operating Systems

- Operating systems made available in source-code format rather than just binary closed-source
  - Counter to the copy protection and Digital Rights Management (DRM) movement
  - Started by Free Software Foundation (FSF), which has “copyleft” GNU Public License (GPL)
- Examples include GNU/Linux, BSD UNIX (including core of Mac OS X), and Sun Solaris

## Operating System Services

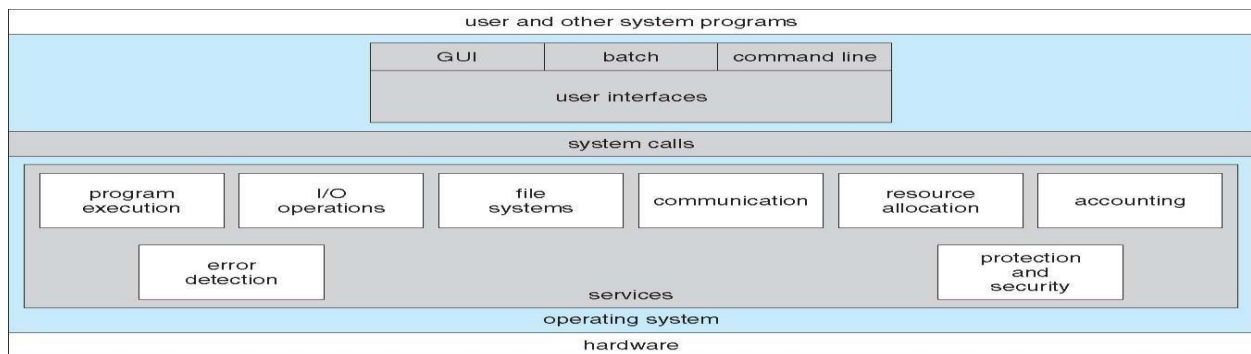
- One set of operating-system services provides functions that are helpful to the user: User interface -
- Almost all operating systems have a user interface (UI)
- Varies between Command-Line (CLI), Graphics User Interface (GUI), Batch

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

I/O operations - A running program may require I/O, which may involve a file or an I/O device

- File-system manipulation - The file system is of particular interest. Obviously, programs need to read and write files and directories, create and delete them, search them, list file information, permission management.

## A View of Operating System Services



## Operating System Services

- One set of operating-system services provides functions that are helpful to the user

Communications – Processes may exchange information, on the same computer or between computers over a network Communications may be via shared memory or through message passing (packets moved by the OS)

- Error detection – OS needs to be constantly aware of possible errors May occur in the CPU and memory hardware, in I/O devices, in user program For each type of error, OS should take the appropriate action to ensure correct and consistent computing Debugging facilities can greatly enhance the user's and programmer's abilities to efficiently use the system

- Another set of OS functions exists for ensuring the efficient operation of the system itself via resource sharing
- **Resource allocation** - When multiple users or multiple jobs running concurrently, resources must be allocated to each of them
  - Many types of resources - Some (such as CPU cycles, main memory, and file storage) may have special allocation code, others (such as I/O devices) may have general request and release code
- **Accounting** - To keep track of which users use how much and what kinds of computer resources
- **Protection and security** - The owners of information stored in a multiuser or networked computer system may want to control use of that information, concurrent processes should not interfere with each other

**Protection** involves ensuring that all access to system resources is controlled

**Security** of the system from outsiders requires user authentication, extends to defending external I/O devices from invalid access attempts

- If a system is to be protected and secure, precautions must be instituted throughout it. A chain is only as strong as its weakest link.

### User Operating System Interface - CLI

- Command Line Interface (CLI) or command interpreter allows direct command entry
  - Sometimes implemented in kernel, sometimes by systems program
  - Sometimes multiple flavors implemented – shells
  - Primarily fetches a command from user and executes it
- Sometimes commands built-in, sometimes just names of programs
- If the latter, adding new features doesn't require shell modification

### User Operating System Interface - GUI

User-friendly desktop metaphor interface Usually mouse, keyboard, and monitor Icons represent files, programs, actions, etc

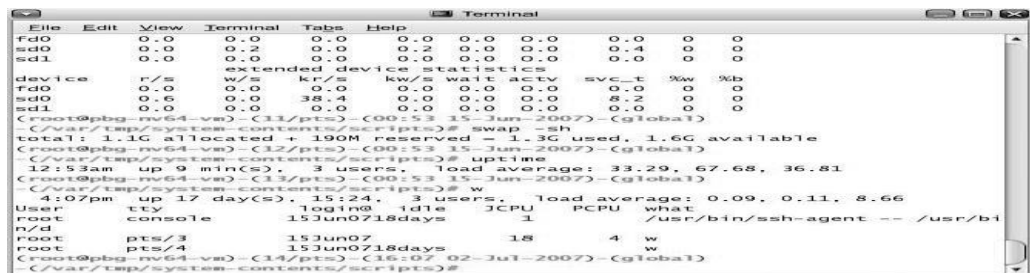
Various mouse buttons over objects in the interface cause various actions (provide information, options, execute function, open directory (known as a folder)

- 
- 
- 
-

Invented at Xerox PARC

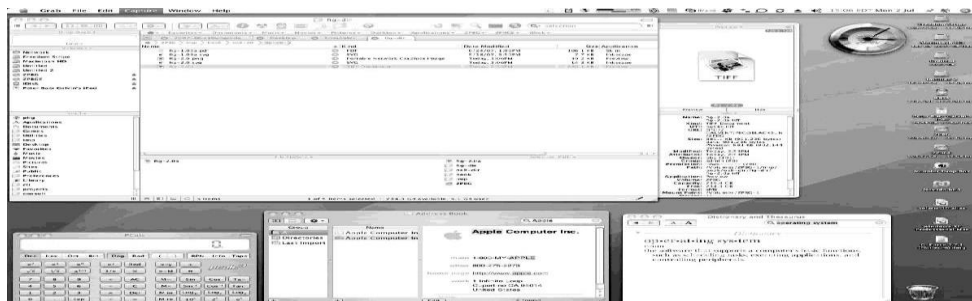
Many systems now include both CLI and GUI interfaces Microsoft Windows is GUI with CLI “command” shell

Apple Mac OS X as “Aqua” GUI interface with UNIX kernel underneath and shells available Solaris is CLI with optional GUI interfaces (Java Desktop, KDE)



```
File Edit View Terminal Tabs Help
rfd0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
sfd0 0.0 0.2 0.0 0.2 0.0 0.0 0.0 0.4 0.0
sdl 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
device r/s w/s kr/s kw/s wait actv svc_t %w %b
rfd0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
sfd0 0.6 0.0 38.4 0.0 0.0 0.0 8.2 0.0 0.0
sdl 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
(root@pbq-nv64-ve)-(11/pts)-(00:53 15-Jun-2007)-(global)
-/var/tmp/system-contents/scripts# swap -sh
total: 1.1G allocated + 190M reserved = 1.3G used, 1.6G available
(root@pbq-nv64-ve)-(12/pts)-(00:53 15-Jun-2007)-(global)
-/var/tmp/system-contents/scripts# uptime
12:53am up 9 min(s), 3 users, load average: 33.29, 67.68, 36.81
(root@pbq-nv64-ve)-(13/pts)-(00:53 15-Jun-2007)-(global)
-/var/tmp/system-contents/scripts# w
4:07pm up 17 day(s), 15:24, 3 users, load average: 0.09, 0.11, 8.66
User tty login idle JCPU PCPU what
root console 15Jun0718days 1 /usr/bin/ssh-agent -- /usr/bi
n/d
root pts/3 15Jun07 18 4 w
root pts/4 15Jun0718days w
(root@pbq-nv64-ve)-(14/pts)-(15:07 02-Jul-2007)-(global)
-/var/tmp/system-contents/scripts#
```

Bourne Shell Command Interpreter



The Mac OS X GUI

## System Calls

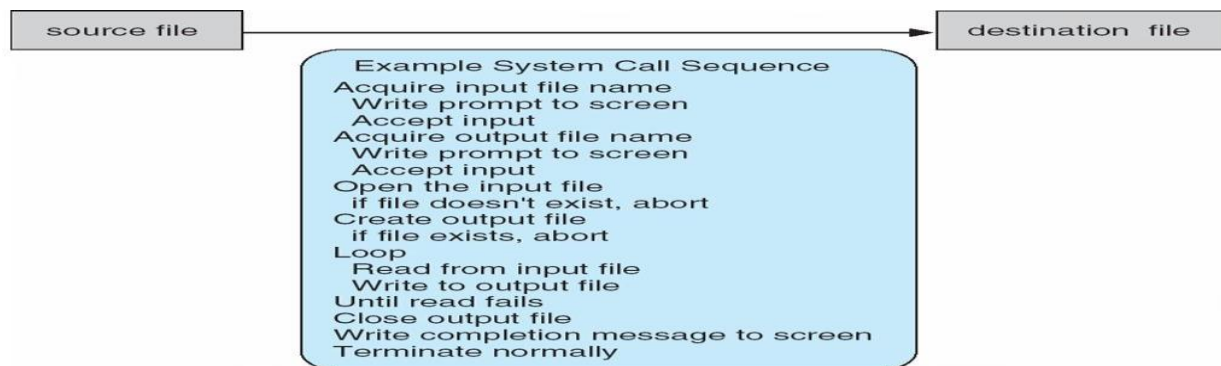
- Programming interface to the services provided by the OS Typically written in a high-level language
- (C or C++)

Mostly accessed by programs via a high-level Application Program Interface (API) rather than direct system call useThree most common APIs are Win32 API for Windows, POSIX API for POSIX-based systems (including virtually all versions of UNIX, Linux, and Mac OS X), and Java API for the Java virtual machine (JVM)



- Why use APIs rather than system calls?(Note that the system-call names used throughout this text are generic)

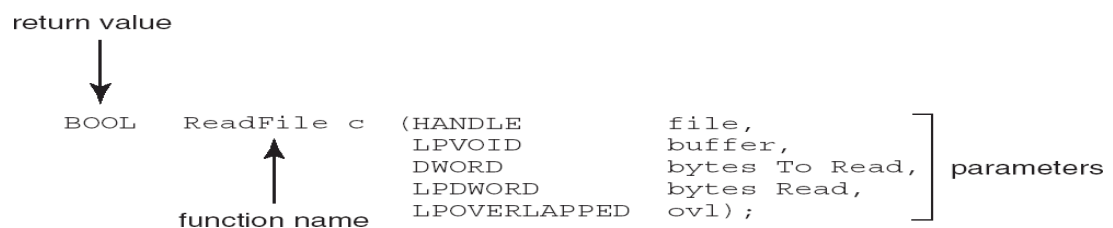
### Example of System Calls



### Example of Standard API

Consider the ReadFile() function in the

Win32 API—a function for reading from a file



A description of the parameters passed to ReadFile() HANDLE file—the file to be read

- LPVOID buffer—a buffer where the data will be read into and written from
- DWORD bytesToRead—the number of bytes to be read into the buffer
- LPDWORD bytesRead—the number of bytes read during the last read
- LPOVERLAPPED ovl—indicates if overlapped I/O is being used

### System Call Implementation

- Typically, a number associated with each system call
- 
-

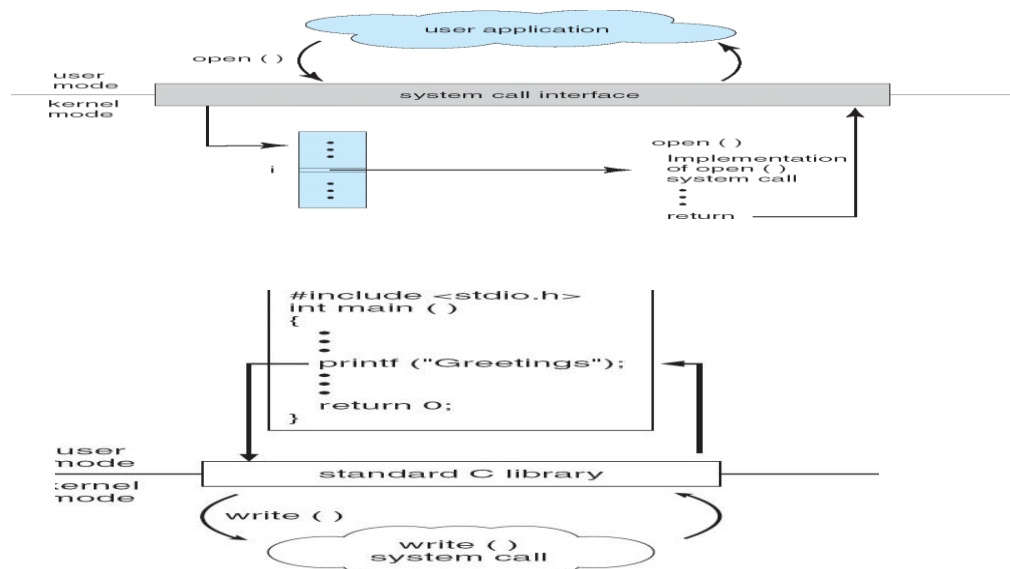
System-call interface maintains a table indexed according to these Numbers

The system call interface invokes intended system call in OS kernel and returns status of the system call and any return values

- The caller need know nothing about how the system call is implemented Just needs to obey API and understand what OS will do as a result call Most details of OS interface hidden from programmer by API

Managed by run-time support library (set of functions built into libraries included with compiler)

### API – System Call – OS Relationship



### System Call Parameter Passing

- Often, more information is required than simply identity of desired system call Exact type and amount of information vary according to OS and call

Three general methods used to pass parameters to the OS Simplest: pass the parameters in *registers*

□ In some cases, may be more parameters than registers

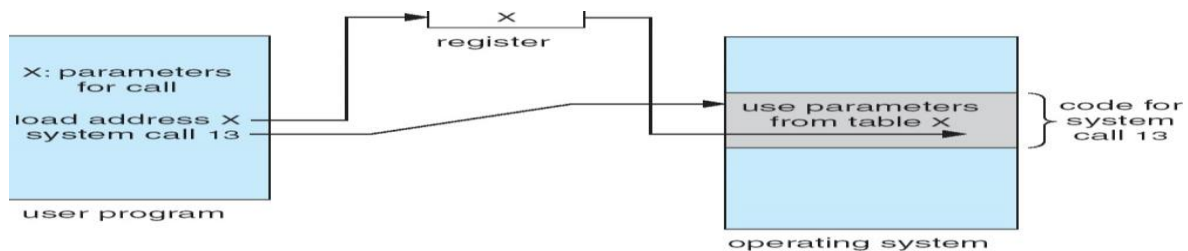
- Parameters stored in a *block*, or table, in memory, and address of block passed as a parameter in a register

This approach taken by Linux and Solaris

- Parameters placed, or *pushed*, onto the *stack* by the program and *popped* off the stack by the operating system

Block and stack methods do not limit the number or length of parameters being passed

### Parameter Passing via Table

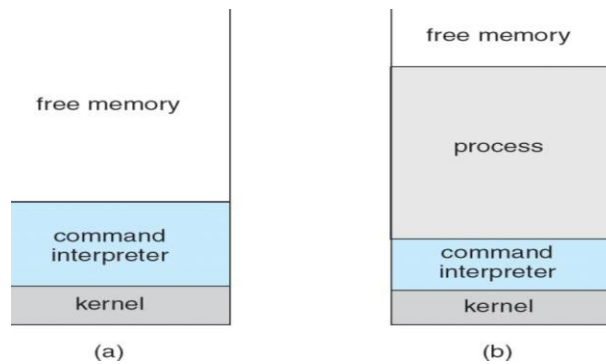


### Types of System Calls

- Process control
- File management
- Device management
- Information maintenance
- Communications
- Protection

	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

### Examples of Windows and Unix System Calls



## MS-DOS execution

(a) At system startup

(b) running a program



## FreeBSD Running Multiple Programs

### System Programs

System programs provide a convenient environment for program development and execution. They can be divided into:

File manipulation Status information File modification

Programming language support Program loading and execution Communications

Application programs

Most users' view of the operation system is defined by system programs, not the actual system calls

Provide a convenient environment for program development and execution

Some of them are simply user interfaces to system calls; others are considerably more complex

File management - Create, delete, copy, rename, print, dump, list, and generally manipulate files and directories

- 
- Status information
- 
- Some ask the system for info - date, time, amount of available memory, disk space, number of users
- Others provide detailed performance, logging, and debugging information

Typically, these programs format and print the output to the terminal or other output devices Some systems implement a registry - used to store and retrieve configuration information

File modification

- 
- Text editors to create and modify files
- Special commands to search contents of files or perform transformations of the text Programming-language support - Compilers, assemblers, debuggers and interpreters sometimes provided
- Program loading and execution- Absolute loaders, relocatable loaders, linkage editors, and overlay- loaders, debugging systems for higher-level and machine language
- Communications - Provide the mechanism for creating virtual connections among processes, users, and computer systems
- Allow users to send messages to one another's screens, browse web pages, send electronic-mail messages, log in remotely, transfer files from one machine to another

### Operating System Design and Implementation

- 
- Design and Implementation of OS not "solvable", but some approaches have proven successful
- Internal structure of different Operating Systems can vary widely
- 
- Start by defining goals and specifications Affected by choice of hardware, type of system *User* goals and *System* goals
- 

User goals – operating system should be convenient to use, easy to learn, reliable, safe, and fast

System goals – operating system should be easy to design, implement, and maintain, as well as flexible, reliable, error-free, and efficient

- 
- Important principle to separate **Policy:** What will be done? **Mechanism:** How to do it?

Mechanisms determine how to do something, policies decide what will be done

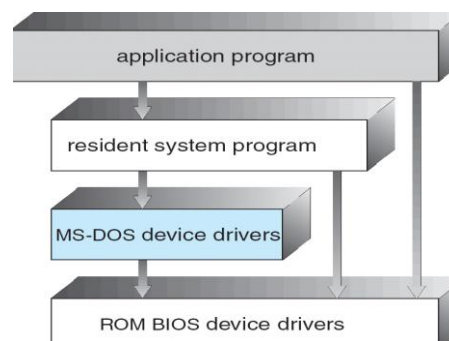
- 
-

The separation of policy from mechanism is a very important principle, it allows maximum flexibility if policy decisions are to be changed later

### Simple Structure

- MS-DOS – written to provide the most functionality in the least space Not divided into modules
- Although MS-DOS has some structure, its interfaces and levels of Functionality are not well separated

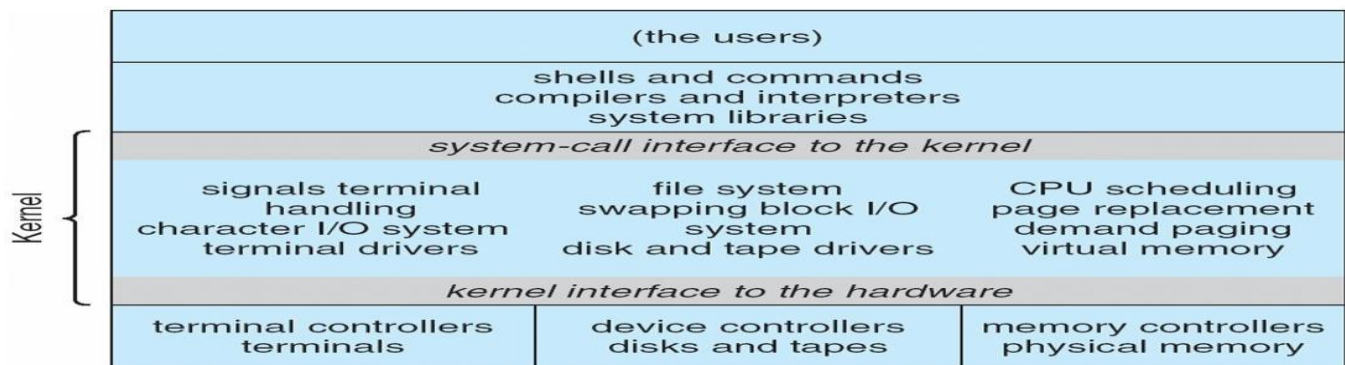
### MS-DOS Layer Structure



### Layered Approach

- The operating system is divided into a number of layers (levels), each built on top of lower layers. The bottom layer (layer 0), is the hardware; the highest (layer N) is the user interface.
- With modularity, layers are selected such that each uses functions (operations) and services of only lower-level layers

## Traditional UNIX System Structure



## UNIX

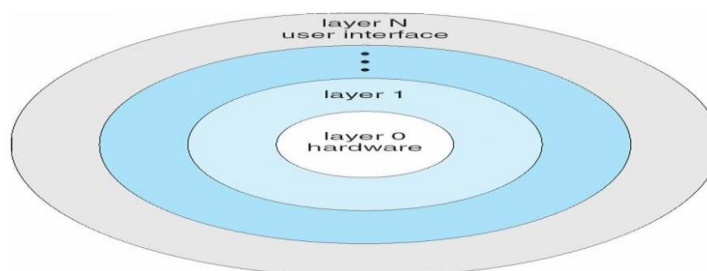
- UNIX – limited by hardware functionality, the original UNIX operating system had limited structuring.

The UNIX OS consists of two separable parts Systems programs

- 
- The kernel

Consists of everything below the system-call interface and above the physical hardware Provides the file system, CPU scheduling, memory management, and other operating-system

functions; a large number of functions for one level

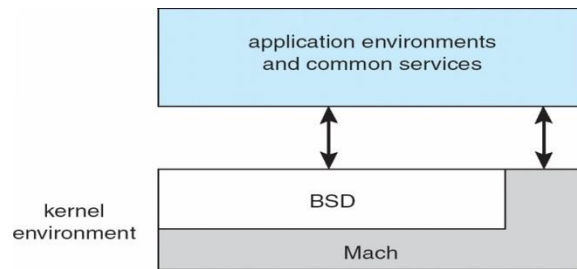


## Layered Operating System

### Micro kernel System Structure

- More secure Detriments:
- Performance overhead of user space to kernel space communication

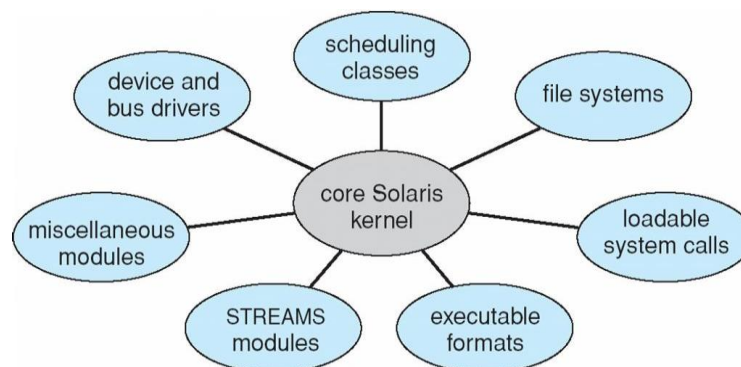
## Mac OS X Structure



## Modules

- Most modern operating systems implement kernel modules Uses object-oriented approach
- Each core component is separate
- Each talks to the others over known interfaces Each is loadable as needed within the kernel Overall,
- similar to layers but with more flexible

## Solaris Modular Approach



## Virtual Machines

- A virtual machine takes the layered approach to its logical conclusion. It treats hardware and the operating system kernel as though they were all hardware
- A virtual machine provides an interface *identical* to the underlying bare hardware
- The operating system host creates the illusion that a process has its own processor and (virtual memory) Each guest provided with a (virtual) copy of underlying computer



## Virtual Machines History and Benefits

- First appeared commercially in IBM mainframes in 1972

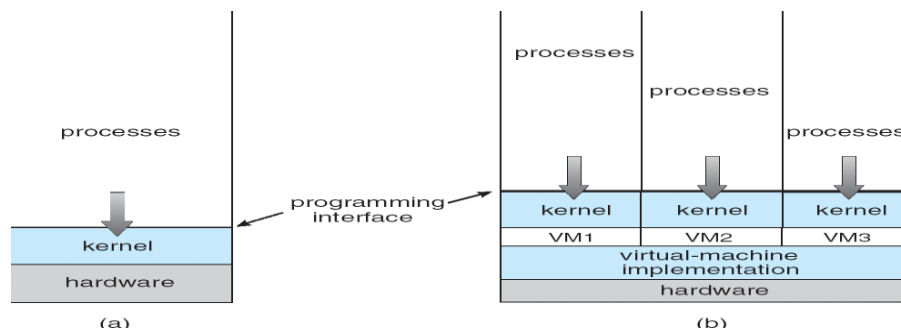
Fundamentally, multiple execution environments (different operating systems) can share the same hardware

Protect from each other

Some sharing of file can be permitted, controlled

Communicate with each other, other physical systems via networking Useful for development, testing

Consolidation of many low-resource use systems onto fewer busier systems

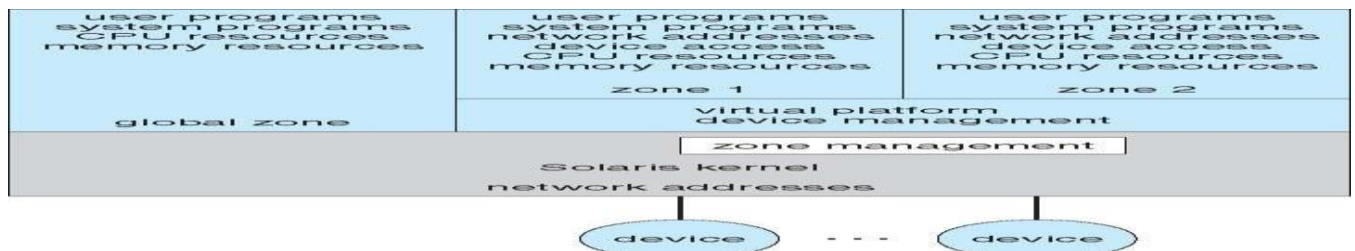


“Open Virtual Machine Format”, standard format of virtual machines, allows a VM to run within many different virtual machine (host) platforms

## Para-virtualization

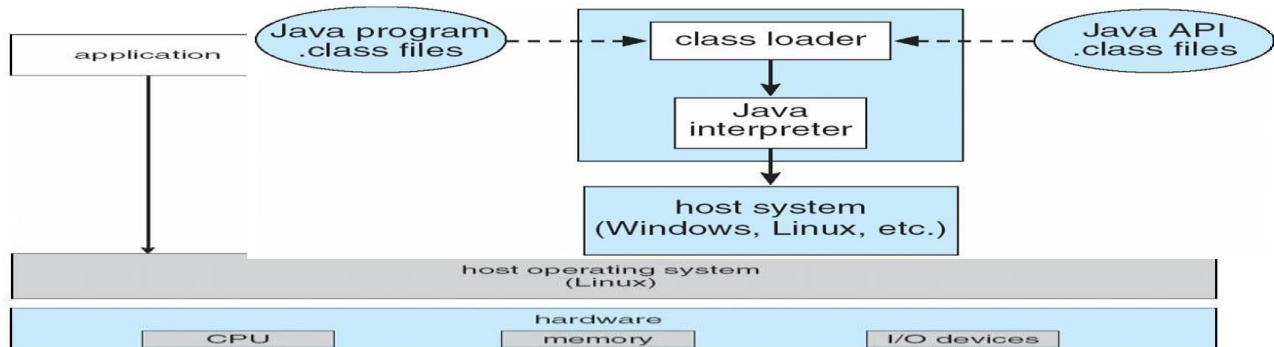
- Presents guest with system similar but not identical to hardware Guest must be modified to run on
- para virtualized hardware

Guest can be an OS, or in the case of Solaris 10 applications running in containers



Solaris 10 with Two Containers

## VMware Architecture



## The Java Virtual Machine

# Operating-System Debugging

- Debugging is finding and fixing errors, or bugs OSES generate log files containing error information
- Failure of an application can generate core dump file capturing memory of the process Operating
- system failure can generate crash dump file containing kernel memory Beyond crashes, performance
- tuning can optimize system performance

Kernighan's Law: "Debugging is twice as hard as writing the code in the rst place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it."

- DTrace tool in Solaris, FreeBSD, Mac OS X allows live instrumentation on production systems  
Probes fire when code is executed, capturing state data and sending it to consumers of those probes

## Solaris 10 dtrace Following System Call

## Operating System Generation

```
# ./all.d 'grep xclock' XEventsQueued
dtrace: script './all.d' matched 52377 probes
CPU FUNCTION
0 --> XEventsQueued U
0 --> XEventsQueued U
0 --> XllTransBytesReadable U
0 --> XllTransBytesReadable U
0 --> XllTransSocketBytesReadable U
0 --> XllTransSocketBytesReadable U
0 --> ioctl U
0 --> ioctl K
0 --> getfd K
0 --> set_active_fd K
0 --> set_active_fd K
0 --> getfd K
0 --> get_udatamodel K
0 --> get_udatamodel K
.
0 --> releasefd K
0 --> clear_active_fd K
0 --> clear_active_fd K
0 --> cv_broadcast K
0 --> cv_broadcast K
0 --> releasefd K
0 --> ioctl K
0 --> ioctl U
0 --> XEventsQueued U
0 --> XEventsQueued U
```

- Operating systems are designed to run on any of a class of machines; the system must be configured

- for each specific computer site

SYSGEN program obtains information concerning the specific configuration of the hardware system

*Booting* – starting a computer by loading the kernel

*Bootstrap program* – code stored in ROM that is able to locate the kernel, load it into memory, and start its execution

### System Boot

- Operating system must be made available to hardware so hardware can start it
- Small piece of code – **bootstrap loader**, locates the kernel, loads it into memory, and starts it
- Sometimes two-step process where **boot block** at fixed location loads bootstrap loader

When power initialized on system, execution starts at a fixed memory location Firmware used to hold initial boot code

## UNIT -2

### PROCESS MANAGEMENT

#### Process Concept

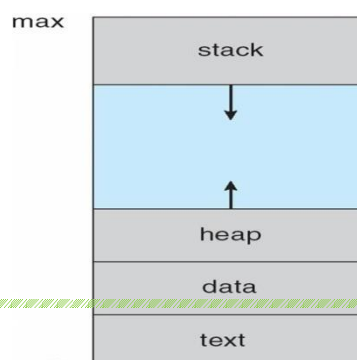
- An operating system executes a variety of programs:
- Batch system – jobs

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



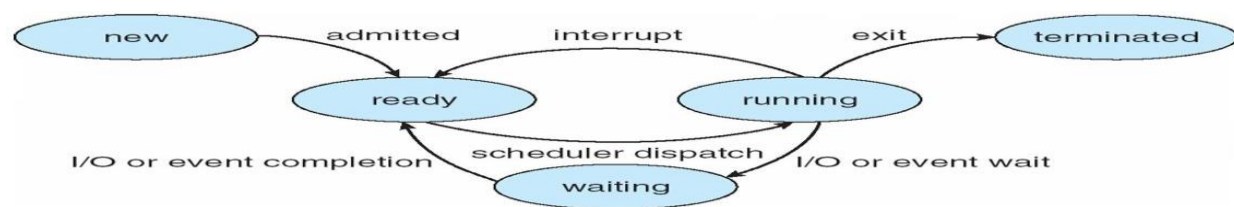
## Process in Memory

### Process State

As a process executes, it changes *state*

- **new**: The process is being created
- **running**: Instructions are being executed
- **waiting**: The process is waiting for some event to occur **ready**: The process is waiting to be assigned to a processor **terminated**: The process has finished execution

### Diagram of Process State



### Process Control Block (PCB)

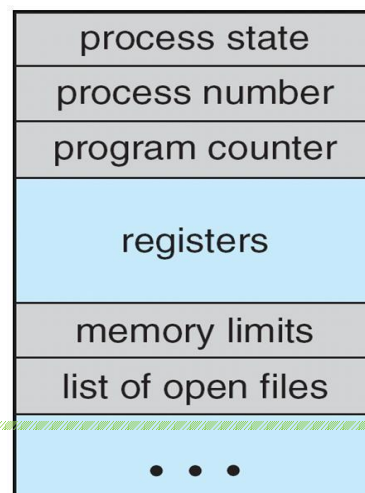
Information associated with each process Process state

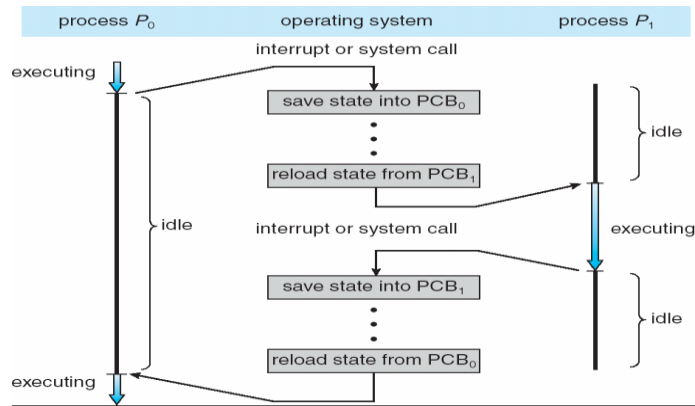
Program counter CPU registers

CPU scheduling information Memory-management information Accounting information

I/O status information

### CPU Switch From Process to Process

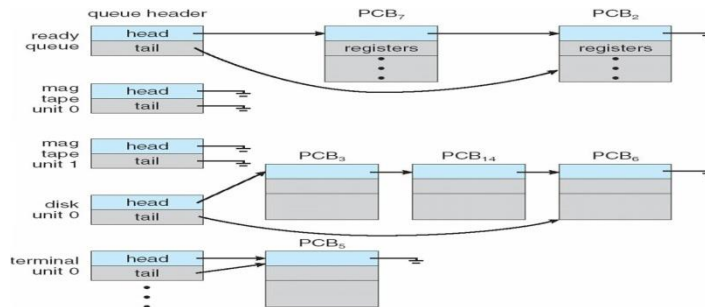




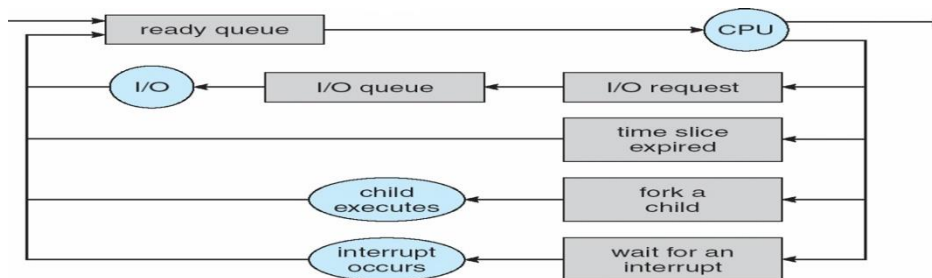
## Process Scheduling Queues

- **Job queue** – set of all processes in the system
- **Ready queue** – set of all processes residing in main memory, ready and waiting to execute
- **Device queues** – set of processes waiting for an I/O device. Processes migrate among the various queues.

## Ready Queue and Various I/O Device Queues



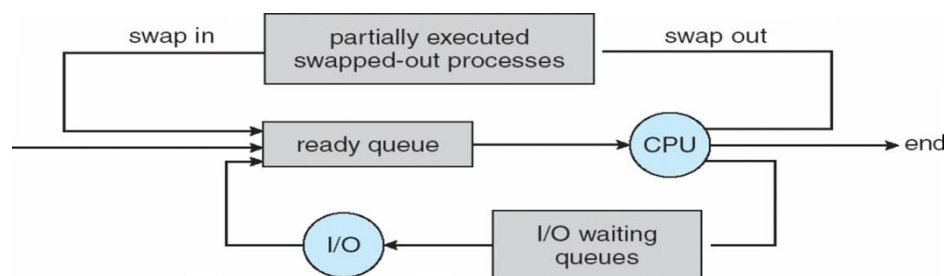
## Representation of Process Scheduling



## Schedulers

- **Long-term scheduler** (or job scheduler) – selects which processes should be brought into the ready queue
- **Short-term scheduler** (or CPU scheduler) – selects which process should be executed next and allocates CPU

### Addition of Medium Term Scheduling



- Short-term scheduler is invoked very frequently (milliseconds) P (must be fast)
- Long-term scheduler is invoked very infrequently (seconds, minutes) P (may be slow) The long-term scheduler controls the *degree of multiprogramming*
- Processes can be described as either:

**I/O-bound process** – spends more time doing I/O than computations, many short CPU bursts

**CPU-bound process** – spends more time doing computations; few very long CPU bursts

### Context Switch

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

### Process Creation

- **Parent** process create **children** processes, which, in turn create other processes, forming a tree of processes

Generally, process identified and managed via a **process identifier (pid)** Resource sharing

Parent and children share all resources Children share subset of parent's resources Parent and child share no resources Execution

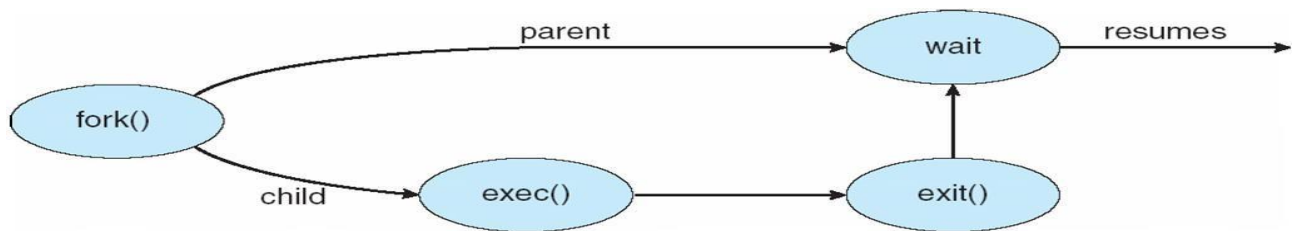
Parent and children execute concurrently Parent waits until children terminate Address space

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 Creation

### C Program Forking Separate Process

```
int main()
{
    pid_t pid;

    /* fork another process */ pid = fork();

    if (pid < 0) { /* error occurred */ fprintf(stderr, "Fork Failed"); exit(-1);
    }

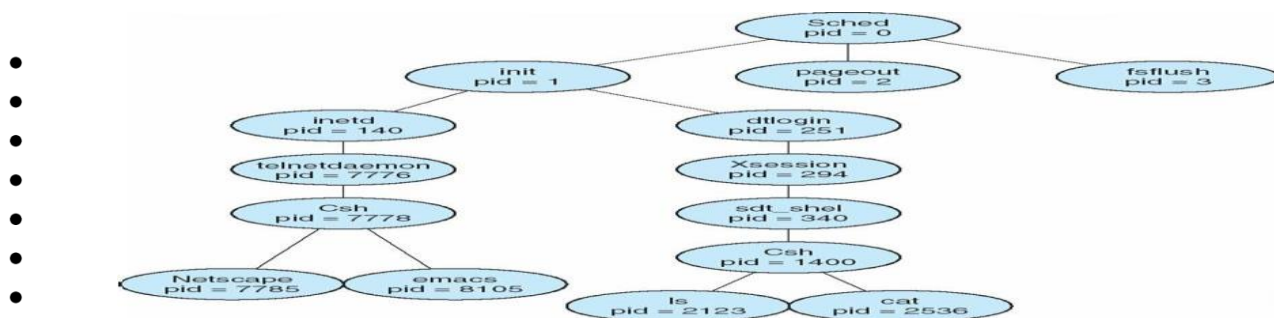
    else if (pid == 0) { /* child process */ execlp("/bin/ls", "ls", NULL);
    }

    else { /* parent process */

        /* parent will wait for the child to complete */ wait (NULL);
    }
}
```

```
printf ("Child Complete"); exit(0);
}
}
```

A tree of processes on a typical Solaris



## Process Termination

Process executes last statement and asks the operating system to delete it (**exit**) Output data from child to parent (via **wait**)

Process' resources are deallocated by operating system Parent may terminate execution of children processes (**abort**) Child has exceeded allocated resources

Task assigned to child is no longer required

If parent is exiting Some operating system do not allow child to continue if its parent terminates All children terminated - **cascading termination**

## Interprocess Communication

Processes within a system may be **independent** or **cooperating**

Cooperating process can affect or be affected by other processes, including sharing data Reasons for cooperating processes:

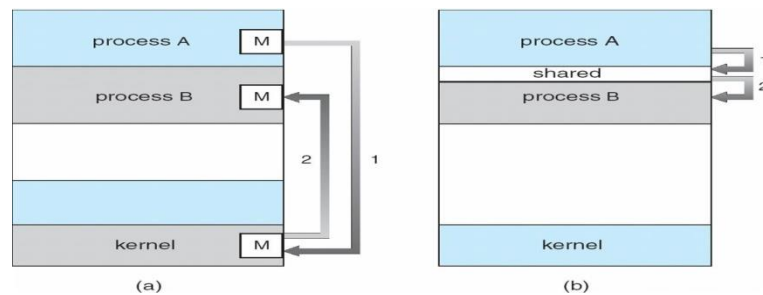
Information sharing Computation speedup Modularity Convenience

Cooperating processes need **interprocess communication (IPC)** Two models of IPC

Shared memory Message passing



## Communications Models



## Cooperating Processes

- **Independent** process cannot affect or be affected by the execution of another process
  - **Cooperating** process can affect or be affected by the execution of another process
- Advantages of process cooperation

Information sharing Computation speed-up Modularity Convenience

## Producer-Consumer Problem

- Paradigm for cooperating processes, *producer* process produces information that is consumed by a *consumer* process
- *unbounded-buffer* places no practical limit on the size of the buffer

*bounded-buffer* assumes that there is a fixed buffer size

## Bounded-Buffer – Shared-Memory Solution

Shared data

```
#define BUFFER_SIZE 10 typedef struct {
```

```
...
```

```
} item;
```

```
item buffer[BUFFER_SIZE]; int in = 0;
```

```
int out = 0;
```

Solution is correct, but can only use BUFFER\_SIZE-1 elements

### Bounded-Buffer – Producer

```
while (true) {  
  
    /* Produce an item */  
  
    while (((in = (in + 1) % BUFFER SIZE count) == out)  
  
    ; /* do nothing -- no free buffers */ buffer[in] = item;  
  
    in = (in + 1) % BUFFER SIZE;  
  
}
```

### Bounded Buffer – Consumer

```
while (true) {  
  
    while (in == out)  
  
    ; // do nothing -- nothing to consume  
  
    // remove an item from the buffer item = buffer[out];  
  
    out = (out + 1) % BUFFER SIZE;  
  
    return item;  
  
}
```

### Interprocess Communication – Message Passing

Mechanism for processes to communicate and to synchronize their actions

Message system – processes communicate with each other without resorting to shared variables IPC facility provides two operations:

**send**(*message*) – message size fixed or variable

**receive**(*message*)

If *P* and *Q* wish to communicate, they need to: establish a *communication link* between them exchange messages via send/receive Implementation of communication link physical (e.g., shared memory, hardware bus) logical (e.g., logical properties)

### Direct Communication

- Processes must name each other explicitly:

- 

- 

- 

- 

-

**send** ( $P, message$ ) – send a message to process P **receive**( $Q, message$ ) – receive a message from process Q Properties of communication link

Links are established automatically

A link is associated with exactly one pair of communicating processes Between each pair there exists exactly one link

The link may be unidirectional, but is usually bi-directional

### Indirect Communication

Messages are directed and received from mailboxes (also referred to as ports) Each mailbox has a unique id

Processes can communicate only if they share a mailbox Properties of communication link

Link established only if processes share a common mailbox A link may be associated with many processes

Each pair of processes may share several communication links Link may be unidirectional or bi-directional

Operations

create a new mailbox

send and receive messages through mailbox destroy a mailbox

Primitives are defined as:

**send**( $A, message$ ) – send a message to mailbox A **receive**( $A, message$ ) – receive a message from mailbox A Mailbox sharing

$P_1$ ,  $P_2$ , and  $P_3$  share mailbox A  $P_1$  sends;  $P_2$  and  $P_3$  receive Who gets the message?

Solutions

Allow a link to be associated with at most two processes Allow only one process at a time to execute a receive operation

Allow the system to select arbitrarily the receiver. Sender is notified who the receiver was.

## Synchronization

- Message passing may be either blocking or non-blocking
- **Blocking** is considered **synchronous**
- **Blocking send** has the sender block until the message is received **Blocking receive** has the receiver block until a message is available **Non-blocking** is considered **asynchronous**

**Non-blocking send** has the sender send the message and continue

**Non-blocking receive** has the receiver receive a valid message or null

## Buffering

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

1. Zero capacity – 0 messages  
Sender must wait for receiver (rendezvous)
2. Bounded capacity – finite length of  $n$  messages  
Sender must wait if link full
3. Unbounded capacity – infinite length  
Sender never waits

## • Examples of IPC Systems - POSIX

- POSIX Shared Memory
- Process first creates shared memory segment
- `segment id = shmget(IPC_PRIVATE, size, S_IRUSR | S_IWUSR);` Process wanting access to that
- shared memory must attach to it `shared memory = (char *) shmat(id, NULL, 0);`
- Now the process could write to the shared memory `printf(shared memory, "Writing to shared memory");`

When done a process can detach the shared memory from its address space `shmdt(shared memory);`

## • Examples of IPC Systems - Mach

- Mach communication is message based Even system calls are messages
- Each task gets two mailboxes at creation- Kernel and Notify Only three system calls needed for
- message transfer `msg_send()`, `msg_receive()`, `msg_rpc()`
- Mailboxes needed for communication, created via `port_allocate()`

## Examples of IPC Systems – Windows XP

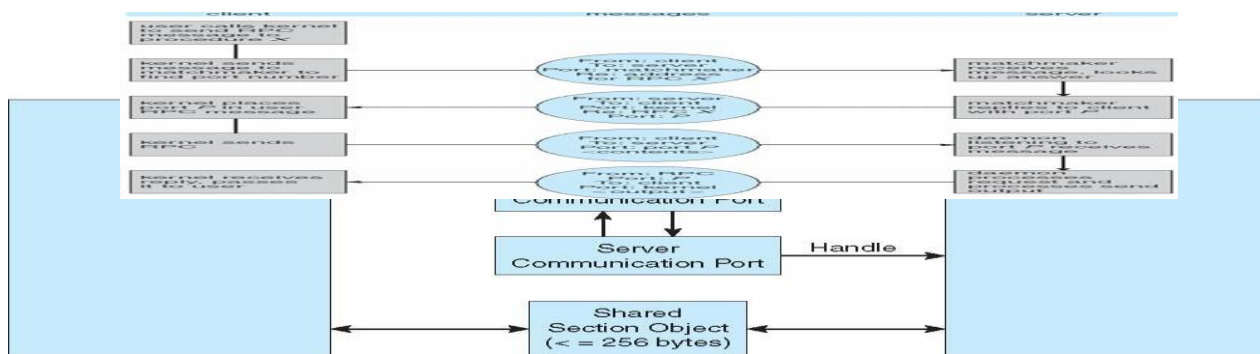
- Message-passing centric via local procedure call (LPC) facility Only works between processes on the same system
- Uses ports (like mailboxes) to establish and maintain communication channels Communication works as follows:

The client opens a handle to the subsystem's connection port object

The client sends a connection request

The server creates two private communication ports and returns the handle to one of them to the client The client and server use the corresponding port handle to send messages or callbacks and to listen for

replies



## Local Procedure Calls in Windows XP

## Communications in Client-Server Systems

- 
- Sockets
- Remote Procedure Calls

Remote Method Invocation (Java)

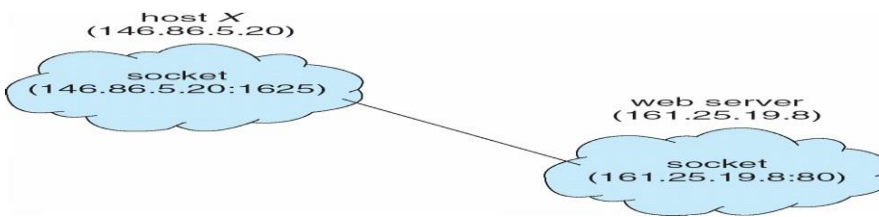
## Sockets

- A socket is defined as an *endpoint for communication*
- 
- Concatenation of IP address and port

The socket **161.25.19.8:1625** refers to port **1625** on host **161.25.19.8**

Communication consists between a pair of sockets

## Socket Communication



## Remote Procedure Calls

- Remote procedure call (RPC) abstracts procedure calls between processes on networked systems
- 
- **Stubs** – client-side proxy for the actual procedure on the server The client-side stub locates the server
- and *marshalls* the parameters

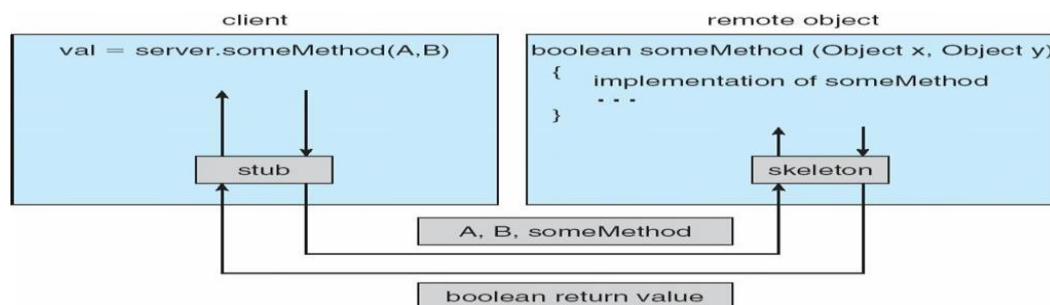
The server-side stub receives this message, unpacks the marshalled parameters, and performs the procedure on the server

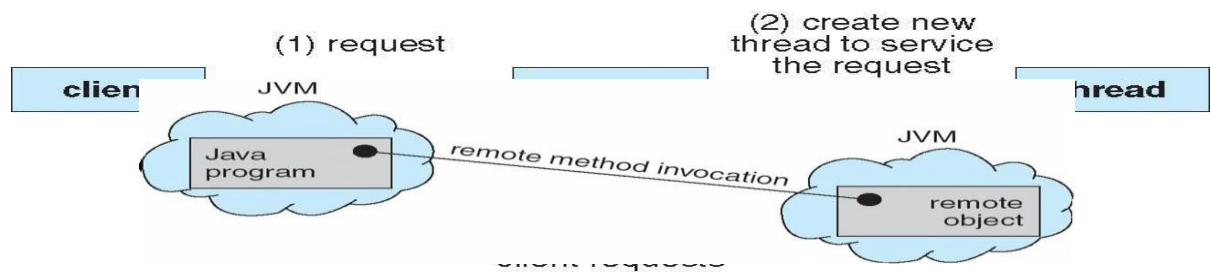
## Execution of RPC

### Remote Method Invocation

- 
- Remote Method Invocation (RMI) is a Java mechanism similar to RPCs

RMI allows a Java program on one machine to invoke a method on a remote object

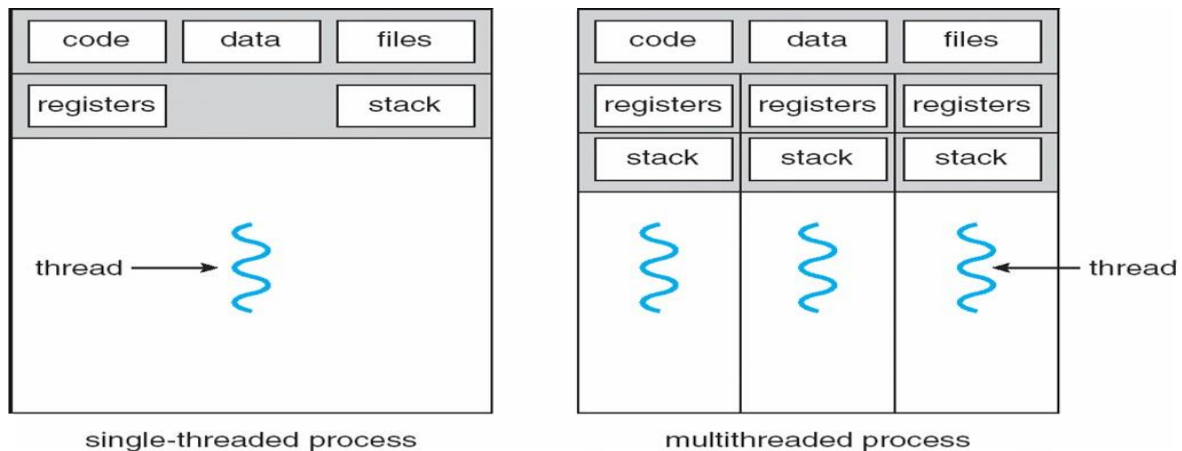




## Marshalling Parameters

### Threads

- To introduce the notion of a thread — a fundamental unit of CPU utilization that forms the basis of multithreaded computer systems
- 
- 



To discuss the APIs for the Pthreads, Win32, and Java thread libraries To examine issues related to multithreaded programmingSingle and Multithreaded Processes

### Benefits

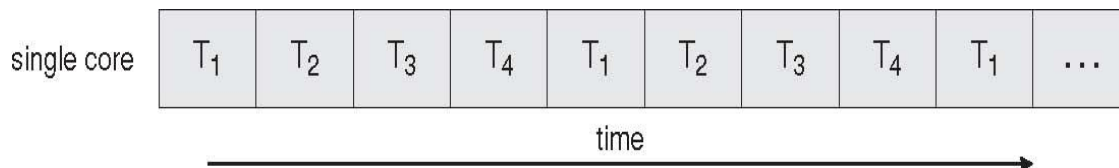
- 
- Responsiveness Resource Sharing Economy Scalability
- 
- **Multicore Programming**

Multicore systems putting pressure on programmers, challenges include

Dividing activities Balance

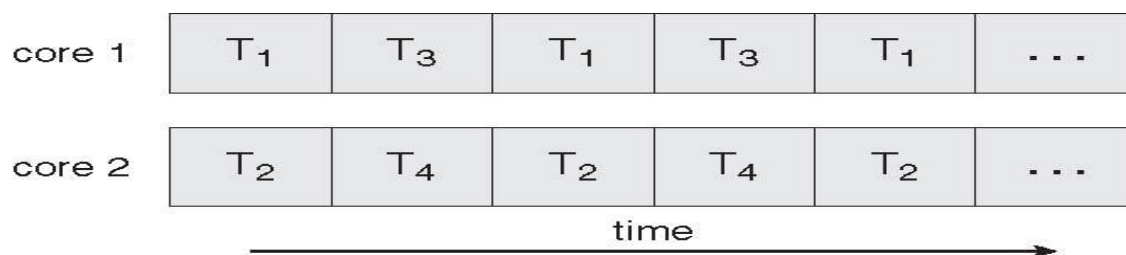
Data splitting Data dependency

Testing and debugging Multithreaded Server Architecture



Concurrent Execution on a Single-core System

Parallel Execution on a Multicore System



User Threads

- 
- Thread management done by user-level threads library
- Three primary thread libraries: POSIX Pthreads Win32 threads

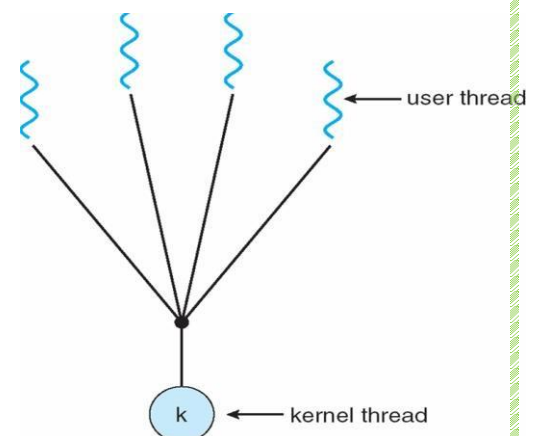
Java threads **Kernel Threads** Supported by the Kernel Examples

Windows XP/2000 Solaris

- Linux
- 
- Tru64 UNIX
- 
- Mac OS X

Multithreading Models

- 
- Many-to-One One-to-One Many-to-Many
- 





## Many-to-One

Many user-level threads mapped to single kernel thread Examples:

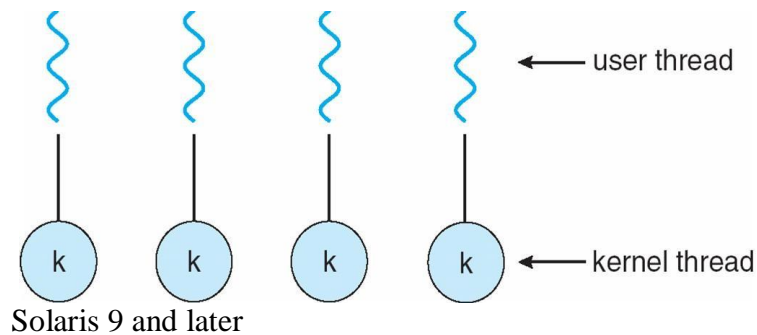
- Solaris Green Threads GNU Portable Threads

- 

## One-to-One

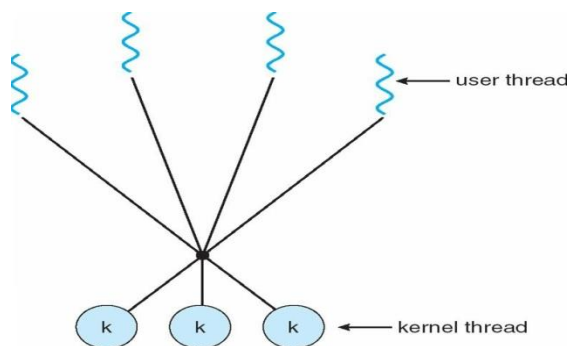
Each user-level thread maps to kernel thread Examples

Windows NT/XP/2000 Linux



## 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 prior to version 9
- 



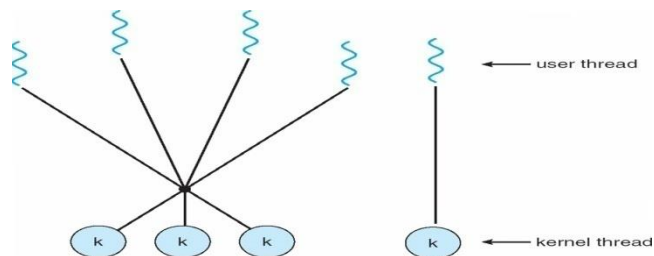
Windows NT/2000 with the *ThreadFiber* package

## Two-level Model

Similar to M:M, except that it allows a user thread to be **bound** to kernel thread Examples

- IRIX HP-UX
- 
- Tru64 UNIX
-

Solaris 8 and earlier



### Thread Libraries

- Thread library provides programmer with API for creating and managing threads Two primary ways of implementing
- Library entirely in user space

Kernel-level library supported by the OS

### Pthreads

- May be provided either as user-level or kernel-level
- A POSIX standard (IEEE 1003.1c) API for thread creation and synchronization

API specifies behavior of the thread library, implementation is up to development of the library  
Common in UNIX operating systems (Solaris, Linux, Mac OS X)

### Java Threads

- Java threads are managed by the JVM
- Typically implemented using the threads model provided by underlying OS Java threads may be created by:
  - Extending Thread class

Implementing the Runnable interface

### Threading Issues

Semantics of **fork()** and **exec()** system calls Thread cancellation of target thread Asynchronous or deferred

Signal handling Thread pools Thread-specific data

Scheduler activations

### Thread Cancellation

- Terminating a thread before it has finished Two general approaches:
- 
-

**Asynchronous cancellation** terminates the target thread immediately

**Deferred cancellation** allows the target thread to periodically check if it should be cancelled

### Signal Handling

- Signals are used in UNIX systems to notify a process that a particular event has occurred A signal handler is used to process signals
- 1. Signal is generated by particular event 2.Signal is delivered to a process 3.Signal is handled
- Options:
  - Deliver the signal to the thread to which the signal applies Deliver the signal to every thread in the process

Deliver the signal to certain threads in the process

Assign a specific thread to receive all signals for the process

### Thread Pools

- Create a number of threads in a pool where they await work Advantages:
- Usually slightly faster to service a request with an existing thread than create a new thread Allows the number of threads in the application(s) to be bound to the size of the pool

### Thread Specific Data

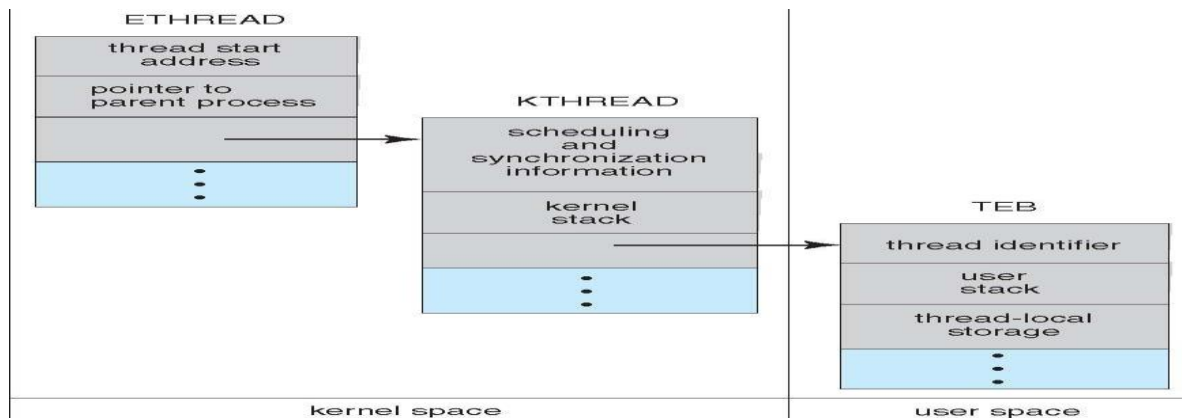
- Allows each thread to have its own copy of data

Useful when you do not have control over the thread creation process (i.e., when using a thread pool)

### Scheduler Activations

- Both M:M and Two-level models require communication to maintain the appropriate number of kernel threads allocated to the application
- Scheduler activations provide upcalls - a communication mechanism from the kernel to the thread library
- This communication allows an application to maintain the correct number kernel threads

## Windows XP Threads



Implements the one-to-one mapping, kernel-level Each thread contains

A thread id Register set

Separate user and kernel stacks Private data storage area

The register set, stacks, and private storage area are known as the context of the threads The primary data structures of a thread include:

ETHREAD (executive thread block) KTHREAD (kernel thread block) TEB (thread environment block)

## Linux Threads

flag	meaning
CLONE_FS	File-system information is shared.
CLONE_VM	The same memory space is shared.
CLONE_SIGHAND	Signal handlers are shared.
CLONE_FILES	The set of open files is shared.

- Linux refers to them as *tasks* rather than *threads*
- 
- Thread creation is done through **clone()** system call

**clone()** allows a child task to share the address space of the parent task (process)

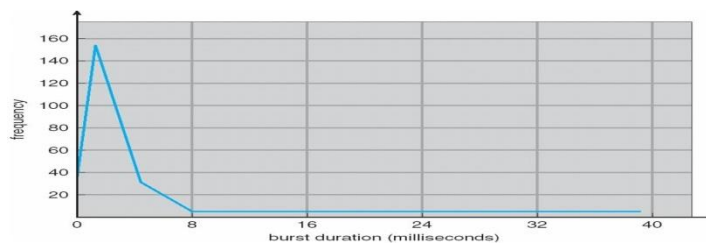
## CPU Scheduling

- To introduce CPU scheduling, which is the basis for multiprogrammed operating systems To
- describe various CPU-scheduling algorithms
- To discuss evaluation criteria for selecting a CPU-scheduling algorithm for a particular system
- Maximum CPU utilization obtained with multiprogramming

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

### CPU burst distribution

#### Histogram of CPU-burst Times



### Alternating Sequence of CPU And I/O Bursts

#### CPU Scheduler

Selects from among the processes in memory that are ready to execute, and allocates the CPU to one



of them CPU scheduling decisions may take place when a process:

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

Scheduling under 1 and 4 is **nonpreemptive** All other scheduling is **preemptive Dispatcher**

- Dispatcher module gives control of the CPU to the process selected by the short-term scheduler; this involves:
- switching context switching to user mode
- jumping to the proper location in the user program to restart that program

**Dispatch latency** – time it takes for the dispatcher to stop one process and start another running

#### Scheduling Criteria

- **CPU utilization** – keep the CPU as busy as possible
- **Throughput** – # of processes that complete their execution per time unit
- **Turnaround time** – amount of time to execute a particular process

**Waiting time** – amount of time a process has been waiting in the ready queue

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

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

#### First-Come, First-Served (FCFS) Scheduling

<u>Process</u>	<u>Burst Time</u>
$P_1$	24
$P_2$	3
$P_3$	3

Suppose that the processes arrive in the order:  $P_1$  ,  $P_2$  ,  $P_3$

The Gantt Chart for the schedule is:

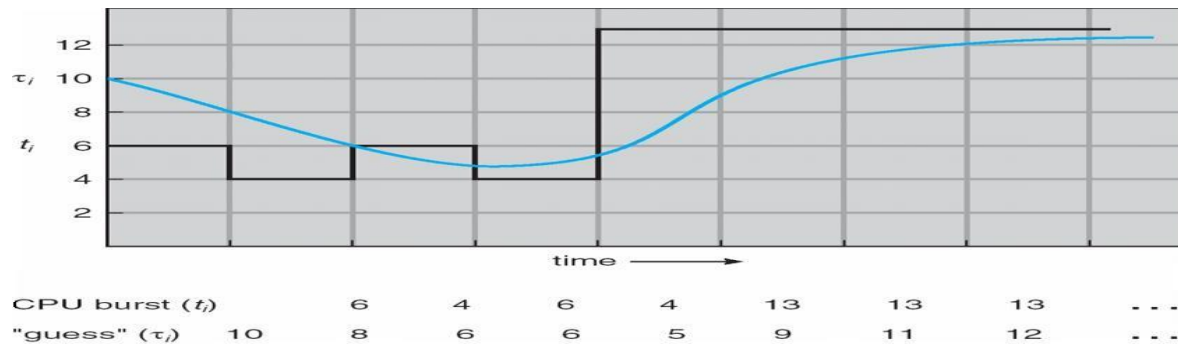
	P <sub>1</sub>	P <sub>2</sub>	P <sub>3</sub>
0	24	27	30

Waiting time for  $P_1 = 0$ ;  $P_2 = 24$ ;  $P_3 = 27$  Average waiting time:  $(0 + 24 + 27)/3 = 17$  Suppose that the processes arrive in the order

$P2, P3, P1$

4. Define :

- Can only estimate the length
- Can be done by using the length of previous CPU bursts, using exponential averaging



Prediction of the Length of the Next CPU Burst

### Examples of Exponential Averaging

$$a = 0 \quad t_{n+1} = t_n$$

Recent history does not count  $a = 1$

$$t_{n+1} = a t_n$$

Only the actual last CPU burst counts If we expand the formula, we get:  $t_{n+1} = a t_n + (1 - a) a t_{n-1} + \dots$

...

$$+ (1 - a) a^j t_{n-j} + \dots$$

$$+ (1 - a)^{n+1} t_0$$

Since both  $a$  and  $(1 - a)$  are less than or equal to 1, each successive term has less weight than its predecessor

### Priority Scheduling

- A priority number (integer) is associated with each process
- 
- 
- 
- 
- 
-

The CPU is allocated to the process with the highest priority (smallest integer ° highest priority)

Preemptive nonpreemptive

SJF is a priority scheduling where priority is the predicted next CPU burst time Problem ° **Starvation**

– low priority processes may never execute

Solution ° **Aging** – as time progresses increase the priority of the process

### Round Robin (RR)

- Each process gets a small unit of CPU time (*time quantum*), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue.

CPU time in chunks of at most  $q$  time units at once. No process waits more than  $(n-1)q$  time units.

Performance

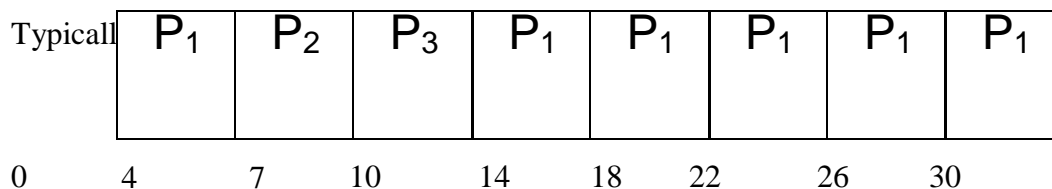
$q$  large P FIFO

$q$  small P  $q$  must be large with respect to context switch, otherwise overhead is too high

### Example of RR with Time Quantum = 4

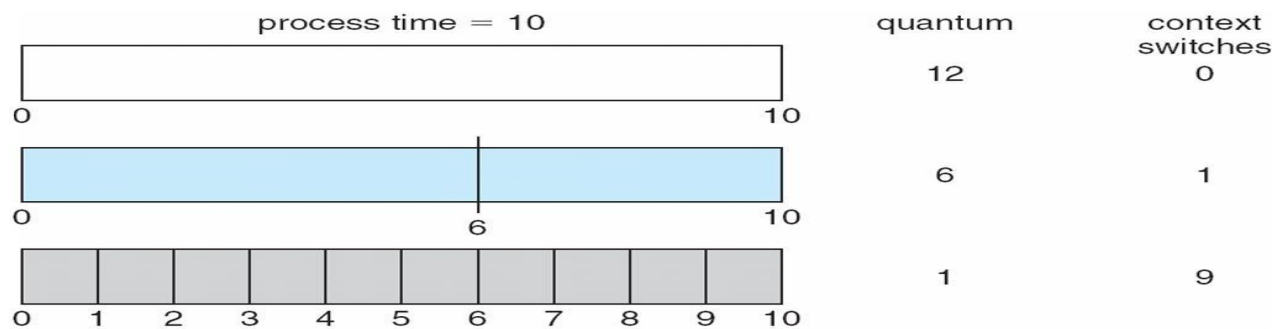
<u>Process</u>	<u>Burst Time</u>
$P_1$	24
$P_2$	3
$P_3$	3

The Gantt chart is:

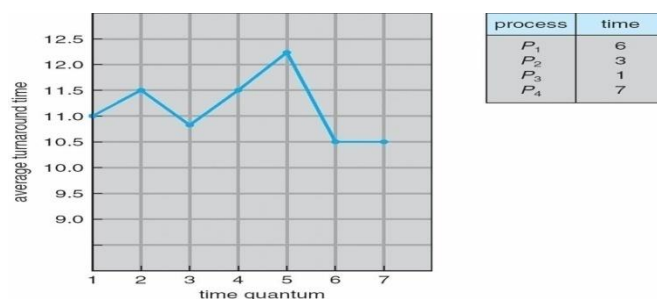




## Time Quantum and Context Switch Time



## Turnaround Time Varies With The Time Quantum



## Multilevel Queue

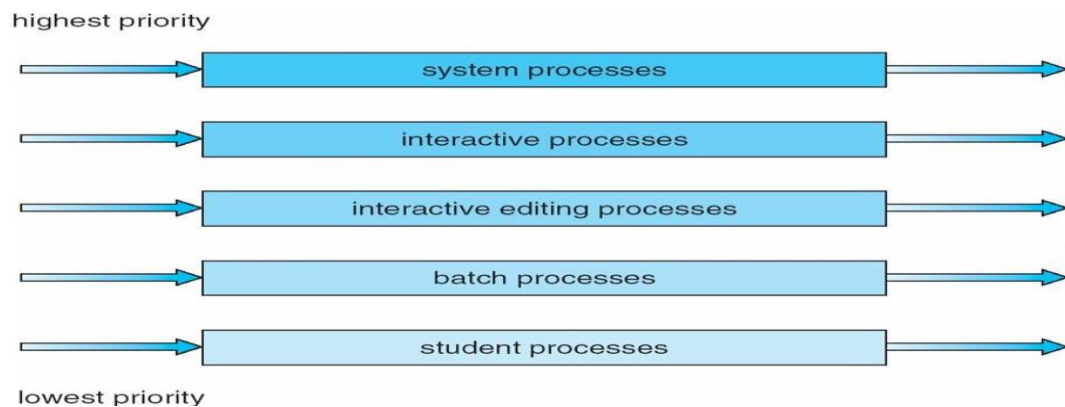
- Ready queue is partitioned into separate queues: foreground (interactive)  
background (batch)
- Each queue has its own scheduling algorithm foreground – RR
- background – FCFS
- Scheduling must be done between the queues

Fixed priority scheduling; (i.e., serve all from foreground then from background). Possibility of starvation.

- Time slice – each queue gets a certain amount of CPU time which it can schedule amongst its processes; i.e., 80% to foreground in RR

20% to background in FCFS

## Multilevel Queue Scheduling



## Multilevel Feedback Queue

- A process can move between the various queues; aging can be implemented this way
- Multilevel-feedback-queue scheduler defined by the following parameters:
- number of queues
- scheduling algorithms for each queue

method used to determine when to upgrade a process    method used to determine when to demote a process

method used to determine which queue a process will enter when that process needs service

## Example of Multilevel Feedback Queue

Three queues:

$Q_0$  – RR with time quantum 8 milliseconds

$Q_1$  – RR time quantum 16 milliseconds

$Q_2$  – FCFS

Scheduling

A new job enters queue  $Q_0$  which is served FCFS. When it gains CPU, job receives 8 milliseconds. If it does not finish in 8 milliseconds, job is moved to queue  $Q_1$ .

- At  $Q_1$  job is again served FCFS and receives 16 additional milliseconds. If it still does not complete, it is preempted and moved to queue  $Q_2$ .



Multilevel Feedback Queues

## Thread Scheduling

- Distinction between user-level and kernel-level threads
- Many-to-one and many-to-many models, thread library schedules user-level threads to run on LWP
- Known as **process-contention scope (PCS)** since scheduling competition is within the process  
Kernel thread scheduled onto available CPU is **system-contention scope (SCS)** – competition among all threads in system

### Pthread Scheduling

- API allows specifying either PCS or SCS during thread creation PTHREAD\_SCOPE\_PROCESS
- schedules threads using PCS scheduling PTHREAD\_SCOPE\_SYSTEM schedules threads using SCS scheduling.

**Pthread Scheduling API** #include <pthread.h> #include <stdio.h>

```
#define NUM_THREADS 5 int main(int argc, char *argv[])
```

```
{
```

```
int i; pthread_t tid[NUM_THREADS]; pthread_attr_t attr;
```

```
/* get the default attributes */ pthread_attr_t init(&attr);
```

```
/* set the scheduling algorithm to PROCESS or SYSTEM */ pthread_attr_t setscope(&attr, PTHREAD_SCOPE_SYSTEM);
```

```

/* set the scheduling policy - FIFO, RT, or OTHER */ pthread_attr_t attr; attr.schedpolicy = SCHED
OTHER);

/* create the threads */

for (i = 0; i < NUM_THREADS; i++)

pthread_create(&tid[i], &attr, runner, NULL);

/* now join on each thread */

for (i = 0; i < NUM_THREADS; i++)

pthread_join(tid[i], NULL);

}

/* Each thread will begin control in this function */ void *runner(void *param)

{

printf("I am a thread\n"); pthread_exit(0);

}

```

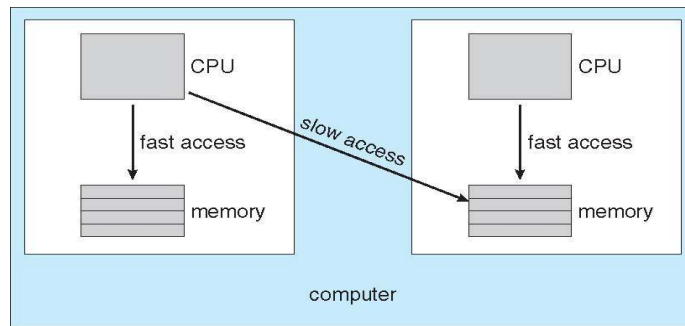
### Multiple-Processor Scheduling

- CPU scheduling more complex when multiple CPUs are available
- **Homogeneous processors** within a multiprocessor

**Asymmetric multiprocessing** – only one processor accesses the system data structures, alleviating the need for data sharing

- **Symmetric multiprocessing (SMP)** – each processor is self-scheduling, all processes in common ready queue, or each has its own private queue of ready processes
- **Processor affinity** – process has affinity for processor on which it is currently running

soft affinity hard affinity

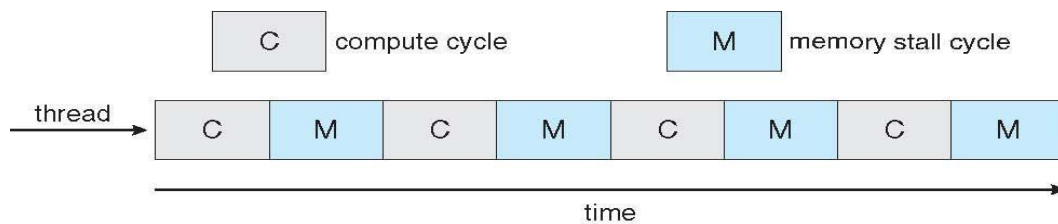


## NUMA and CPU Scheduling

### Multicore Processors

- Recent trend to place multiple processor cores on same physical chip Faster and consume less power
- Multiple threads per core also growing

Takes advantage of memory stall to make progress on another thread while memory retrieve happens



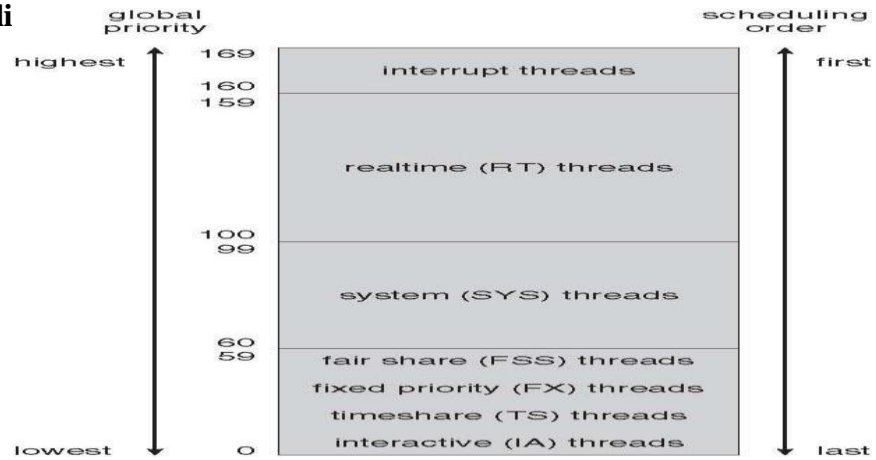
### Multithreaded Multicore System

#### Operating System Examples Solaris scheduling Windows XP scheduling Linux scheduling

- Solaris Dispatch Table

priority	time quantum	time quantum expired	return from sleep
0	200	0	50
5	200	0	50
10	160	0	51
15	160	5	51
20	120	10	52
25	120	15	52
30	80	20	53
35	80	25	54
40	40	30	55
45	40	35	56
50	40	40	58
55	40	45	58
59	20	49	59

## Solaris Scheduling



	real-time	high	above normal	normal	below normal	idle priority
time-critical	31	15	15	15	15	15
highest	26	15	12	10	8	6
above normal	25	14	11	9	7	5
normal	24	13	10	8	6	4
below normal	23	12	9	7	5	3
lowest	22	11	8	6	4	2
idle	16	1	1	1	1	1

## Windows XP Priorities

## Linux Scheduling

- Constant order  $O(1)$  scheduling time
- 
- Two priority ranges: time-sharing and real-time

**Real-time** range from 0 to 99 and **nice** value from 100 to 140

numeric priority	relative priority		time quantum
0	highest	real-time tasks	200 ms
•			
•			
99			
100		other tasks	10 ms
•			
•			
140	lowest		

Priorities and Time-slice length

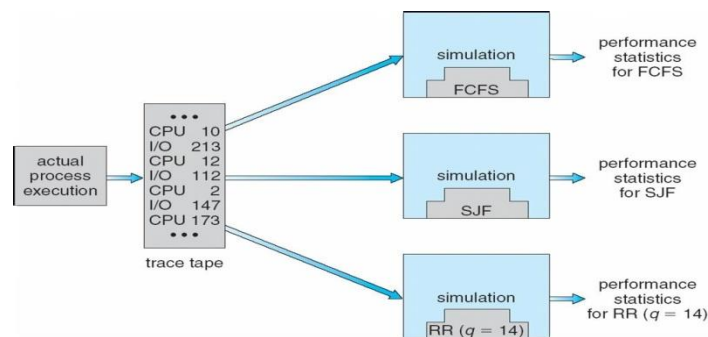
### List of Tasks Indexed According to Priorities



### Algorithm Evaluation

- Deterministic modeling – takes a particular predetermined workload and defines the performance of each algorithm for that workload
- Queuing models Implementation

## Evaluation of CPU schedulers by Simulation



## UNIT-3 CONCURRENCY

### Process Synchronization

- To introduce the critical-section problem, whose solutions can be used to ensure the consistency of shared data
  - To present both software and hardware solutions of the critical-section problem
  - To introduce the concept of an atomic transaction and describe mechanisms to ensure atomicity
- Concurrent access to shared data may result in data inconsistency

Maintaining data consistency requires mechanisms to ensure the orderly execution of cooperating processes

- Suppose that we wanted to provide a solution to the consumer-producer problem that fills all the buffers. We can do so by having an integer count that keeps track of the number of full buffers. Initially, count is set to 0. It is incremented by the producer after it produces a new buffer and is decremented by the consumer after it consumes a buffer

#### Producer

```
while (true) {  
  
    /* produce an item and put in nextProduced */ while (count == BUFFER_SIZE)  
  
    ; // do nothing  
  
    buffer [in] = nextProduced;  
  
    in = (in + 1) % BUFFER_SIZE;  
  
    count++;  
}
```



```
}
```

### Consumer

```
while (true) {  
    while (count == 0)  
        ; // do nothing  
    nextConsumed = buffer[out];  
    out = (out + 1) % BUFFER_SIZE;  
    count--;  
    /* consume the item in nextConsumed  
}
```

### Race Condition

count++ could be implemented as

register1 = count register1 = register1 + 1 count = register1

count-- could be implemented as

register2 = count register2 = register2 - 1 count = register2

Consider this execution interleaving with “count = 5” initially:

S0: producer execute register1 = count {register1 = 5} S1: producer execute register1 = register1 + 1  
{register1 = 6} S2: consumer execute register2 = count {register2 = 5}

S3: consumer execute register2 = register2 - 1 {register2 = 4} S4: producer execute count = register1  
{count = 6 }

S5: consumer execute count = register2 {count = 4}

### Solution to Critical-Section Problem

1. Mutual Exclusion - If process  $P_i$  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 Assume that each process executes at a nonzero speed

No assumption concerning relative speed of the N processes

#### Peterson's Solution

- Two process solution
- Assume that the LOAD and STORE instructions are atomic; that is, cannot be interrupted. The two
- processes share two variables:
- `int turn; Boolean flag[2]`

The variable turn indicates whose turn it is to enter the critical section.

The flag array is used to indicate if a process is ready to enter the critical section. `flag[i] = true` implies that process  $P_i$  is ready!

#### Algorithm for Process $P_i$

```
do {  
    flag[i] = TRUE; turn = j;  
    while (flag[j] && turn == j); critical section  
    flag[i] = FALSE;  
    remainder section  
} while (TRUE);
```

#### Synchronization Hardware

- Many systems provide hardware support for critical section code Uniprocessors – could disable
- interrupts
- Currently running code would execute without preemption Generally too inefficient on multiprocessor systems

Operating systems using this not broadly scalable

- Modern machines provide special atomic hardware instructions Atomic = non-interruptable
-

Either test memory word and set value Or swap contents of two memory words

### Solution to Critical-section Problem Using Locks

```
do {  
  
    acquire lock  
  
    critical section  
    release lock  
  
    remainder section  
  
} while (TRUE);
```

### **TestAndSet Instruction Definition:**

```
boolean TestAndSet (boolean *target)  
{  
    boolean rv = *target;  
    *target = TRUE; return rv;  
}
```

### Solution using TestAndSet

Shared boolean variable lock., initialized to false. Solution:

```
do {  
  
    while ( TestAndSet (&lock ))  
        ; // do nothing  
  
    //                critical section  
    lock = FALSE;  
  
    //                remainder section  
  
} while (TRUE);
```

### Swap Instruction

Definition:

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

```
*a = *b;  
*b = temp;  
}
```

### Solution using Swap

Shared Boolean variable lock initialized to FALSE; Each process has a local Boolean variable key  
Solution:

```
do {  
    key = TRUE;  
    while ( key == TRUE) Swap (&lock, &key );  
    //critical section lock = FALSE;  
    //remainder section  
} while (TRUE);
```

### Bounded-waiting Mutual Exclusion with TestAndSet()

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

## Semaphore

- Synchronization tool that does not require busy waiting
- Semaphore  $S$  – integer variable
- Two standard operations modify  $S$ : wait() and signal()
- Originally called P() and V() Less complicated

Can only be accessed via two indivisible (atomic) operations

wait ( $S$ ) {

while  $S \leq 0$

; // no-op

S--;

}

signal ( $S$ ) {

S++;

}

## Semaphore as General Synchronization Tool

- Counting semaphore – integer value can range over an unrestricted domain
- Binary semaphore – integer value can range only between 0

and 1; can be simpler to implement

- Also known as mutex lock
- Can implement a counting semaphore  $S$  as a binary semaphore
- Provides mutual exclusion

Semaphore mutex; // initialized to 1

wait (mutex);

// Critical Section

signal (mutex);

// remainder section

} while (TRUE);

## Semaphore Implementation

- Must guarantee that no two processes can execute wait () and signal () on the same semaphore at the same time

- Thus, implementation becomes the critical section problem where the wait and signal code are placed in the critical section.
- Could now have busy waiting in critical section implementation But implementation code is short

Little busy waiting if critical section rarely occupied

- Note that applications may spend lots of time in critical sections and therefore this is not a good solution.

### Semaphore Implementation with no Busy waiting

- With each semaphore there is an associated waiting queue. Each entry in a waiting queue has two data items:
  - value (of type integer)
  - pointer to next record in the list
- Two operations:
  - block – place the process invoking the operation on the appropriate waiting queue.
  - wakeup – remove one of processes in the waiting queue and place it in the ready queue.

Implementation of wait:

```
wait(semaphore *S) {
    S->value--;
    if (S->value < 0) {
        add this process to S->list; block();
    }
}
```

Implementation of signal:

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

```
}
}
```

## Deadlock and Starvation

- Deadlock – two or more processes are waiting indefinitely for an event that can be caused by only one of the waiting processes

Let S and Q be two semaphores initialized to 1

<i>P0</i> wait (S); wait (Q); .	<i>P1</i> wait (Q); wait (S); .	
.	.	.
signal (S);	signal (Q);	.
signal (Q);	signal (S);	

- Starvation – indefinite blocking. A process may never be removed from the semaphore queue in which it is suspended
- Priority Inversion - Scheduling problem when lower-priority process holds a lock needed by higher- priority process

## Classical Problems of Synchronization Bounded-Buffer Problem Readers and Writers Problem

- Dining-
- Philosophers Problem

## Bounded-Buffer Problem

- *N* buffers, each can hold one item Semaphore mutex initialized to the value 1 Semaphore full initialized to the value 0 Semaphore empty initialized to the value *N*.
- The structure of the producer process

```
do {
    // produce an item in nextp wait (empty);
    wait (mutex);
```

```

// add the item to the buffer signal (mutex);

signal (full);

} while (TRUE);

The structure of the consumer process do { wait (full);

wait (mutex);

// remove an item from buffer to nextc signal (mutex);

signal (empty);

// consume the item in nextc

} while (TRUE);

```

### Readers-Writers Problem

A data set is shared among a number of concurrent processes

- Readers – only read the data set; they do **not** perform any updates

Writers – can both read and write  
 Problem – allow multiple readers to read at the same time. Only one single writer can access the shared data at the same time

Shared Data

- Data set
- Semaphore mutex initialized to 1 Semaphore wrt initialized to 1 Integer readcount initialized to 0
- The structure of a writer process

```

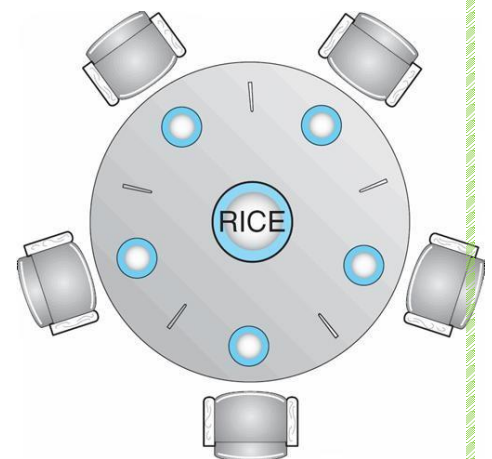
do { wait (wrt) ;

// writing is performed signal (wrt) ;

} while (TRUE);

```

The structure of a reader process do {





```

wait (mutex) ; readcount ++ ;

if (readcount == 1)

wait (wrt) ; signal (mutex)

// reading is performed wait (mutex) ;

readcount - - ;

if (readcount == 0)

signal (wrt) ; signal (mutex) ;

} while (TRUE);

```

### Dining-Philosophers Problem

- 
- 
- 
- 

```

do {

Shared data

Bowl of rice (data set)

Semaphore chopstick [5] initialized to 1 The structure of Philosopher i:

wait ( chopstick[i] );

wait ( chopStick[ (i + 1) % 5] );

// eat

signal ( chopstick[i] );

signal ( chopstick[ (i + 1) % 5] );

// think

} while (TRUE);

```

### Problems with Semaphores

Incorrect use of semaphore operations:

l signal (mutex)

....

wait (mutex)

wait (mutex) ... wait (mutex)

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

## 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 P1 (...) { .... }

...

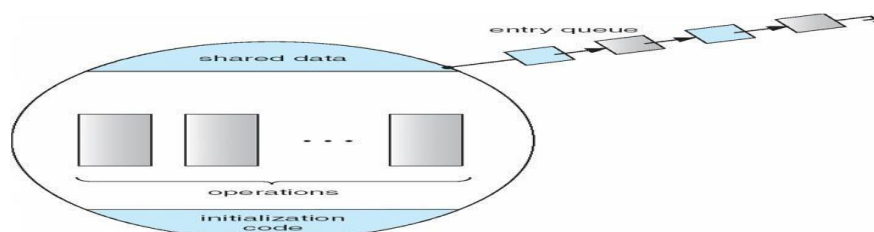
procedure Pn (...) {.....} Initialization code ( ....) { ... }

...

}

}

## Schematic view of a Monitor



## Condition Variables

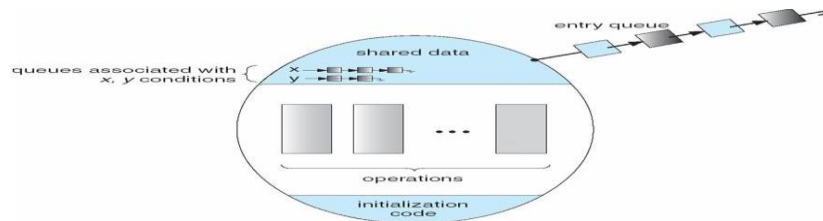
condition x, y;

Two operations on a condition variable:

x.wait () – a process that invokes the operation is suspended.

x.signal () – resumes one of processes (if any) that invoked x.wait ()

## Monitor with Condition Variables



## Solution to Dining Philosophers

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;

}

}

```

Each philosopher  $I$  invokes the operations pickup() and putdown() in the following sequence:

DiningPhilosophers.pickup (i); EAT

DiningPhilosophers.putdown (i);

### Monitor Implementation Using Semaphores

#### Variables

semaphore mutex; // (initially = 1) semaphore next; // (initially = 0)

int next-count = 0; Each procedure  $F$  will be replaced by wait(mutex);

... body of  $F$ ;

...

if (next\_count > 0) signal(next)

else

signal(mutex); Mutual exclusion within a monitor is ensured

### Synchronization Examples

- Solaris Windows XP Linux Pthreads
- Solaris Synchronization
  - **Implements a variety of locks to support** multitasking, multithreading (including real-time threads), and multiprocessing
  - Uses adaptive mutexes for efficiency when protecting data from short code segments

Uses condition variables and readers-writers locks when longer sections of code need access to data  
Uses turnstiles to order the list of threads waiting to acquire either an adaptive mutex or reader-writer lock

### Windows XP Synchronization

- Uses interrupt masks to protect access to global resources on uniprocessor systems Uses spinlocks on
- multiprocessor systems
- Also provides dispatcher objects which may act as either mutexes and semaphores Dispatcher objects may also provide events

An event acts much like a condition variable

### Linux Synchronization

Linux: IPrior to kernel Version 2.6, disables interrupts to implement short critical sections Version 2.6 and later, fully preemptive

Linux provides:

semaphores spin locks

### P threads Synchronization

- Pthreads API is OS-independent It provides:
- mutex locks
- condition variablesn Non-portable extensions include: read-write locks
- spin locks

### Atomic Transactions

- System Model
- Log-based Recovery Checkpoints
- Concurrent Atomic Transactions

### System Model

Assures that operations happen as a single logical unit of work, in its entirety, or not at all Related to field of database systems

Challenge is assuring atomicity despite computer system failures

Transaction - collection of instructions or operations that performs single logical function Here we are concerned with changes to stable storage – disk

Transaction is series of read and write operations

Terminated by commit (transaction successful) or abort (transaction failed) operation Aborted transaction must be rolled back to undo any changes it performed

### Types of Storage Media

Volatile storage – information stored here does not survive system crashes Example: main memory, cache

Nonvolatile storage – Information usually survives crashes Example: disk and tape

Stable storage – Information never lost

Not actually possible, so approximated via replication or RAID to devices with independent failure modes

- Goal is to assure transaction atomicity where failures cause loss of information on volatile storage

### Log-Based Recovery

- Record to stable storage information about all modifications by a transaction Most common is write-ahead logging

Log on stable storage, each log record describes single transaction write operation, including Transaction name

Data item name Old value

New value

- $\langle T_i \text{ starts} \rangle$  written to log when transaction  $T_i$  starts
- $\langle T_i \text{ commits} \rangle$  written when  $T_i$  commits

Log entry must reach stable storage before operation on data occurs

## Log-Based Recovery Algorithm

**Using the log, system can handle any volatile memory errors** Undo( $T_i$ ) restores value of all data

- updated by  $T_i$
- 
- Redo( $T_i$ ) sets values of all data in transaction  $T_i$  to new values Undo( $T_i$ ) and redo( $T_i$ ) must be
- idempotent

Multiple executions must have the same result as one execution

- If system fails, restore state of all updated data via log
- If log contains  $\langle T_i \text{ starts} \rangle$  without  $\langle T_i \text{ commits} \rangle$ , undo( $T_i$ ) If log contains  $\langle T_i \text{ starts} \rangle$  and  $\langle T_i$
- $\text{commits} \rangle$ , redo( $T_i$ )

## Checkpoints

Log could become long, and recovery could take long Checkpoints shorten log and recovery time.

Checkpoint scheme:

1. Output all log records currently in volatile storage to stable storage 2. Output all modified data from volatile to stable storage

3. Output a log record  $\langle \text{checkpoint} \rangle$  to the log on stable storage

Now recovery only includes  $T_i$ , such that  $T_i$  started executing before the most recent checkpoint, and all transactions after  $T_i$  All other transactions already on stable storage

## Concurrent Transactions

- 
- Must be equivalent to serial execution – serializability Could perform all transactions in critical
- section Inefficient, too restrictive
- 
- Concurrency-control algorithms provide serializability

## Serializability

- Consider two data items A and B Consider Transactions  $T_0$  and  $T_1$  Execute  $T_0, T_1$  atomically
- Execution sequence called schedule
- 
- Atomically executed transaction order called serial schedule For N transactions, there are  $N!$  valid
- serial schedules
-

Schedule 1: T0 then T1

Schedule 2: Concurrent Serializable Schedule

$T_0$	$T_1$
read(A)	read(A)
write(A)	write(A)
read(B)	read(B)
write(B)	write(B)

### Locking Protocol

- Ensure serializability by associating lock with each data item Follow locking protocol for access
- control
- 
- Locks
- 
- Shared –  $T_i$  has shared-mode lock (S) on item Q,  $T_i$  can read Q but not write Q Exclusive –  $T_i$  has
- exclusive-mode lock (X) on Q,  $T_i$  can read and write Q Require every transaction on item Q acquire
- appropriate lock

If lock already held, new request may have to wait Similar to readers-writers algorithm

### Two-phase Locking Protocol

- 
- Generally ensures conflict serializability
- 
- Each transaction issues lock and unlock requests in two phases Growing – obtaining locks
- Shrinking – releasing locks Does not prevent deadlock

### Timestamp-based Protocols

- 
- Select order among transactions in advance – timestamp-ordering Transaction  $T_i$  associated with
- timestamp  $TS(T_i)$  before  $T_i$  starts  $TS(T_i) < TS(T_j)$  if  $T_i$  entered system before  $T_j$
- 
- TS can be generated from system clock or as logical counter incremented at each entry of transaction
- Timestamps determine serializability order



If  $TS(T_i) < TS(T_j)$ , system must ensure produced schedule equivalent to serial schedule where  $T_i$  appears before  $T_j$

### Timestamp-based Protocol Implementation

- Data item  $Q$  gets two timestamps
  - 
  - $W\text{-timestamp}(Q)$  – largest timestamp of any transaction that executed  $\text{write}(Q)$  successfully
  - $R\text{-timestamp}(Q)$  – largest timestamp of successful  $\text{read}(Q)$
  - Updated whenever  $\text{read}(Q)$  or  $\text{write}(Q)$  executed
  - Timestamp-ordering protocol assures any conflicting read and write executed in timestamp order
- Suppose  $T_i$  executes  $\text{read}(Q)$

If  $TS(T_i) < W\text{-timestamp}(Q)$ ,  $T_i$  needs to read value of  $Q$  that was already overwritten read operation rejected and  $T_i$  rolled back

If  $TS(T_i) \geq W\text{-timestamp}(Q)$  read executed,  $R\text{-timestamp}(Q)$  set to  $\max(R\text{-timestamp}(Q), TS(T_i))$

### Timestamp-ordering Protocol

Suppose  $T_i$  executes  $\text{write}(Q)$

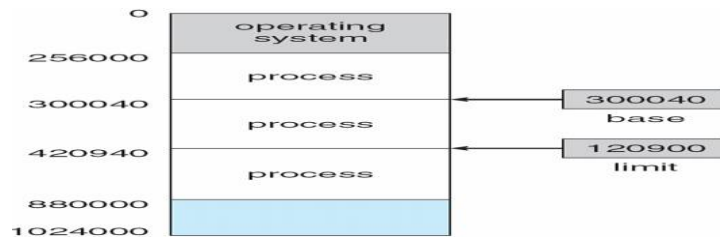
If  $TS(T_i) < R\text{-timestamp}(Q)$ , value  $Q$  produced by  $T_i$  was needed previously and  $T_i$  assumed it would never be produced Write operation rejected,  $T_i$  rolled back If  $TS(T_i) < W\text{-timestamp}(Q)$ ,  $T_i$  attempting to write obsolete value of  $Q$  Write operation rejected and  $T_i$  rolled back Otherwise, write executed Any rolled back transaction  $T_i$  is assigned new timestamp and restarted Algorithm ensures conflict serializability and freedom from deadlock **Schedule Possible Under Timestamp Protocol**

$T_2$	$T_3$
$\text{read}(B)$	$\text{read}(B)$
	$\text{write}(B)$
$\text{read}(A)$	$\text{read}(A)$
	$\text{write}(A)$

## UNIT IV

### Memory Management

- To provide a detailed description of various ways of organizing memory hardware
- 
- 



To discuss various memory-management techniques, including paging and segmentation

To provide a detailed description of the Intel Pentium, which supports both pure segmentation and segmentation with paging

Program must be brought (from disk) into memory and placed within a process for it to be run Main memory and registers are only storage CPU can access directly

Register access in one CPU clock (or less) Main memory can take many cycles

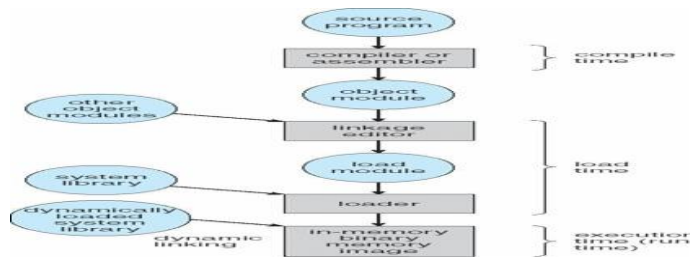
**Cache** sits between main memory and CPU registers Protection of memory required to ensure correct operation

## Base and Limit Registers

A pair of **base** and **limit** registers define the logical address space

Binding of Instructions and Data to Memory

- Address binding of instructions and data to memory addresses can happen at three different stages
- **Compile time:** If memory location known a priori, **absolute code** can be generated; must recompile code if starting location changes
- **Load time:** Must generate **relocatable code** if memory location is not known at compile time
- Execution time:** Binding delayed until run time if the process can be moved during its execution from one memory segment to another. Need hardware support for address maps (e.g., base and limit registers)



## Multistep Processing of a User Program

### Logical vs. Physical Address Space

- The concept of a logical address space that is bound to a separate **physical address space** is central to proper memory management
- **Logical address** – generated by the CPU; also referred to as **virtual address**
- **Physical address** – address seen by the memory unit

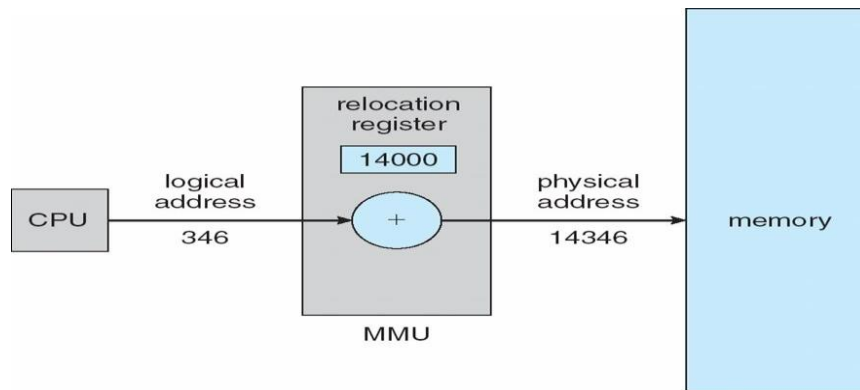
Logical and physical addresses are the same in compile-time and load-time address-binding schemes; logical (virtual) and physical addresses differ in execution-time address-binding scheme

### Memory-Management Unit (MMU)

- 
- Hardware device that maps virtual to physical address

In MMU scheme, the value in the relocation register is added to every address generated by a user process at the time it is sent to memory

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



### Dynamic relocation using a relocation register

#### Dynamic Loading

- Routine is not loaded until it is called
- Better memory-space utilization; unused routine is never loaded
- Useful when large amounts of code are needed to handle infrequently occurring cases

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

#### Dynamic Linking

- Linking postponed until execution time
- Small piece of code, *stub*, used to locate the appropriate memory-resident library routine 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 Dynamic linking is particularly useful for libraries

System also known as **shared libraries**

#### Swapping

A process can be swapped temporarily out of memory to a backing store, and then brought back into memory for continued execution.

**Backing store** – fast disk large enough to accommodate copies of all memory images for all users; must provide direct access to these memory images.

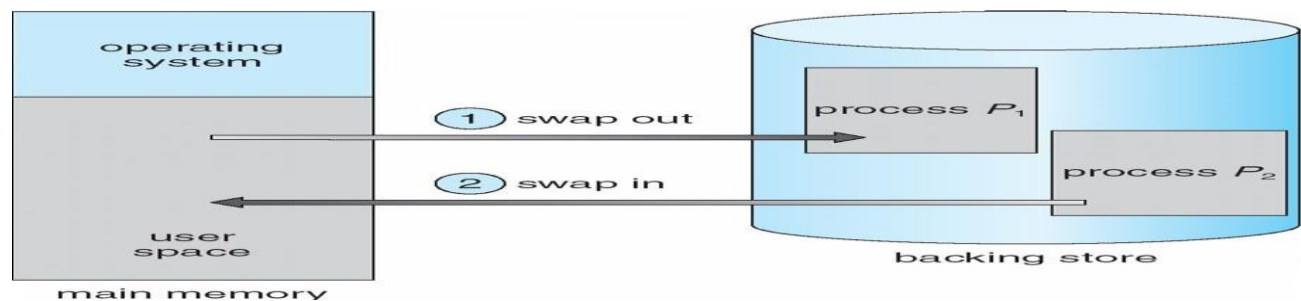
**Roll out, roll in** – swapping variant used for priority-based scheduling algorithms; lower-priority process is swapped out so higher-priority process can be loaded and executed.

Major part of swap time is transfer time; total transfer time is directly proportional to the amount of memory swapped.

Modified versions of swapping are found on many systems (i.e., UNIX, Linux, and Windows).

System maintains a **ready queue** of ready-to-run processes which have memory images on disk.

### Schematic View of Swapping



- Main memory usually into two partitions:
- Resident operating system, usually held in low memory with interrupt vector

User processes then held in high memory.

Relocation registers used to protect user processes from

### Contiguous Allocation

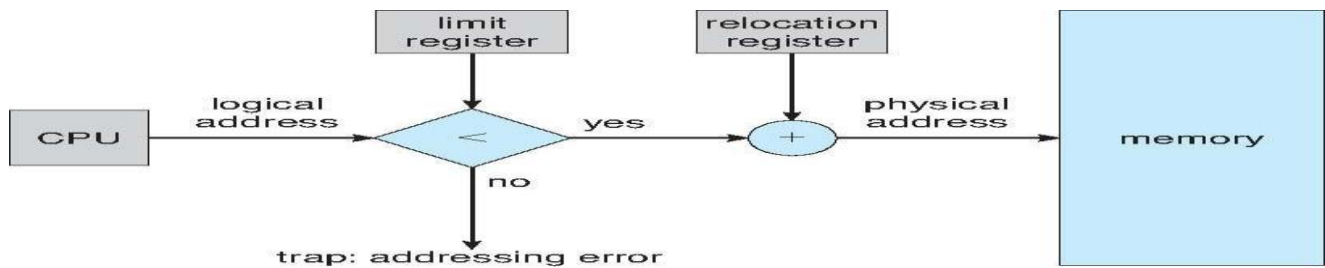
- each user,
- 

and from changing operating-system code and data

Base register contains value of smallest physical address

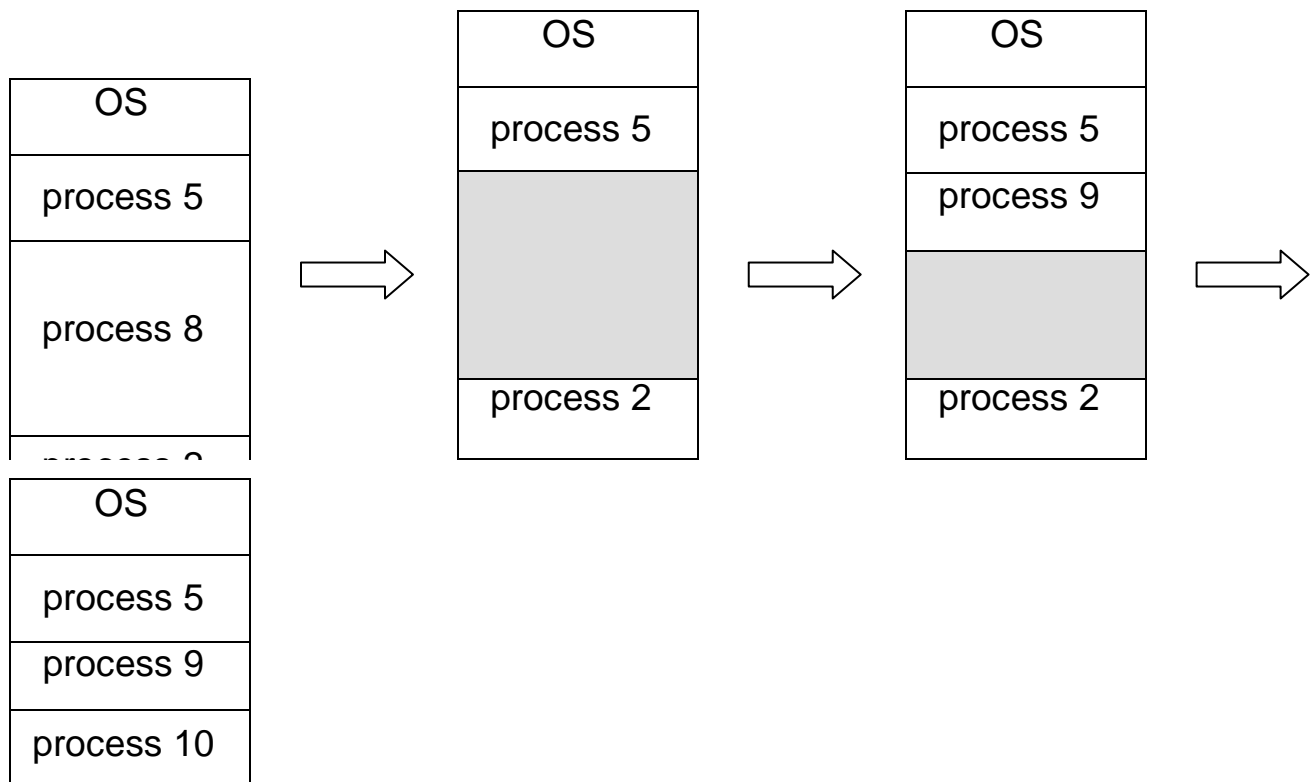
Limit register contains range of logical addresses – each logical address must be less than the limit register

- MMU maps logical address *dynamically*



### Hardware Support for Relocation and Limit Registers

- Operating system maintains information about:
  - a) allocated partitions
  - b) free partitions (hole)



### 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 better than worst-fit in terms of speed and storage utilization

## • **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

Latch job in memory while it is involved in I/O

Do I/O only into OS buffers

## **Paging**

- 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 8,192 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

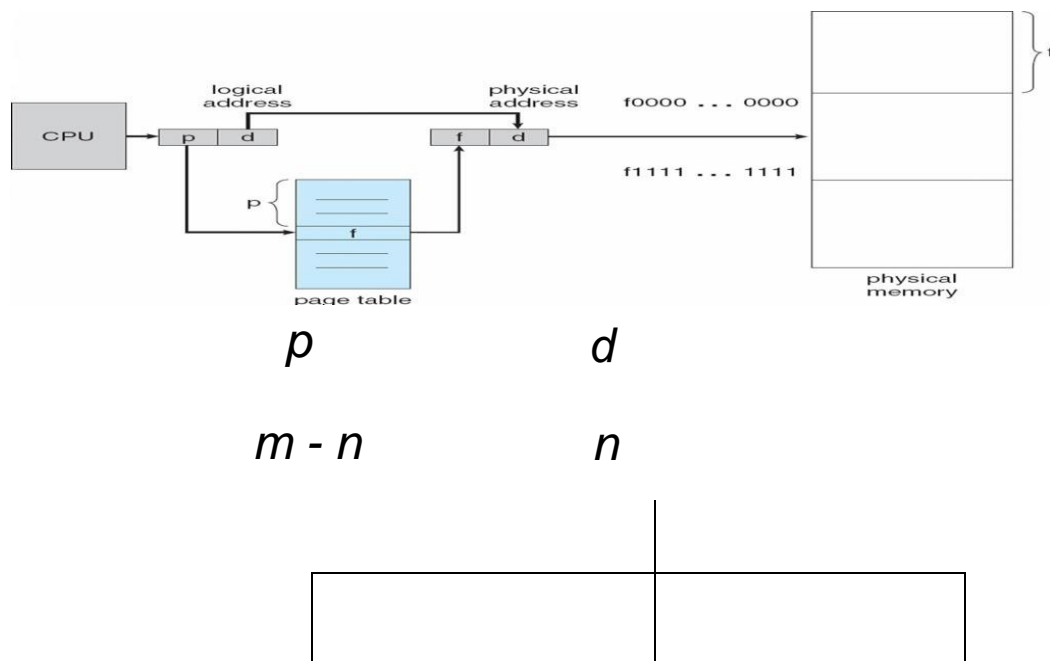
- translate logical to physical addresses

- Internal fragmentation

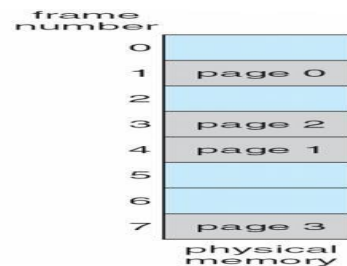
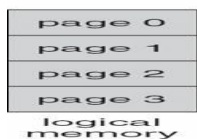
## **Address Translation Scheme**

- 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
- For given logical address space  $2^m$  and page size  $2^n$

### Paging Hardware



### Paging Model of Logical and Physical Memory



### Paging Example





## 32-byte memory and 4-byte pages



## Free Frames

## Implementation of Page Table

- Page table is kept in main memory
- **Page-table base register (PTBR)** points to the page table
- **Page-table length register (PRLR)** indicates size of the page table

In this scheme every data/instruction access requires two memory accesses. One for the page table and one for the data/instruction.

- The two memory access problem can be solved by the use of a special fast-lookup hardware cache called **associative memory** or **translation look-aside buffers (TLBs)**
- Some TLBs store **address-space identifiers (ASIDs)** in each TLB entry – uniquely identifies each process to provide address-space protection for that process

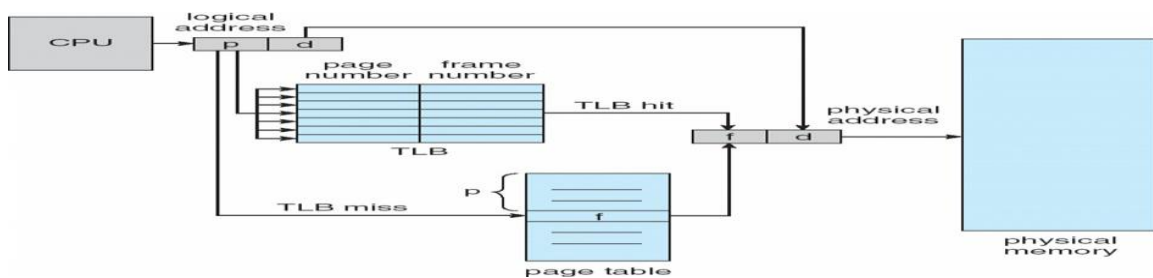
## Associative Memory

Associative memory – parallel search Address translation (p, d)

If p is in associative register, get frame # out Otherwise get frame # from page table in memory

Page #	Frame #

### Paging Hardware With TLB



### Effective Access Time

- Associative Lookup = e time unit
- Assume memory cycle time is 1 microsecond

Hit ratio – percentage of times that a page number is found in the associative registers; ratio related to number of associative registers

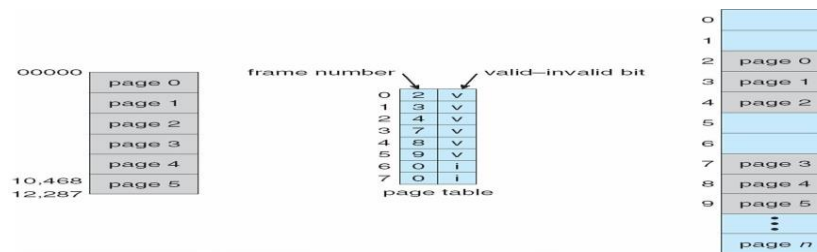
- Hit ratio = an **Effective Access Time** (EAT)

$$\text{EAT} = (1 + e) a + (2 + e)(1 - a)$$

$$= 2 + e - a$$

## Memory Protection

- - Memory protection implemented by associating protection bit with each frame
  - 
  - **Valid-invalid** bit attached to each entry in the page table:
  -
- “valid” indicates that the associated page is in the process’ logical address space, and is thus a legal page “invalid” indicates that the page is not in the process’ logical address space



## Valid (v) or Invalid (i) Bit In A Page Table

### Shared Pages 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 processes

### Private code and data

- 
- Each process keeps a separate copy of the code and data

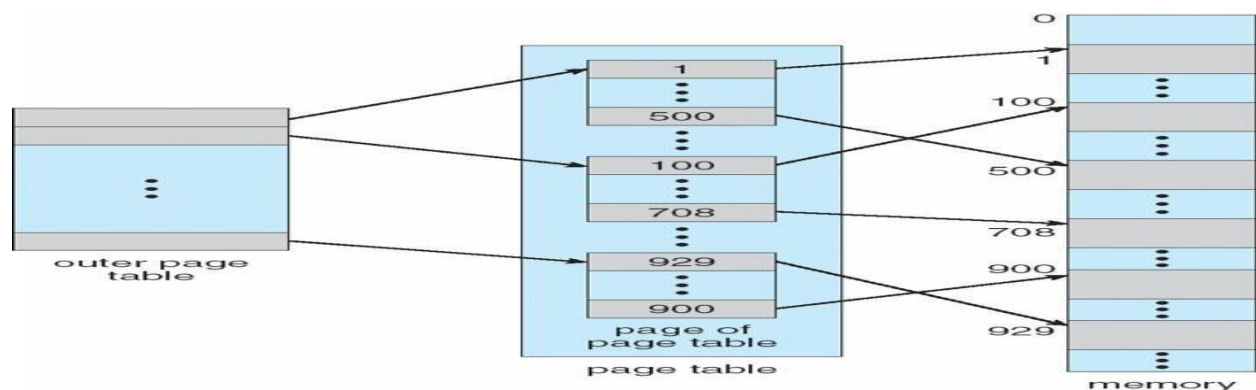
The pages for the private code and data can appear anywhere in the logical address space



## Shared Pages Example

## Structure of the Page Table

- Hierarchical Paging Hashed Page Tables Inverted Page Tables
- 
- **Hierarchical Page Tables**
- Break up the logical address space into multiple page tables A simple technique is a two-level page
- table



## Two-Level Page-Table Scheme

### Two-Level Paging Example

- A logical address (on 32-bit machine with 1K page size) is divided into: a page number consisting of
- 22 bits
- a page offset consisting of 10 bits
- Since the page table is paged, the page number is further divided into:
- a 12-bit page number a 10-bit page offset

Thus, a logical address is as follows:

where  $p_i$  is an index into the outer page table, and  $p_2$  is the displacement within the page of the outer page table

p

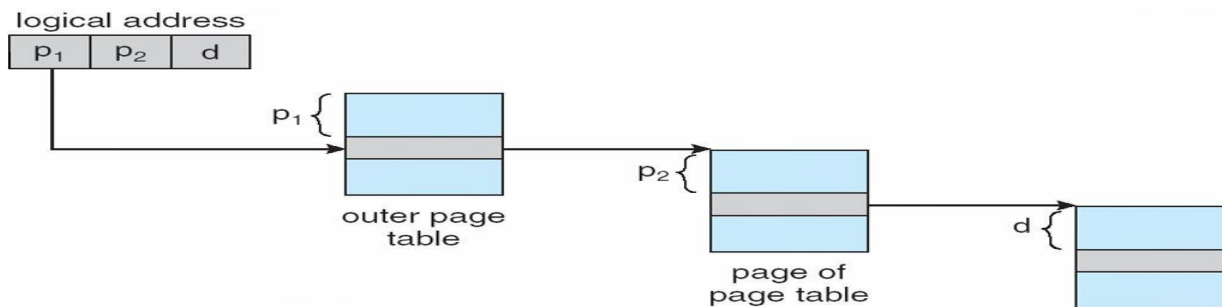
age number		page offset
$p_i$	$p_2$	$d$

12

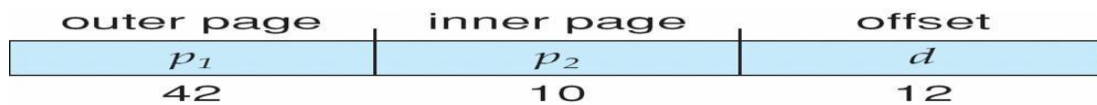
10

10

### Address-Translation Scheme



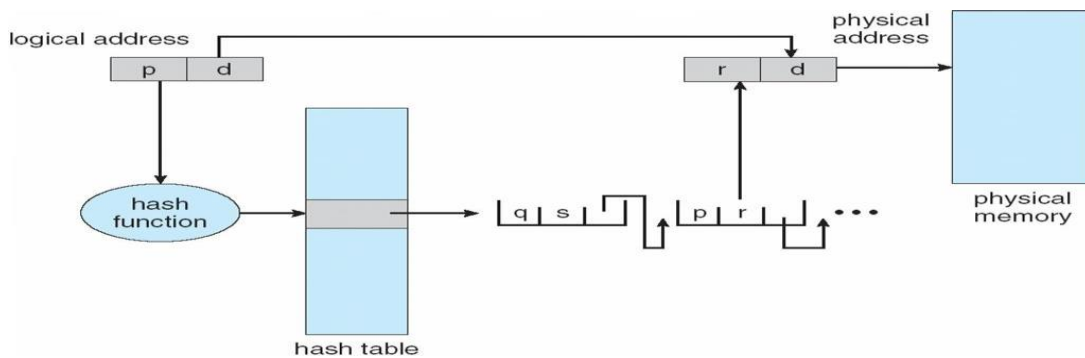
### Three-level Paging Scheme



### Hashed Page Tables

- Common in address spaces  $> 32$  bits
- 
- The virtual page number is hashed into a page table
- 
- This page table contains a chain of elements hashing to the same location Virtual page numbers are compared in this chain searching for a match

If a match is found, the corresponding physical frame is extracted

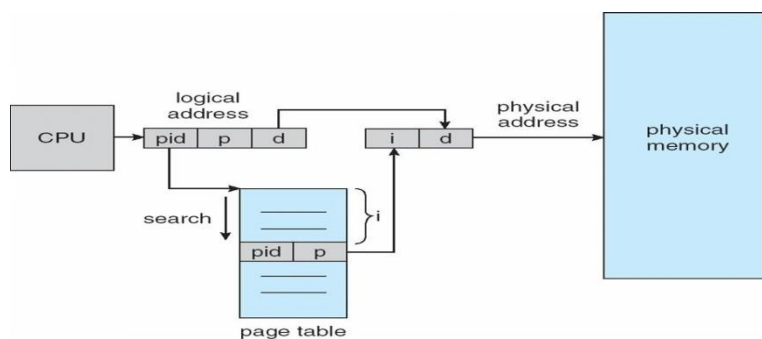


## Hashed Page Table

## Inverted Page Table

- One entry for each real page of memory
- Entry consists of the virtual address of the page stored in that real memory location, with information about the process that owns that page
- Decreases memory needed to store each page table, but increases time needed to search the table when a page reference occurs

Use hash table to limit the search to one — or at most a few — page-table entries

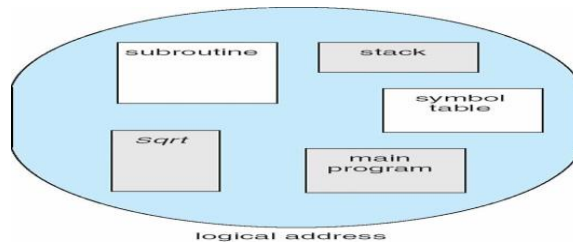


## Inverted Page Table Architecture

## Segmentation

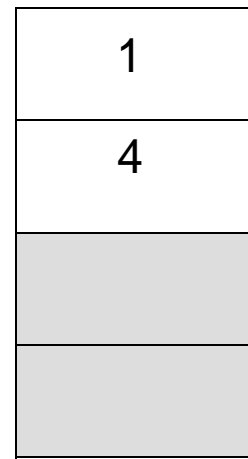
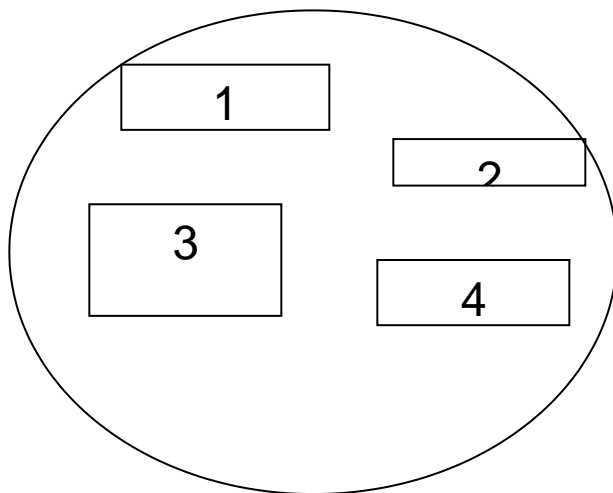
Memory-management scheme that supports user view of memory A program is a collection of segments

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



## User's View of a Program

## Logical View of Segmentation



## Segmentation Architecture

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

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

**limit** – specifies the length of the segment

**Segment-table base register (STBR)** points to the segment table's location in memory

**Segment-table length register (STLR)** indicates number of segments used by a program; segment number  $s$  is legal if  $s < \text{STLR}$

Protection

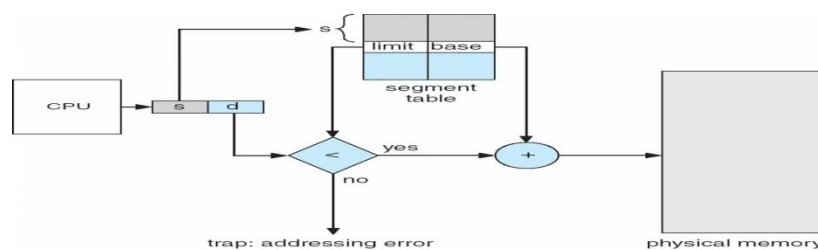
With each entry in segment table associate:

validation bit = 0  $\Rightarrow$  illegal segment read/write/execute privileges

□

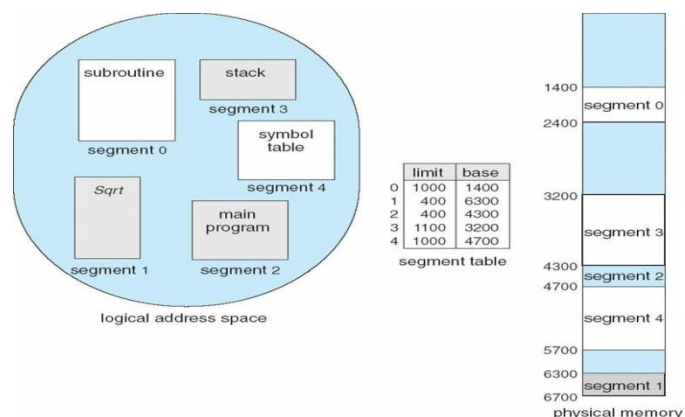
Protection bits associated with segments; code sharing occurs at segment level

Since segments vary in length, memory allocation is a dynamic storage-allocation problem. A segmentation example is shown in the following diagram.



## Segmentation Hardware

### Example of Segmentation





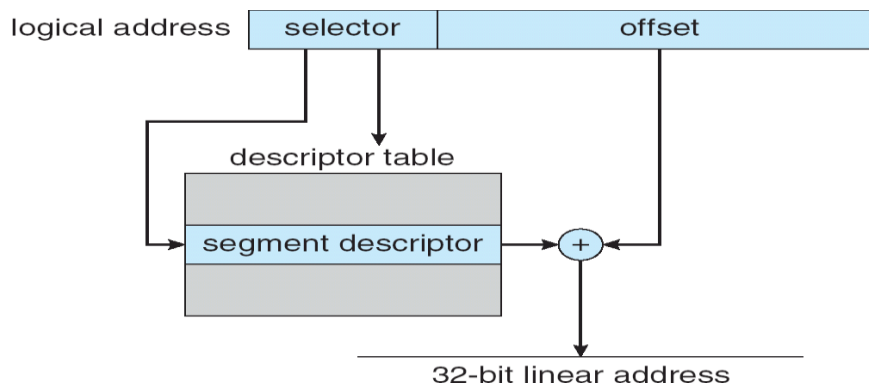
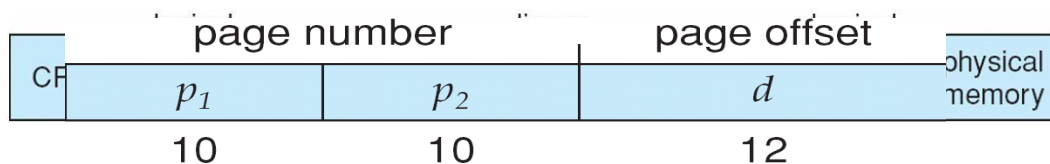
### Example: The Intel Pentium

- Supports both segmentation and segmentation with paging CPU generates logical address
- Given to segmentation unit

Which produces linear addresses • Linear address given to paging unit

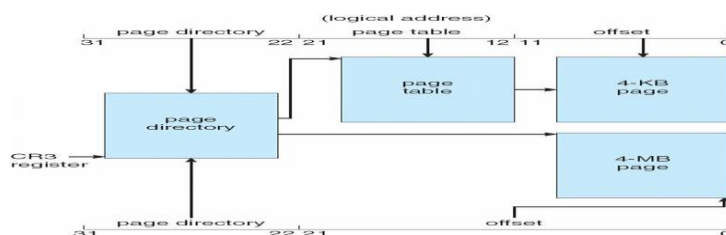
Which generates physical address in main memory Paging units form equivalent of MMU

### Logical to Physical Address Translation in Pentium



### Intel Pentium Segmentation

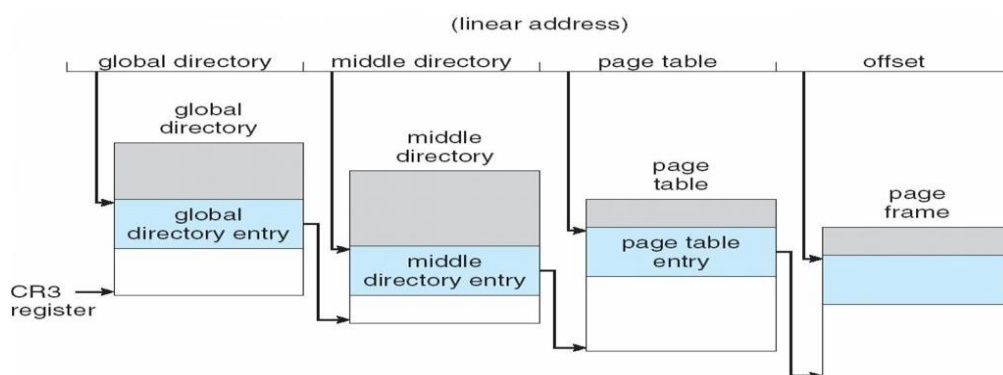
### Pentium Paging Architecture



global directory	middle directory	page table	offset
---------------------	---------------------	---------------	--------

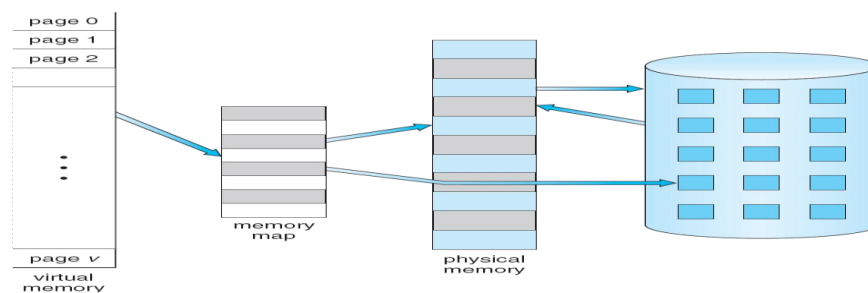
## Linear Address in Linux

### Three-level Paging in Linux



## UNIT – 5

### VIRTUAL MEMORY



### Objective

- To describe the benefits of a virtual memory system.
- To explain the concepts of demand paging, page-replacement algorithms, and allocation of page frames.

■ To discuss the principle of the working-set model.

## **Virtual Memory**

■ Virtual memory is a technique that allows the execution of process that may not be completely in memory. The main visible advantage of this scheme is that programs can be larger than physical memory.

■ Virtual memory is the separation of user logical memory from physical memory this separation allows an extremely large virtual memory to be provided for programmers when only a smaller physical memory is available ( Fig ).

■ Following are the situations, when entire program is not required to load fully.

1. User written error handling routines are used only when an error occurs in the data or computation.
2. Certain options and features of a program may be used rarely.
3. Many tables are assigned a fixed amount of address space even though only a small amount of the table is actually used.

■ The ability to execute a program that is only partially in memory would counter many benefits.

1. Less number of I/O would be needed to load or swap each user program into memory.
2. A program would no longer be constrained by the amount of physical memory that is available.
3. Each user program could take less physical memory, more programs could be run the same time, with a corresponding increase in CPU utilization and throughput.

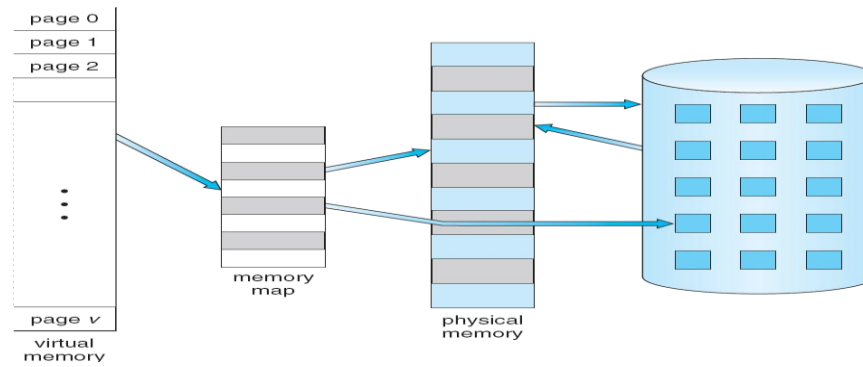
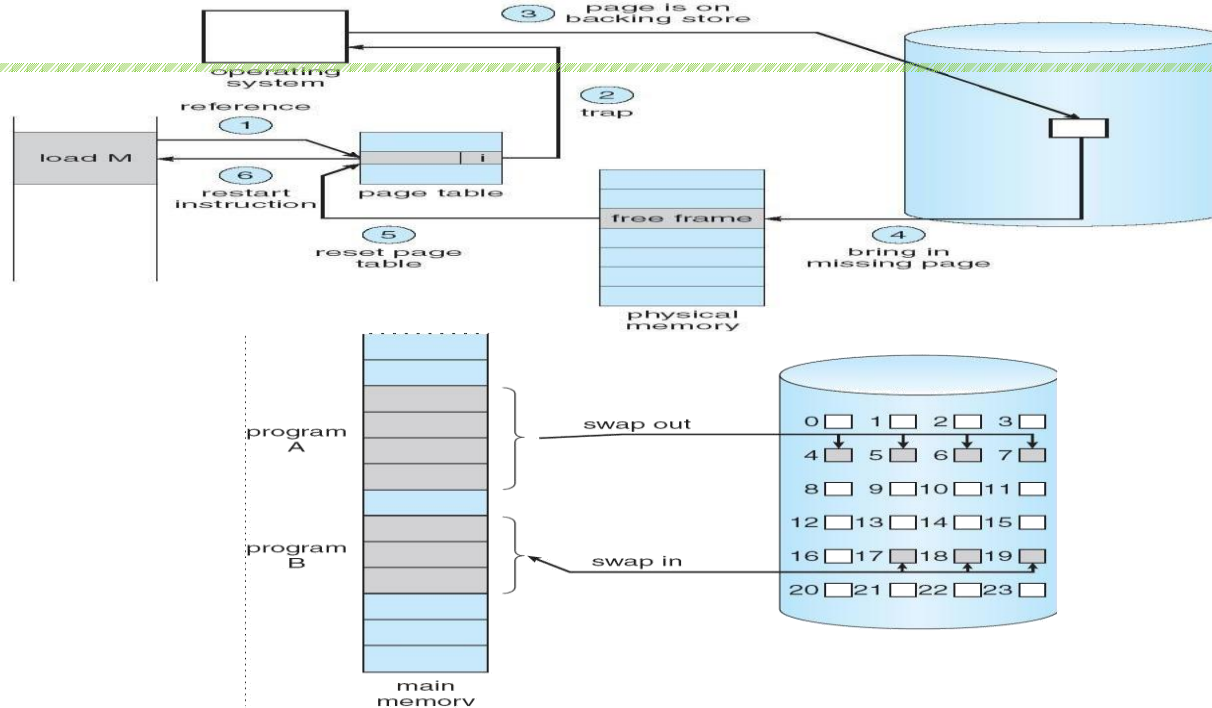


Fig. Diagram showing virtual memory that is larger than physical memory. Virtual memory is commonly implemented by demand paging. It can also be implemented in a segmentation system. Demand segmentation can also be used to provide virtual memory.

## Demand Paging

A demand paging is similar to a paging system with swapping(Fig 5.2). When we want to execute a process, we swap it into memory. Rather than swapping the entire process into memory. When a process is to be swapped in, the pager guesses which pages will be used before the process is swapped out again. Instead of swapping in a whole process, the pager brings only those necessary pages into memory. Thus, it avoids reading into memory pages that will not be used in anyway, decreasing the swap time and the amount of physical memory needed. Hardware support is required to distinguish between those pages that are in memory and those pages that are on the disk using the valid-invalid bit scheme. Where valid and invalid pages can be checked checking the bit and marking a page will have no effect if the process never attempts to access the pages. While the process executes and accesses pages that are memory resident, execution proceeds normally.



**Fig. Transfer of a paged memory to continuous disk space**

Access to a page marked invalid causes a page-fault trap. This trap is the result of the operating system's failure to bring the desired page into memory. But page fault can be handled as following (Fig 5.3):

**Fig. Steps in handling a page fault**

1. We check an internal table for this process to determine whether the reference was a valid or invalid memory access.
2. If the reference was invalid, we terminate the process. If it was valid, but we have not yet brought in that page, we now page in the latter.
3. We find a free frame.
4. We schedule a disk operation to read the desired page into the newly allocated frame.
5. When the disk read is complete, we modify the internal table kept with the process and the page table to indicate that the page is now in memory.
6. We restart the instruction that was interrupted by the illegal address trap. The process can now access the page as though it had always been memory.

Therefore, the operating system reads the desired page into memory and restarts the process as though the page had always been in memory. The page replacement is used to make the frame free if they are not in used. If no frame is free then other process is called in.

### **Advantages of Demand Paging:**

1. Large virtual memory.
2. More efficient use of memory.
3. Unconstrained multiprogramming. There is no limit on degree of multiprogramming.

### **Disadvantages of Demand Paging:**

4. Number of tables and amount of processor over head for handling page interrupts are greater than in the case of the simple paged management techniques.
5. due to the lack of an explicit constraints on a jobs address space size.

### **Page Replacement Algorithm**

There are many different page replacement algorithms. We evaluate an algorithm by running it on a particular string of memory reference and computing the number of page faults. The string of memory references is called reference string. Reference strings are generated artificially or by tracing a given system and recording the address of each memory reference. The latter choice produces a large number of data.

1. For a given page size we need to consider only the page number, not the entire address.
2. if we have a reference to a page p, then any immediately following references to page p will never cause a page fault. Page p will be in memory after the first reference; the immediately following references will not fault.

Eg:- consider the address sequence

0100, 0432, 0101, 0612, 0102, 0103, 0104, 0101, 0611, 0102, 0103, 0104, 0101, 0610, 0102,  
0103, 0104, 0104, 0101, 0609, 0102, 0105

and reduce to 1, 4, 1, 6, 1, 6, 1, 6, 1, 6, 1

To determine the number of page faults for a particular reference string and page replacement algorithm, we also need to know the number of page frames available. As the number of frames available increase, the number of page faults will decrease.

### **FIFO Algorithm**

The simplest page-replacement algorithm is a FIFO algorithm. A FIFO replacement algorithm associates with each page the time when that page was brought into memory. When a page must be replaced, the oldest page is chosen. We can create a FIFO queue to hold all pages in memory. The first three references (7, 0, 1) cause page faults, and are brought into these empty eg. 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1 and consider 3 frames. This replacement means that the next reference to 0 will fault. Page 1 is then replaced by page 0.

### **Optimal Algorithm**

An optimal page-replacement algorithm has the lowest page-fault rate of all algorithms. An optimal page-replacement algorithm exists, and has been called OPT or MIN. It is simply Replace the page that will not be used for the longest period of time. Now consider the same string with 3 empty frames. The reference to page 2 replaces page 7, because 7 will not be used until reference 15, whereas page 0 will be used at 5, and page 1 at 14. The reference to page 3 replaces page 1, as page 1 will be the last of the three pages in memory to be referenced again. Optimal replacement is much better than a FIFO. The optimal page-replacement algorithm is difficult to implement, because it requires future knowledge of the reference string.

### **LRU Algorithm**

The FIFO algorithm uses the time when a page was brought into memory; the OPT algorithm uses the time when a page is to be used. In LRU replace the page that has not been used for the longest period of time. LRU replacement associates with each page the time of that page's last use. When a page must be replaced, LRU chooses that page that has not been used for the longest period of time. Let  $S^R$  be the reverse of a reference string  $S$ , then the page-fault rate for the OPT algorithm on  $S$  is the same as the page-fault rate for the OPT algorithm on  $S^R$ .

### **LRU Approximation Algorithms**

Some systems provide no hardware support, and other page-replacement algorithm. Many systems provide some help, however, in the form of a reference bit. The reference bit for a page is set, by the hardware, whenever that page is referenced. Reference bits are associated with each entry in the page table. Initially, all bits are cleared (to 0) by the operating system. As a user process executes, the bit associated with each page referenced is set (to 1) by the hardware.

### **Additional-Reference-Bits Algorithm**

The operating system shifts the reference bit for each page into the high-order or of its 5-bit byte, shifting the other bits right 1 bit, discarding the low-order bit.

These 5-bit shift registers contain the history of page use for the last eight time periods. If the shift register contains 00000000, then the page has not been used for eight time periods; a page that is used at least once each period would have a shift register value of 11111111.

### **Second-Chance Algorithm**

The basic algorithm of second-chance replacement is a FIFO replacement algorithm. When a page gets a second chance, its reference bit is cleared and its arrival time is reset to the current time.

### **Enhanced Second-Chance Algorithm**

The second-chance algorithm described above can be enhanced by considering both the reference bit and the modify bit as an ordered pair.

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

### **Counting Algorithms**

There are many other algorithms that can be used for page replacement.



- **LFU Algorithm:** The least frequently used (LFU) page-replacement algorithm requires that the page with the smallest count be replaced. This algorithm suffers from the situation in which a page is used heavily during the initial phase of a process, but then is never used again.
- **MFU Algorithm:** The most frequently used (MFU) page-replacement algorithm is based on the argument that the page with the smallest count was probably just brought in and has yet to be used.

### **Page Buffering Algorithm**

When a page fault occurs, a victim frame is chosen as before. However, the desired page is read into a free frame from the pool before the victim is written out. This procedure allows the process to restart as soon as possible, without waiting for the victim page to be written out. When the victim is later written out, its frame is added to the free-frame pool. When the FIFO replacement algorithm mistakenly replaces a page that is still in active use, that page is quickly retrieved from the free-frame buffer, and no I/O is necessary. The free-frame buffer provides protection against the relatively poor, but simple, FIFO replacement algorithm.

## **UNIT VI**

### **Principles of deadlock**

To develop a description of deadlocks, which prevent sets of concurrent processes from completing their tasks. To present a number of different methods for preventing or avoiding deadlocks in a computer system

### **The Deadlock Problem**

A set of blocked processes each holding a resource and waiting to acquire a resource held by another process in the set

Example

System has 2 disk drives

$P_1$  and  $P_2$  each hold one disk drive and each needs another one Example

semaphores  $A$  and  $B$ , initialized to 1

$P_0$

$P_1$

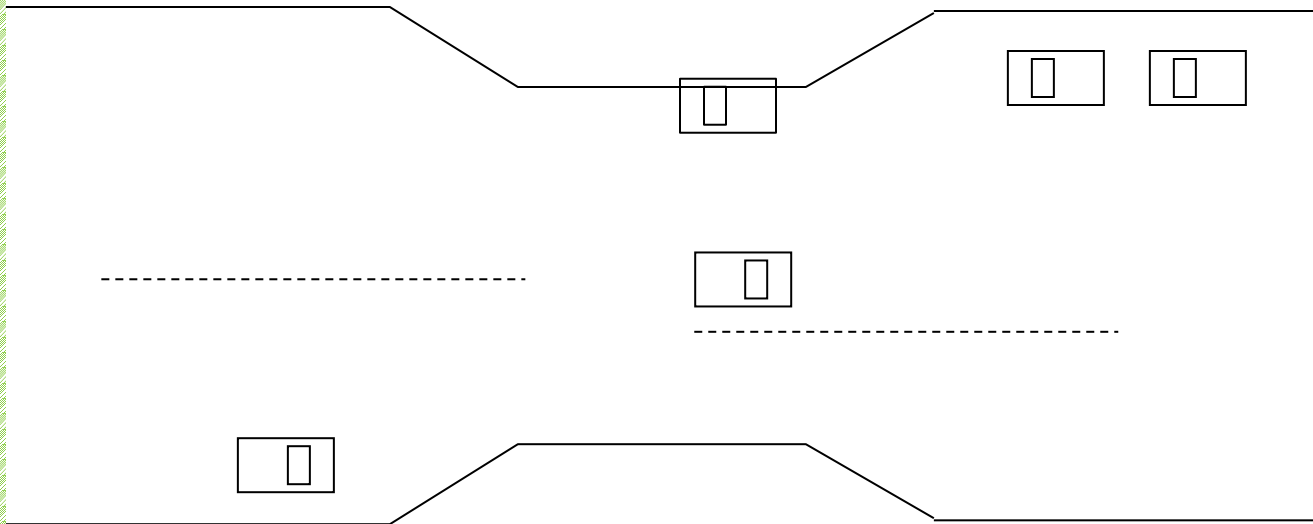
wait (A);

wait(B)

wait (B);

wait(A)

### Bridge Crossing Example



Traffic only in one direction

Each section of a bridge can be viewed as a resource

If a deadlock occurs, it can be resolved if one car backs up (preempt resources and rollback) Several cars may have to be backed up if a deadlock occurs

Starvation is possible

Note – Most OSes do not prevent or deal with deadlocks

### System Model

Resource types  $R_1, R_2, \dots, R_m$

*CPU cycles, memory space, I/O devices* Each resource type  $R_i$  has  $W_i$  instances. Each process utilizes a resource as follows:

Request use release

## Deadlock Characterization

Deadlock can arise if four conditions hold simultaneously

**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  $\{P_0, P_1, \dots, P_0\}$  of waiting processes such that  $P_0$  is waiting for a resource that is held by  $P_1$ ,  $P_1$  is waiting for a resource that is held by

$P_2, \dots, P_{n-1}$  is waiting for a resource that is held by

$P_n$ , and  $P_0$  is waiting for a resource that is held by  $P_0$ .

n

## Resource-Allocation Graph

*A set of vertices V and a set of edges E*

V is partitioned into two types:

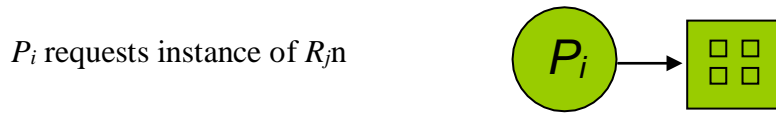
$P = \{P_1, P_2, \dots, P_n\}$ , the set consisting of all the processes in the system  $R = \{R_1, R_2, \dots, R_m\}$ , the set consisting of all resource types in the system request edge – directed edge  $P_i \rightarrow R_j$

assignment edge – directed edge  $R_j \rightarrow P_i$

Process

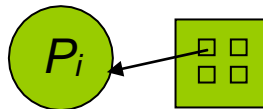


Resource Type with 4 instances

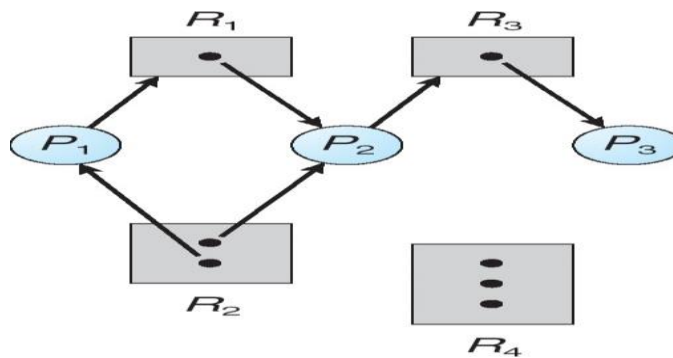


$R_j$

$P_i$  is holding an instance of  $R_j$

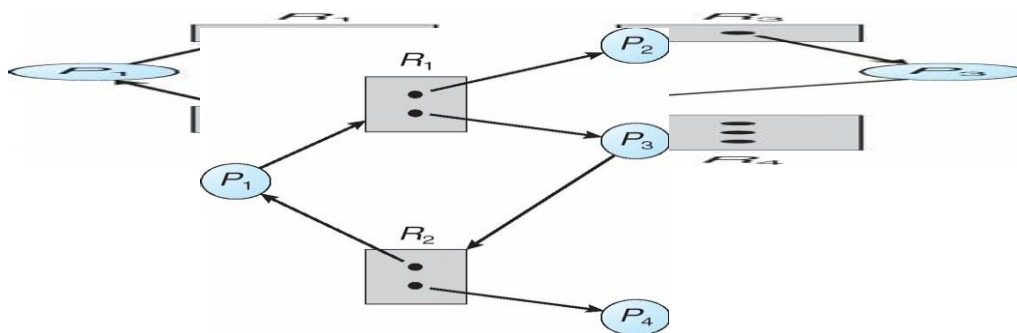


$R_j$



**Example of a Resource Allocation Graph**

**Graph With A Cycle But No Deadlock**



## Basic Facts

If graph contains no cycles  $\Rightarrow$  no deadlock If graph contains a cycle  $\wedge$  if only one instance per resource type, then deadlock if several instances per resource type, possibility of deadlock

## Methods for Handling Deadlocks

Ensure that the system will *never* enter a deadlock state Allow the system to enter a deadlock state and then recover Ignore the problem and pretend that deadlocks never occur in the system; used by most operating systems, including UNIX

## Deadlock Prevention

Restrain the ways request can be made

**Mutual Exclusion** – not required for sharable resources; must hold for nonsharable resources

**Hold and Wait** – must guarantee that whenever a process requests a resource, it does not hold any other resources. Require process to request and be allocated all its resources before it begins execution, or allow process to request resources only when the process has none. Low resource utilization; starvation possible

No Preemption –

If a process that is holding some resources requests another resource that cannot be immediately allocated to it, then all resources currently being held are released. Preempted resources are added to the list of resources for which the process is waiting. Process will be restarted only when it can regain its old resources, as well as the new ones that it is requesting **Circular Wait** – impose a total ordering of all resource types, and require that each process requests resources in an increasing order of enumeration.

## Deadlock Avoidance

Requires that the system has some additional *a priori* information available

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

The deadlock-avoidance algorithm dynamically examines the resource-allocation state to ensure that there can never be a circular-wait condition. Resource-allocation *state* is defined by the number of available and allocated resources, and the maximum demands of the processes

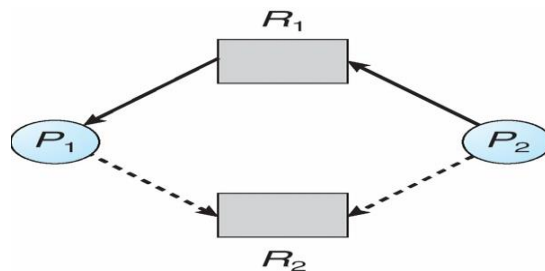
## Safe State

When a process requests an available resource, system must decide if immediate allocation leaves the system in a safe state. System is in safe state if there exists a sequence  $\langle P_1, P_2, \dots, P_n \rangle$  of ALL the processes in the system such that for each  $P_i$ , the resources that  $P_i$  can still request can be satisfied by currently available resources + resources held by all the  $P_j$ , with  $j < i$ . That is:

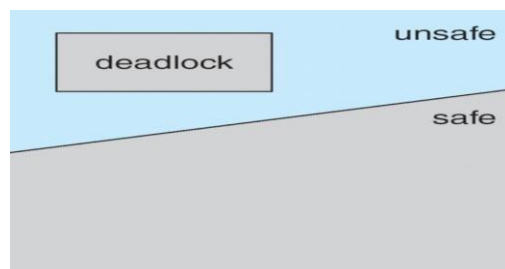
If  $P_i$  resource needs are not immediately available, then  $P_i$  can wait until all  $P_j$  have finished

When  $P_j$  is finished,  $P_i$  can obtain needed resources, execute, return allocated resources, and terminate. When  $P_i$  terminates,  $P_{i+1}$  can obtain its needed resources, and so on

## Basic Facts



If a system is in a safe state, no deadlocks exist. If a system is in an unsafe state, there is a possibility of a deadlock.



Deadlock Avoidance ensures that a system will never enter an unsafe state.

## Safe, Unsafe, Deadlock State

**Avoidance algorithms** Single instance of a resource type Use a resource-allocation graph

Multiple instances of a resource type Use the banker's algorithm

### Resource-Allocation Graph Scheme

nClaim edge  $P_i \rightarrow R_j$  indicated that process  $P_i$  may request resource  $R_j$ ; represented by a dashed line  
nClaim edge converts to request edge when a process requests a resource  
nRequest edge converted to an assignment edge when the resource is allocated to the process

nWhen a resource is released by a process, assignment edge reconverts to a claim edge  
nResources must be claimed *a priori* in the system

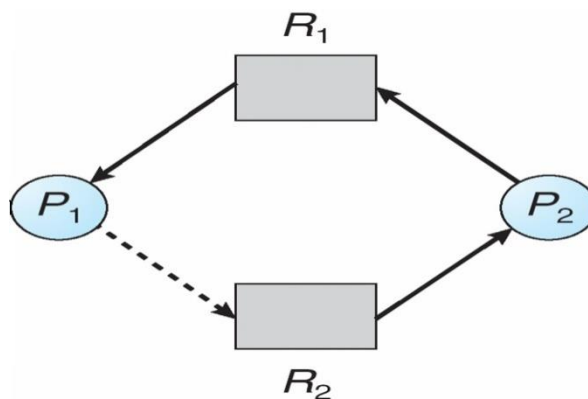
### Resource-Allocation Graph

#### Unsafe State In Resource-Allocation Graph

#### Resource-Allocation Graph Algorithm

Suppose that process  $P_i$  requests a resource  $R_j$ .

nThe request can be granted only if converting the request edge to an assignment edge does not result in the formation of a cycle in the resource allocation graph



### Banker's Algorithm

Multiple instances Each process must a priori claim maximum use When a process requests a resource it may have to wait  $n$  When a process gets all its resources it must return them in a finite amount of time

### Data Structures for the Banker's Algorithm

Let  $n$  = number of processes, and  $m$  = number of resources types.

**nAvailable:** Vector of length  $m$ . If  $available[j] = k$ , there are  $k$  instances of resource type  $R_j$  available

**nMax:**  $n \times m$  matrix. If  $Max[i,j] = k$ , then process  $P_i$  may request at most  $k$  instances of resource type

$R_j$ **nAllocation:**  $n \times m$  matrix. If  $Allocation[i,j] = k$  then  $P_i$  is currently allocated  $k$  instances of

$R_j$ **nNeed:**  $n \times m$  matrix. If  $Need[i,j] = k$ , then  $P_i$  may need  $k$  more instances of  $R_j$  to complete its task

$$Need[i,j] = Max[i,j] - Allocation[i,j]$$

### Safety Algorithm

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

*Work* = *Available*

*Finish* [ $i$ ] = *false* for  $i = 0, 1, \dots, n-1$

2. Find and  $i$  such that both:

(a) *Finish* [ $i$ ] = *false* (b)  $Need_i \leq Work$

If no such  $i$  exists, go to step 4

3.  $Work = Work + Allocation_i$  *Finish* [ $i$ ] = *true*

go to step 2

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

### Resource-Request Algorithm for Process $P_i$



Request =  
vector for  
If  $Request_i$   
process  $P_i$   
instances of  
type  $R_j$ 1.

A B C	A B C	A B C		
	$P_0$	0 1 0	7 5 3	3 3 2
	$P_1$	2 0 0	3 2 2	
	$P_2$	3 0 2	9 0 2	
	$P_3$	2 1 1	2 2 2	
	$P_4$	0 0 2	4 3 3	

request  
process  $P_i$ .  
[j] = k then  
wants k  
resource  
since process has exceeded its max

2. If  $Request_i \leq Available$ , go to step 3. Otherwise  $P_i$  must wait, since resources are not available

3. Pretend to allocate requested resources to  $P_i$  by modifying the state as follows:  
 $Available = Available - Request$ ;  $Allocation_i = Allocation_i + Request$ ;  $Need_i = Need_i - Request$ ;

If safe  $P$  the resources are allocated to  $P_i$

If unsafe  $P$   $P_i$  must wait, and the old resource-allocation state is restored

### Example of Banker's Algorithm

5 processes  $P_0$  through  $P_4$ ; 3 resource types:

A (10 instances), B (5 instances), and C (7 instances) Snapshot at time  $T_0$ :

Allocation

Max Available

The content of the matrix  $Need$  is defined to be  $Max - Allocation$  Need A B C

$P_0$  7 4 3

$P_1$  1 2 2

$P_2$  6 0 0

$P_3$  0 1 1

$P_4$  4 3 1

The system is in a safe state since the sequence  $\langle P_1, P_3, P_4, P_2, P_0 \rangle$  satisfies safety criteria

**Example:  $P_1$  Request (1,0,2)**

Check that Request $\leq$ Available (that is, $(1,0,2) \leq (3,3,2)$ is true	<u>Allocation</u>	<u>Need</u>	
<u>Available</u>	<u>A B C</u>	<u>A B C</u>	<u>A B C</u>
$P_0$	0 1 0	7 4 3	2 3 0
$P_1$	3 0 2	0 2 0	
$P_2$	3 0 1	6 0 0	
$P_3$	2 1 1	0 1 1	
$P_4$	0 0 2	4 3 1	

Executing safety algorithm shows that sequence  $\langle P_1, P_3, P_4, P_0, P_2 \rangle$  satisfies safety requirement Can request for  $(3,3,0)$  by  $P_4$  be granted?

Can request for  $(0,2,0)$  by  $P_0$  be granted?

### Deadlock Detection

Allow system to enter deadlock state Detection algorithm Recovery scheme

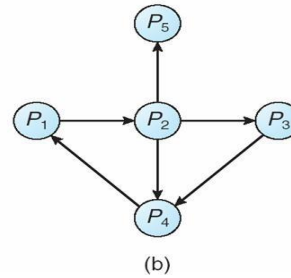
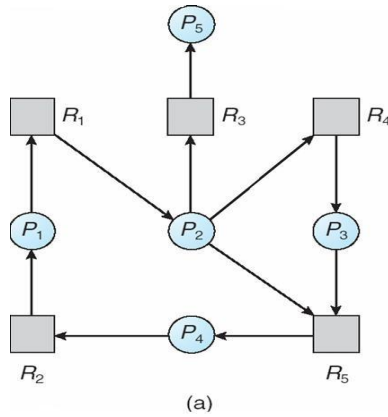
### Single Instance of Each Resource Type

Maintain *wait-for* graph Nodes are processes

$P_i \otimes P_j$  if  $P_i$  is waiting for  $P_j$

Periodically invoke an algorithm that searches for a cycle in the graph. If there is a cycle, there exists a deadlock An algorithm to detect a cycle in a graph requires an order of  $n^2$  operations, where  $n$  is the number of vertices in the graph

### Resource-Allocation Graph and Wait-for Graph



Resource-Allocation Graph

Corresponding wait-for graph

### Several Instances of a Resource Type

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

### Detection Algorithm

1. Let  $Work$  and  $Finish$  be vectors of length  $m$  and  $n$ , respectively Initialize:
    - (a)  $Work = Available$  (b) For  $i = 1, 2, \dots, n$ , if  $Allocation_i \neq 0$ , then  $Finish[i] = false$ ; otherwise,  $Finish[i] = true$
  2. Find an index  $i$  such that both:
    - (a)  $Finish[i] == false$  (b)  $Request_i \leq Work$
 If no such  $i$  exists, go to step 4
  3.  $Work = Work + Allocation_i$   $Finish[i] = true$
- go to step 2. If  $Finish[i] == false$ , for some  $i$ ,  $1 \leq i \leq n$ , then the system is in deadlock state. Moreover, if  $Finish[i] == false$ , then  $P_i$  is deadlocked

Algorithm requires an order of  $O(m \times n^2)$  operations to detect whether the system is in deadlocked state

<b>Example of Detection Algorithm</b>			
Five processes $P_0$ through $P_4$ ; three resource types A (7 instances), B (2 instances), and C (6 instances) Snapshot at time $T_0$ :			
	<u>Allocation</u>	<u>Request</u>	<u>Available</u>
	A B C	A B C	A B C
$P_0$	0 1 0	0 0 0	0 0 0
$P_1$	2 0 0	2 0 2	
$P_2$	3 0 3	0 0 0	
$P_3$	2 1 1	1 0 0	
$P_4$	0 0 2	0 0 2	
Sequence $\langle P_0, P_2, P_3, P_1, P_4 \rangle$ will result in $Finish[i] = \text{true}$ for all $i$			
$P_2$ requests an additional instance of type C		<u>Request</u> A B C	
		0 0 0	
$P_0$			
$P_1$		2 0 1	
$P_2$		0 0 1	
$P_3$		1 0 0	
$P_4$		0 0 2	

State of system?

Can reclaim resources held by process  $P_0$ , but insufficient resources to fulfill other processes; requests Deadlock exists, consisting of processes  $P_1, P_2, P_3$ , and  $P_4$

### Detection-Algorithm Usage

When, and how often, to invoke depends on:

How often a deadlock is likely to occur?

How many processes will need to be rolled back?

one for each disjoint cycle. If detection algorithm is invoked arbitrarily, there may be many cycles in the resource graph and so we would not be able to tell which of the many deadlocked processes “caused” the deadlock.

### **Recovery from Deadlock: Process Termination**

Abort all deadlocked processes. Abort one process at a time until the deadlock cycle is eliminated. In which order should we choose to abort?

Priority of the process

How long process has computed, and how much longer to completion

Resources the process has used

Resources process needs to complete

How many processes will need to be terminated

Is process interactive or batch?

### **Recovery from Deadlock: Resource Preemption**

Selecting a victim – minimize cost

Rollback – return to some safe state, restart process for that state

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

## **UNIT VII**

### **FILE SYSTEM INTERFACE**

#### **The Concept Of a File**

A file is a named collection of related information that is recorded on secondary storage. The information in a file is defined by its creator. Many different types of information may be stored in a file.

## **File attributes:-**

A file is named and for the user's convenience is referred to by its name. A name is

usually a string of characters. One user might create file, where as another user might edit that file by specifying its name. There are different types of attributes.

- 1) **name:-** the name can be in the human readable form.
- 2) **type:-** this information is needed for those systems that support different types. 3)**location:-** this information is used to a device and to the location of the file on that device.
- 4)**size:-** this indicates the size of the file in bytes or words. 5)**protection:-**

## **6)time,date, and user identifications:-**

the information about all files is kept in the directory structure, that also resides on secondary storage.

## **File operations:- Creating a file:-**

Two steps are necessary to create a file first, space in the file system must be found for the file. Second , an entry for the new file must be made in the directory. The directory entry records the name of the file and the location in the system.

## **Writing a file:-**

To write a file give the name of the file, the system search the directory to find the location of the file. The system must keep the *writer* pointer to the location in the file where the next write is to take place. The write pointer must be updated whenever a write occurs.

**Reading a file:-** to read from a file, specifies the name of the file and directory is search for the associated directory entry, and the system needs to keep *read* pointer to the location in the file where the next read is to take place. Once the read has taken place, read pointer is updated.

## **Repositioning with in a file:-**

The directory is searched for the appropriate entry and the current file position is set to given value. this is also known as a file seek.

**Deleting a file:-** to delete a file , we search the directory for the name file. Found that file in the directory entry, we release all file space and erase the directory entry.

**Truncate a file:-** this function allows all attributes to remain unchanged(except for file length) but for the file to be reset to length zero.

**Appending:-** add new information to the end of an existing file .

**Renaming:-** give new name to an existing file.

**Open a file:-**if file need to be used, the first step is to open the file, using the *open* system call.

**Close:-** close is a system call used to terminate the use of an already used file.

### **File Types:-**

A common technique for implementing file type is to include the type as part of the file name. The name is split in to two parts

1)            the name 2) and an extension .

the system uses the extension to indicate the type of the file and the type of operations that can be done on that file.

### **: ACCESSMETHODS:-**

**There are** several ways that the information in the file can be accessed. **1)sequential method 2) direct access method 3) other access methods.** 1)sequential access method:-

the simplest access method is S.A. information in the file is processed in order, one after the other. the bulk of the operations on a file are reads & writes. It is based on a tape model of a file. Fig 10.3

2)            Direct access:- or relative access:-

a file is made up of fixed length records, that allow programs to read and write record rapidly in no particular order. For direct access, file is viewed as a numbered sequence of blocks or records. A direct access file allows, blocks to be read & write. So we may read block15, block 54 or write block10. there is no restrictions on the order of reading or writing for a direct access file. It is great useful for immediate access to large amount of information.

The file operations must be modified to include the block number as a parameter. We have read n, where n is the block number.

### 3) other access methods:-

the other access methods are based on the index for the file. The indexed contain pointers to the various blocks. To find an entry in the file, we first search the index and then use the pointer to access the file directly and to find the desired entry. 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 file would contain pointers to secondary index files which would point to the actual data items

### **Directory Structures:-**

operations that are be on a directory (read in text book)

#### **single level directory:-**

the simple directory structure is the single level directory. All files are contained in

the same directory. Which is easy to understand. Since all files are in same directory, they must have unique names.

In a single level directory there is some limitations. When the no.of files

increases or when there is more than one user some problems can occurs. If the no.of files increases, it becomes difficult to remember the names of all the files. FIG 10.7 **Two-level directory:-**

The major disadvantages to a single level directory is the confusion of file names between different users. The standard solution is to create separate directory for each user.

In 2-level directory structure, each user has her own user file directory(ufd). Each ufd has a similar structure, the user first search the master file directory. the mfd is indexed by user name and each entry point to the ufd for that user.fig 10.8

To create a file for a user, the O.S search only that user's ufd to find whether another file of that name exists. To delete a file the O.S only search to the local ufd and it can not accidentally delete another user's file that has the same name.



This solves the name collision problem, but it still have another. This is disadvantages when the user wants to cooperate on some task and to access one another's file . some systems simply do not allow local user files to be accessed by other user.

Any file is accessed by using path name. Here the user name and a file name defines a path name.

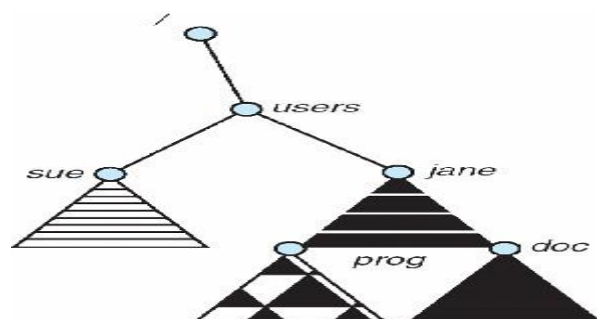
Ex:- user1/ob

In MS-DOS a file specification is C:/directory name/file name **Tree structured directory:-**

This allows users to create their own subdirectories and to organize their files accordingly. here the tree hasa root directory. And 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.FIG 10.9.

A directory contains a set of subdirectories or files. A directory is simply another file, but it is treated in a special way. Here the path names can be of two types. 1)absolute path and 2) relative path.

An absolute path name begins at the root and follows a path down to the specified file, giving the directory **Mount Point**



### **:File Sharing**

Sharing of files on multi-user systems is desirableSharing may be done through a **protection** schemenOn distributed systems, files may be shared

across a networknNetwork File System (NFS) is a common distributed filessharing method

### **File Sharing – Multiple Users**

**User IDs** identify users, allowing permissions and protections to be perusern**Group IDs** allow users to be in groups, permitting group access rights

## File Sharing – Remote File Systems

**n**Uses networking to allow file system access between systems

**I**Manually via programs like FTP

**I**Automatically, seamlessly using **distributed file systems**

**I**Semi automatically via the **world wide web**

**n****Client-server** model allows clients to mount remote file systems from servers

**I**Server can serve multiple clients

**I**Client and user-on-client identification is insecure or complicated

**I****NFS** is standard UNIX client-server file sharing protocol

**I****CIFS** is standard Windows protocol

**I**Standard operating system file calls are translated into remote calls **n**Distributed Information Systems (**distributed naming services**) such as LDAP, DNS, NIS, Active Directory implement unified access to information needed for remote computing

## File Sharing – Failure Modes

Remote file systems add new failure modes, due to network failure, server failure, Recovery from failure can involve state information about status of each remote request

Stateless protocols such as NFS include all information in each request, allowing easy recovery but less security

## File Sharing – Consistency Semantics

**n****Consistency semantics** specify how multiple users are to access a shared file simultaneously

**I**Similar to Ch 7 process synchronization algorithms

☐ Tend to be less complex due to disk I/O and network latency (for remote file systems)

**I**Andrew File System (AFS) implemented complex remote file sharing semantics

**I**Unix file system (UFS) implements:

- ☐ Writes to an open file visible immediately to other users of the same open file
- ☐ Sharing file pointer to allow multiple users to read and write concurrently

**l**AFS has session semantics

- ☐ Writes only visible to sessions starting after the file is closed

## **:Protection**

File owner/creator should be able to control:

**l**what can be done

**l**by whom **n**Types of access

**l**Read **l**Write **l**Execute **l**Append **l**Delete **l**List

## **Protection:-]**

When the information is kept in the system the major worry is its protection from the both physical damage (Reliability) and improper access(Protection).

The reliability is generally provided by duplicate copies of files.

The protection can be provided in many ways . for some single system user, we might provide protection by physically removing the floppy disks . in a multi-user systems, other mechanism are needed.

### **1) types of access:-**

if the system do not permit access to the files of other users, protection is not needed. Protection mechanism provided by controlling accessing. This can be provided by types of file access. Access is permitted or denied depending on several factors. Suppose we mentioned read that file allows only for read

Read:- read from the file. Write:- write or rewrite the file.

Execute:- load the file in to memory and execute it. Append:- write new information at the end of the file. Delete:- delete the file and free its space for possible reuse.

## FILE SYSTEM IMPLEMENTATION

### :File allocation methods:-

There are 3 major methods of allocating disk space.

#### 1) Contiguous allocation:-

1) The contiguous allocation method requires each file to occupy a set of contiguous block on the disk.

2) Contiguous allocation of a file is defined by the disk address and length of the first block. If the file is 'n' block long and starts at location 'b', then it occupies blocks b,b+1,b+2,...,b+n-1;

3) The directory entry for each file indicates the address of the starting block and length of the area allocated for this file. Fig 11.3

4) Contiguous allocation of file is very easy to access. For the *sequential access*, the file system remembers the disk address of the last block referenced and, when necessary read next block. For *direct access* to block 'i' of a file that starts at block 'b', we can immediately access block b+i. Thus both sequential and direct access can be supported by contiguous allocation.

5) One difficulty with this method is finding space for a new file.

6) Also there are many problems with this method

a) **external fragmentation:-** files are allocated and deleted, the free disk space is broken in to little pieces. The E.F exists when free space is broken in to chunks(large piece) and these chunks are not sufficient for a request of new file.

There is a solution for E.F i.e compaction. All free space compact in to one contiguous space. But the cost of compaction is time.

b) Another problem is determining how much space is needed for a file. When file is created the creator must specifies the size of that file. This becomes to big problem. Suppose if we allocate too little space to a file, sometimes it may not sufficient.

Suppose if we allocate large space sometimes space is wasted.

c) Another problem is if one large file is deleted, that large space is becomes to empty. Another file is loaded in to that space whose size is very small then some space is wasted. that wastage of space is called internal fragmentation.

## 2) **Linked allocation:-**

- 1) Linked allocation solves all the problems of contagious allocation. With linked allocation , each file is a linked list of disk blocks, the disk block may be scattered any where on the disk.
- 2) The directory contains a pointer to the first and last blocks of the file. **Fig11.4** Ex:- a file have five blocks start at block 9, continue at block 16, then block 1, block 10 and finally block 25. each block contains a pointer to the next block. These pointers are not available to the user.
- 3) To create a new file we simply create a new entry in directory. With linked allocation, each directory entry has a pointer to the first disk block of the file.
- 3) There is no external fragmentation with linked allocation. Also there is no need to declare the size of a file when that file is created. A file can continue to grow as long as there are free blocks.
- 4) But it has disadvantage. The major problem is that it can be used only for sequential access-files.
- 5) To find the I<sup>th</sup> block of a file , we must start at the beginning of that file, and follow the pointers until we get to the I<sup>th</sup> block. It can not support the direct access.
- 6) Another disadvantage is it requires space for the pointers. If a pointer requires 4 bytes out of 512 byte block, then 0.78% of disk is being used for pointers, rather than for information.
- 7) The solution to this problem is to allocate blocks in multiples, called clusters and to allocate the clusters rather than blocks.
- 8) Another problem is reliability. The files are linked together by pointers scattered all over the disk what happen if a pointer were lost or damaged. **FAT( file allocation table):-**

An important variation on the linked allocation method is the use of a file allocation table.

The table has one entry for each disk block, and is indexed by block number. The FAT is used much as is a linked list.

The directory entry contains the block number of the first block of the file. The table entry contains the block number then contains the block number of the next block in the file. This chain continues until the last block, which has a special end of file value as the table entry. Unused blocks are indicated by a '0' table value. Allocating a new block to a file is simple. First finding the first 0-value table entry, and replacing the previously end of file value with the address of the new block. The 0 is then replaced with end of file value.

**Fig 11.5**

**3) Indexed allocation:-**

1) linked allocation solves the external fragmentation and size declaration problems of contiguous allocation. However in the absence of a FAT, linked allocation can not support efficient direct access.

2) The pointers to the blocks are scattered with the blocks themselves all over the disk and need to be retrieved in order.

3) Indexed allocation solves this problem by bringing all the pointers together in to one location i.e *the index block*.

4) Each file has its own index block, which is an array of disk block addresses. The  $I^{th}$  entry in the index block points to the  $i^{th}$  block of the file.

5) The directory contains the address of the index block. **Fig 11.6**

To read the  $i^{th}$  block we use the pointer in the  $i^{th}$  index block entry to find and read the desired block.

6) When the file is created, all pointers in the index block are set to nil. When the  $i^{th}$  block is first written, a block is obtained from the free space manager, and its address is put in the  $i^{th}$  index block entry.

7) It supports the direct access without suffering from external fragmentation, but it suffers from the wasted space. The pointer overhead of the index block is generally greater than the pointer overhead of linked allocation.

**:Free space management:-**

1) to keep track of free disk space, the system maintains a free space list. The free space list records all disk blocks that are free.

2) To create a file we search the free space list for the required amount of space, and allocate that space to the new file. This space is then removed from the free space list.

3) When the file is deleted, its disk space is added to the free space list. There are many methods to find the free space.

**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.

**Ex:-** consider a disk where blocks 2,3,4,5,8,9,10,11,12,13,17,18,25, are free and rest of blocks are allocated the free space bit map would be **001111001111110001100000010000.....**

the main advantage of this approach is that it is relatively simple and efficient to find the first free block or 'n' consecutive free blocks on the disk

## **2) Linked list:-**

Another approach is to link together all the free disk blocks, keeping a pointer

to the first free block in a special location on the disk and caching it in memory. This first block contain a pointer to the next free disk block, and so on. However this scheme is not efficient to traverse the list, we must read each block, which requires I/O time.

Disk space is also wasted to maintain the pointer to next free space.

## **3) Grouping:-**

Another method is store the addresses of 'n' free blocks in the first free block.

The first (n-1) of these blocks are actually free. The last block contains the addresses of another 'n' free blocks and so on. **Fig 11.8**

**Advantages:-** the main advantage of this approach is that the addresses of a large no.of blocks can be found quickly.

## **4) Counting:-**

Another approach is counting. Generally several contiguous blocks may be allocated or freed simultaneously. Particularly when space is allocated with the contiguous allocation algorithm rather than keeping a list of 'n' free disk address. We can keep the address of first free block and the number 'n' of free contiguous blocks that follow the first block. Each entry in the free space list then consists of a disk address and a count.

## **:Directory Implementation:-**

### **1) Linear list:-**

1) the simple method of implement ting a directory is to use a linear list of file names with pointers to the data blocks.

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

- 3) This method is simple to program but is time consuming to execute.
- 4) To create a new file, we must first search the directory to be sure that no existing file has the same name. Then, we add a new entry at the end of the directory.
- 5) To delete a file we search the directory for the named file, then release the space allocated to it.
- 6) To reuse directory entry, we can do one of several things.
- 7) We can mark the entry as unused or we can attach it to a list of free directory entries.

Disadvantage:- the disadvantage of a linear list of directory entries is the linear search to find a file.

## **UNIT VIII**

### **MASS-STORAGE STRUCTURE**

#### **Mass-Storage Systems**

nDescribe the physical structure of secondary and tertiary storage devices and the resulting effects on the uses of the devices  
nExplain the performance characteristics of mass-storage devices  
nDiscuss operating-system services provided for mass storage, including RAID and HSM

#### **:Overview of Mass Storage Structure**

Magnetic disks provide bulk of secondary storage of modern computers Drives rotate at 60 to 200 times per second

Transfer rate is rate at which data flow between drive and computer

Positioning time (random-access time) is time to move disk arm to desired cylinder (seek time) and time for desired sector to rotate under the disk head (rotational latency) Head crash results from disk head making contact with the disk surface

☐ That's bad

Disks can be removable

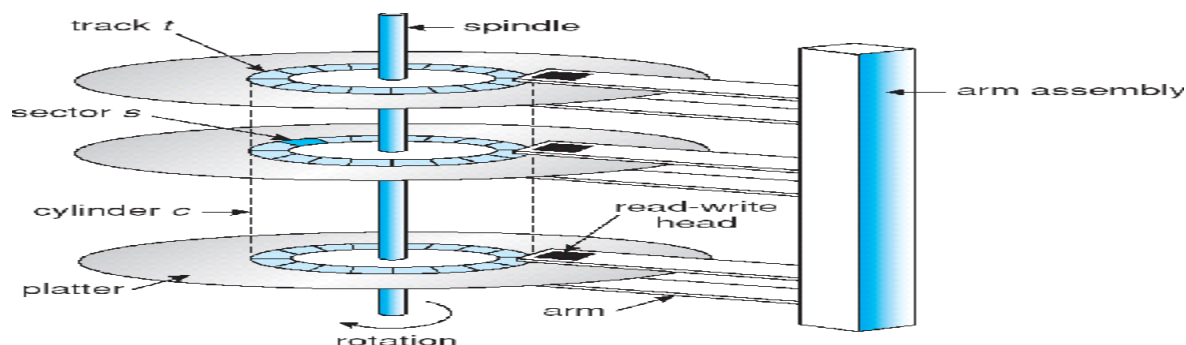
Drive attached to computer via I/O bus

Busses vary, including EIDE, ATA, SATA, USB, Fibre Channel, SCSI



Host controller in computer uses bus to talk to disk controller built into drive or storage array

### Moving-head Disk Mechanism



### Magnetic tape

Was early secondary-storage medium

Relatively permanent and holds large quantities of data Access time slow

Random access ~1000 times slower than disk

Mainly used for backup, storage of infrequently-used data, transfer medium between systems

Kept in spool and wound or rewound past read-write head Once data under head, transfer rates comparable to disk 20-200GB typical storage

Common technologies are 4mm, 8mm, 19mm, LTO-2 and SDLT

### :Disk Structure

Disk drives are addressed as large 1-dimensional arrays of logical

blocks, where the 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 cylinder, and then through the rest of the cylinders from outermost to innermost

### 8.3:Disk Attachment

Host-attached storage accessed through I/O ports talking to I/O busses SCSI itself is a bus, up to 16 devices on one cable, SCSI initiator requests operation and SCSI targets perform tasks

Each target can have up to 8 logical units (disks attached to device controller FC is high-speed serial architecture

Can be switched fabric with 24-bit address space – the basis of storage area networks (SANs) in which many hosts attach to many storage units

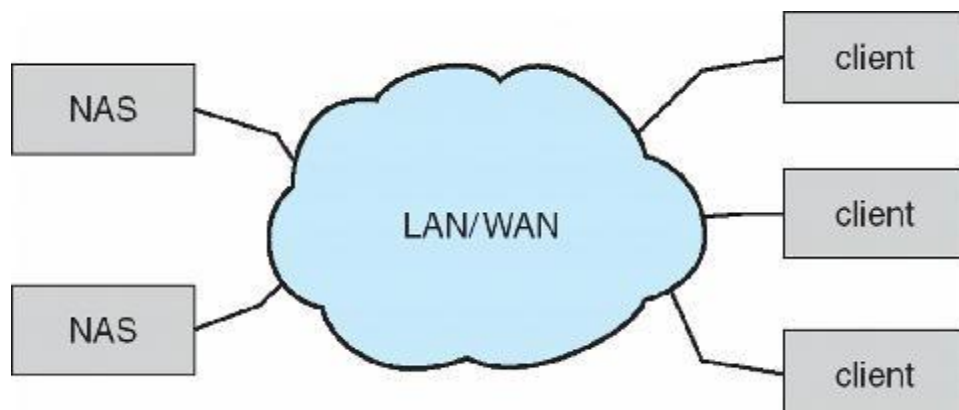
Can be arbitrated loop (FC-AL) of 126 devices

### Network-Attached Storage

Network-attached storage (NAS) is storage made available over a network rather than over a local connection (such as a bus)

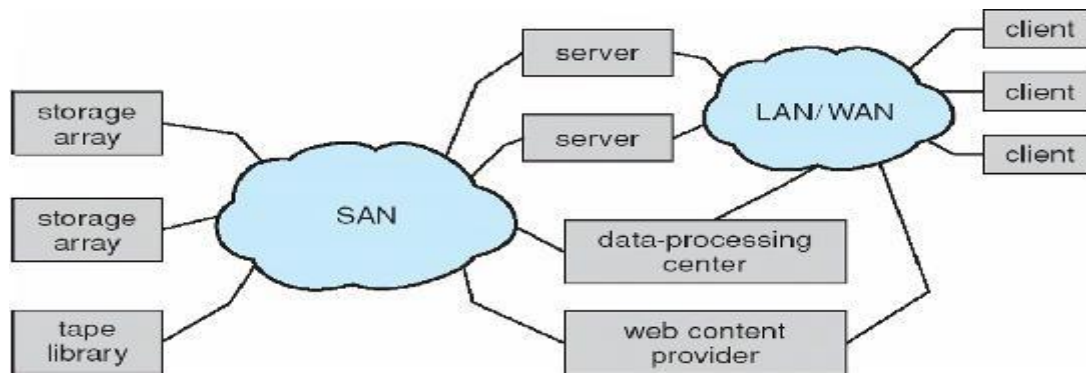
NFS and CIFS are common protocols

Implemented via remote procedure calls (RPCs) between host and storage New iSCSI protocol uses IP network to carry the SCSI protocol



### Storage Area Network

Common in large storage environments (and becoming more common) Multiple hosts attached to multiple storage arrays – flexible



## :Disk Scheduling

The operating system is responsible for using hardware efficiently — for the disk drives, this means having a fast access time and disk bandwidth

Access time has two major components

Seek time is the time for the disk are to move the heads to the cylinder containing the desired sector

Rotational latency is the additional time waiting for the disk to rotate the desired sector to the disk head

Minimize seek time

Seek time » seek distance

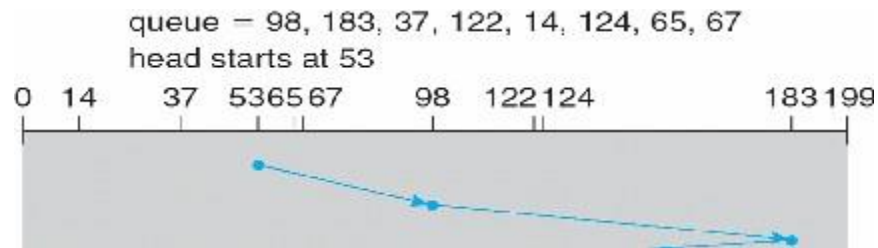
Disk bandwidth is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer Several algorithms exist to schedule the servicing of disk I/O requests nWe illustrate them with a request queue (0-199)

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

Head pointer 53

FCFS

Illustration shows total head movement of 640 cylinders



## SSTF

Selects the request with the minimum seek time from the current head position SSTF scheduling is a form of SJF scheduling; may cause starvation of some requests

Illustration shows total head movement of 236 cylinders

## SCAN

The disk arm starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the other end of the disk, where the head movement is reversed and servicing continues. SCAN algorithm Sometimes called the elevator algorithm

Illustration shows total head movement of 208 cylinders

## C-SCAN

Provides a more uniform wait time than SCAN

The head moves from one end of the disk to the other, servicing requests as it goes

When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip

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

## C-LOOK

Version of C-SCAN

Arm only goes as far as the last request in each direction, then reverses direction immediately, without first going all the way to the end of the disk **Selecting a Disk-Scheduling Algorithm**

SSTF is common and has a natural appeal

SCAN and C-SCAN perform better for systems that place a heavy load on the disk

Performance depends on the number and types of requests

Requests for disk service can be influenced by the file-allocation method The disk-scheduling algorithm should be written as a separate module of the

operating system, allowing it to be replaced with a different algorithm if necessary Either SSTF or LOOK is a reasonable choice for the default algorithm

### Disk Management

Low-level formatting, or physical formatting — Dividing a disk into sectors that the disk controller can read and write

To use a disk to hold files, the operating system still needs to record its own data structures on the disk

Partition the disk into one or more groups of cylinders Logical formatting or “making a file system”

To increase efficiency most file systems group blocks into clusters

☐ Disk I/O done in blocks

☐ File I/O done in clusters Boot block initializes system

The bootstrap is stored in ROM Bootstrap loader program

Methods such as sector sparing used to handle bad blocks

### Booting from a Disk in Windows 2000

#### :Swap-Space Management

Swap-space — Virtual memory uses disk space as an extension of main memory

Swap-space can be carved out of the normal file system, or, more commonly, it can be in a separate disk partition

Swap-space management

4.3BSD allocates swap space when process starts; holds text segment (the program) and data segment

Kernel uses swap maps to track swap-space use

Solaris 2 allocates swap space only when a page is forced out of physical memory, not when the virtual memory page is first created

## Data Structures for Swapping on Linux Systems

### **RAID Structure**

RAID – multiple disk drives provides reliability via redundancyIncreases the mean time to failure  
Frequently combined with NVRAM to improve write performance

RAID is arranged into six different levels

Several improvements in disk-use techniques involve the use of multiple disks working cooperatively  
Disk striping uses a group of disks as one storage unit  
RAID schemes improve performance and improve the reliability of the storage system by storing redundant data

Mirroring or shadowing (RAID 1) keeps duplicate of each disk

Striped mirrors (RAID 1+0) or mirrored stripes (RAID 0+1) provides high

performance and high reliability  
Block interleaved parity (RAID 4, 5, 6) uses much less redundancy  
RAID within a storage array can still fail if the array fails, so automatic

replication of the data between arrays is common

Frequently, a small number of hot-spare disks are left unallocated, automatically replacing a failed disk and having data rebuilt onto them

### RAID (0 + 1) and (1 + 0)

### **Extensions**

RAID alone does not prevent or detect data corruption or other errors, just disk failures

Solaris ZFS adds checksums of all data and metadata

Checksums kept with pointer to object, to detect if object is the right one and whether it changed

Can detect and correct data and metadata corruption ZFS also removes volumes, partitions

Disks allocated in pools

Filesystems with a pool share that pool, use and release space like “malloc” and “free” memory allocate / release calls

ZFS Checksums All Metadata and Data Traditional and Pooled Storage

### **Stable-Storage Implementation**

Write-ahead log scheme requires stable storageTo implement stable storage: Replicate information on more than one nonvolatile storage media with independent failure modes

Update information in a controlled manner to ensure that we can recover the stable data after any failure during data transfer or recovery

### **Tertiary Storage Devices**

Low cost is the defining characteristic of tertiary storageGenerally, tertiary storage is built using removable mediaCommon examples of removable media are floppy disks and CD-ROMs; other types are available

### **Removable Disks**

Floppy disk — thin flexible disk coated with magnetic material, enclosed in a protective plastic caseMost floppies hold about 1 MB; similar technology is used for removable disks that hold more than 1 GB

Removable magnetic disks can be nearly as fast as hard disks, but they are at a greater risk of damage from exposure

A magneto-optic disk records data on a rigid platter coated with magnetic material

Laser heat is used to amplify a large, weak magnetic field to record a bit Laser light is also used to read data (Kerr effect)

The magneto-optic head flies much farther from the disk surface than a magnetic disk head, and the magnetic material is covered with a protective layer of plastic or glass; resistant to head

crashes  
Optical disks do not use magnetism; they employ special materials that are altered by laser light

### WORM Disks

The data on read-write disks can be modified over and over

WORM (“Write Once, Read Many Times”) disks can be written only once Thin aluminum film sandwiched between two glass or plastic platters

To write a bit, the drive uses a laser light to burn a small hole through the aluminum; information can be destroyed by not altered

Very durable and reliable

Read-only disks, such as CD-ROM and DVD, come from the factory with the data pre-recorded

### Tapes

Compared to a disk, a tape is less expensive and holds more data, but random access is much slower

Tape is an economical medium for purposes that do not require fast random access, e.g., backup copies of disk data, holding huge volumes of data Large tape installations typically use robotic tape changers that move tapes between tape drives and storage slots in a tape library

stacker – library that holds a few tapes  
silo – library that holds thousands of tapes

A disk-resident file can be archived to tape for low cost storage; the computer can stage it back into disk storage for active use

### Operating System Support

Major OS jobs are to manage physical devices and to present a virtual machine abstraction to applications  
For hard disks, the OS provides two abstractions:

Raw device – an array of data blocks  
File system – the OS queues and schedules the interleaved requests from

several applications



## Application Interface

Most OSs handle removable disks almost exactly like fixed disks — a new cartridge is formatted and an empty file system is generated on the disk. Tapes are presented as a raw storage medium, i.e., and application does not open a file on the tape, it opens the whole tape drive as a raw device. Usually the tape drive is reserved for the exclusive use of that application.

Since the OS does not provide file system services, the application must decide how to use the array of blocks.

Since every application makes up its own rules for how to organize a tape, a tape full of data can generally only be used by the program that created it. **Tape Drives**

The basic operations for a tape drive differ from those of a disk drive. `locate()` positions the tape to a specific logical block, not an entire track (corresponds to `seek()`).

The `read position()` operation returns the logical block number where the tape head is.

The `space()` operation enables relative motion.

Tape drives are “append-only” devices; updating a block in the middle of the tape also effectively erases everything beyond that block.

An EOT mark is placed after a block that is written.

## File Naming

The issue of naming files on removable media is especially difficult when we want to write data on a removable cartridge on one computer, and then use the cartridge in another computer.

Contemporary OSs generally leave the name space problem unsolved for removable media, and depend on applications and users to figure out how to access and interpret the data.

Some kinds of removable media (e.g., CDs) are so well standardized that all computers use them the same way.

## (Hierarchical Storage Management HSM)

A hierarchical storage system extends the storage hierarchy beyond

primary memory and secondary storage to incorporate tertiary storage — usually implemented as a jukebox of tapes or removable disks

Usually incorporate tertiary storage by extending the file system Small and frequently used files remain on disk

Large, old, inactive files are archived to the jukebox

HSM is usually found in supercomputing centers and other large installations that have enormous volumes of data

## Speed

Two aspects of speed in tertiary storage are bandwidth and latencynBandwidth is measured in bytes per second

Sustained bandwidth – average data rate during a large transfer; # of bytes/transfer time

Data rate when the data stream is actually flowing

Effective bandwidth – average over the entire I/O time, including seek() or

locate(), and cartridge switching Drive's overall data rate

Access latency – amount of time needed to locate data

Access time for a disk – move the arm to the selected cylinder and wait for the rotational latency; < 35 milliseconds

Access on tape requires winding the tape reels until the selected block reaches the tape head; tens or hundreds of seconds

Generally say that random access within a tape cartridge is about a thousand times slower than random access on disk

The low cost of tertiary storage is a result of having many cheap cartridges share a few expensive drives

A removable library is best devoted to the storage of infrequently used data, because the library can only satisfy a relatively small number of I/O requests per hour

### Reliability

A fixed disk drive is likely to be more reliable than a removable disk or tape drive. An optical cartridge is likely to be more reliable than a magnetic disk or tape. A head crash in a fixed hard disk generally destroys the data, whereas the failure of a tape drive or optical disk drive often leaves the data cartridge unharmed.

### Cost

Main memory is much more expensive than disk storage. The cost per megabyte of hard disk storage is competitive with magnetic tape if only one tape is used per drive. The cheapest tape drives and the cheapest disk drives have had about the same storage capacity over the years. Tertiary storage gives a cost savings only when the number of cartridges is considerably larger than the number of drives.

## COMPUTER GRAPHICS (303)

### NOTES

**Definition:** - Computer graphics tells us that what the actual workings of graphics are. Computer Graphics remains one of the most existing and rapidly growing computer field. Computer Graphics

as the pictorial representation or graphical representation of a computer. Advantages of Interactive Graphics:--Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D and 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly and efficiently. In Many design, implementation, and construction processes today, the information pictures can give is virtually indispensable. Scientific visualization became an important field in the late 1980s, when scientists and engineers realized that they could not interpret the data and prodigious quantities of data produced in supercomputer runs without summarizing the data and highlighting trends and phenomena in various kinds of graphical representations. Creating and reproducing pictures, however, presented technical problems that stood in the way of their widespread use. Thus, the ancient Chinese proverb "a picture is worth ten thousand words" became a cliché in our society only after the advent of inexpensive and simple technology for producing pictures---first the printing press, then photography. Interactive computer graphics is the most important means of producing pictures since the invention of photography and television; it has the added advantage that, with the computer, we can make pictures not only of concrete, "real-world" objects but also of abstract, synthetic objects, such as mathematical surfaces in 4D and of data that have no inherent geometry, such as survey results.

Furthermore, we are not confined to static images. Although static pictures are a good means of communicating information, dynamically varying pictures are frequently even better--to time-varying phenomena, both real (e.g., growth trends, such as nuclear energy use in the United States or population movement from cities to suburbs and back to the cities). Thus, a movie can show changes over time more graphically than can a sequence of slides. Thus, a sequence of frames displayed on a screen at more than 15 frames per second can convey smooth motion or changing form better than can a jerky sequence, with several seconds between individual frames. The use of dynamics is especially effective when the user can control the animation by adjusting the speed, the portion of the total scene in view, the amount of detail shown, the geometric relationship of the objects in the another, and so on. Much of interactive graphics technology therefore contains hardware and software for user-controlled motion dynamics and update dynamics. With motion dynamics, objects can be moved and tumbled with respect to a stationary observer. The objects can also remain stationary and the viewer can move around them , pan to select the portion in view, and zoom in or out for more or less detail, as though looking through the viewfinder of a rapidly moving video

camera. In many cases, both the objects and the camera are moving. A typical example is the flight simulator, which combines a mechanical platform supporting a mock cockpit with display screens for windows. Computers control platform motion, gauges, and the simulated world of both stationary and moving objects through which the pilot navigates. These multimillion-dollar systems train pilots by letting the pilots maneuver a simulated craft over a simulated 3D landscape and around simulated vehicles. Much simpler flight simulators are among the most popular games on personal computers and workstations.

Amusement parks also offer "motion-simulator" rides through simulated terrestrial and extraterrestrial landscapes. Video arcades offer graphics-based dexterity games and racecar-driving simulators, video games exploiting interactive motion dynamics: The player can change speed and direction with the "gas pedal" and "steering wheel," as trees, buildings, and other cars go whizzing by. Similarly, motion dynamics lets the user fly around the through buildings, molecules, and 3D or 4D mathematical space. In another type of motion dynamics, the "camera" is held fixed, and the objects in the scene are moved relative to it. For example, a complex mechanical linkage, such as the linkage on a steam engine, can be animated by moving or rotating all the pieces appropriately. Update dynamics is the actual change of the shape, color, or other properties of the objects being viewed. For instance, a system can display the deformations of an airplane structure in flight or the state changes in a block diagram of a nuclear reactor in response to the operator's manipulation of graphical representations of the many control mechanisms.

The smoother the change, the more realistic and meaningful the result. Dynamic interactive graphics offers a large number of user-controllable modes with which to encode and communicate information: the 2D or 3D shape of objects in a picture, their gray scale or color, and the time variations of these properties. With the recent development of digital signal processing (DSP) and audio synthesis chips, audio feedback can now be provided to augment the graphical feedback and to make the simulated environment even more realistic. Interactive computer graphics thus permits extensive, high-bandwidth user-computer interaction. This significantly enhances our ability to understand data, to perceive trends, and to visualize real or imaginary objects--indeed, to create "virtual worlds" that we can explore from arbitrary points of view. By making communication more efficient, graphics make possible higher-quality and more precise results or products, greater productivity, and lower analysis and design costs.

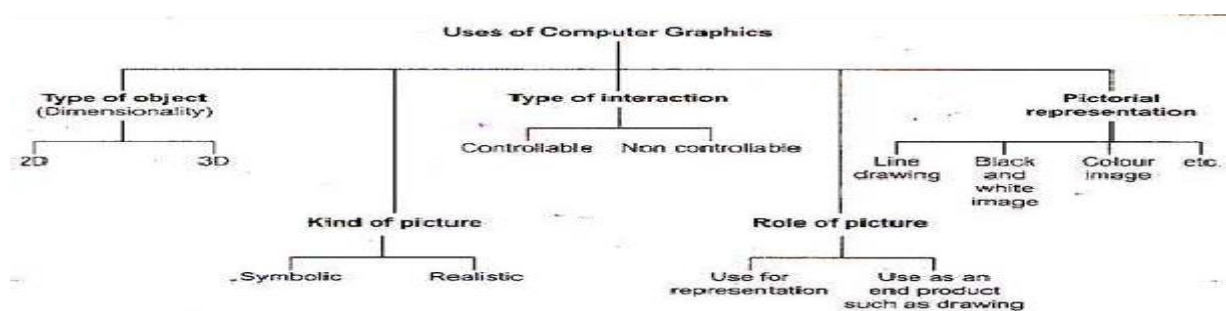
---

**Development of hardware and software for computer graphics:-** The development of hardware of computer graphics involves the development of input and output device technology. Therefore, in all development of computer graphics involves the development in three fields: **1. Output technology 2. Input technology and 3. Software technology**

## Output Technology

Figure 1.3 shows the historical development in the output technology. In early days of computer the hardcopy devices such as teletype printer and line printer were in use with computer driven CRT displays. In mid fifties command and control CRT display consoles were introduced. The more display devices developed in mid-sixties and in common use until the mid-eighties, are called vector, stroke, line drawing or calligraphic displays. The term vector is used as synonyms for line; a stroke is a short line, and characters are made of sequence of such strokes

and characters are made of sequence of such strokes.



## Input Technology

Input technology has also improved greatly over the years. Numbers of input devices were developed over the years. These devices are punch cards, light pens, keyboard, tables, mouse and scanners

## Software Technology

Like output and input technology there is a lot of development in the software technology. In early days low level software were available. Over the years software technology moved from low level to device dependent and then to device independent packages. The device independent packages are high level packages with can drive a wide variety of display and printer devices. As a need for the device independent package standardization is made and specifications are decided. The first graphics specification to be officially standardized was GKS (the Graphical Kernel System). GKS

supports the grouping of logically related primitives such as lines, polygons, and character strings and their attributes in collected form called segments. In 1988, a 3D extension of GKS, became an official standard, as did a much more sophisticated but even more complex graphics system called PHIGS (Programmers Hierarchical Interactive Graphics System). PHIGS, as its name implies, supports nested hierarchical grouping of 3D primitives, called structures. In PHIGS, all primitives are subjected to geometric transformations such as scaling, rotation and translation to accomplish dynamic movement. PHIGS also supports a database of structures the programmer may edit and modify. PHIGS automatically updates the display whenever the database has been modified

### **Application of computer graphics:**

- (1) Computer – Aided Design
- (2) Presentation Graphics
- (3) Computer Art
- (4) Entertainment
- (5) Education and Training
- (6) Graphics provides one of the most natural means of communicating with a computer.
- (7) Interactive computer graphics permits extensive, high-bandwidth user-computer interaction.

---

### **Representative Uses of Computer Graphics**

Computer graphics is used today in many different areas of industry, business, government, education, and entertainment.

- User interfaces: GUI, etc.
- Business, science and technology: histograms, bar and pie charts, etc.
- Office automation and electronic publishing: text, tables, graphs, hypermedia systems, etc.
- Computer-aided design (CAD): structures of building, automobile bodies, etc.
- Simulation and animation for scientific visualization and entertainment: flight simulation, games, movies, virtual reality, etc.
- Art and commerce: terminals in public places such as museums, etc.
- Cartography: map making

- Simulation and animation for scientific visualization and entertainment.
- Multimedia textbooks

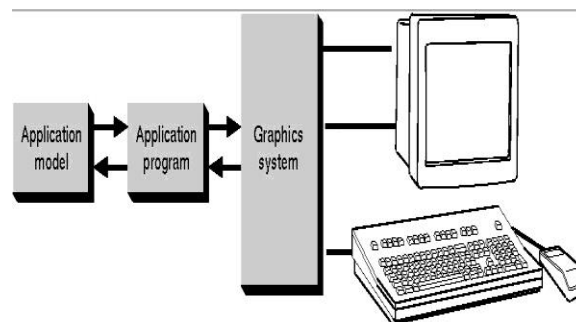
### Classification of Applications

- **Paint programs:** Allow you to create rough freehand drawings.
- **Animation software:** Enables you to chain and sequence a series of images to simulate movement. Each image is like a frame in a movie
- **CAD software:** Enables architects and engineers to draft designs. It is the acronym for computer-aided design. A CAD system is a combination of hardware and software that enables engineers and architects to design everything from furniture to airplanes.
- **Desktop publishing:** Provides a full set of word-processing features as well as fine control over placement of text and graphics, so that you can create newsletters, advertisements, books, and other types of documents
- **business software:** enables users to create highly stylized images for slide shows and reports. The software includes functions for creating various types of charts and graphs and for inserting text in a variety of fonts.

---

### Conceptual Framework for Interactive Graphics.

The high-level conceptual framework shown here can be used to describe almost any interactive graphics system



- The three major parts of the framework are:
- Application Modeling:-Calculating what is to be displayed
- Displaying the Model:-Calling the graphics API routines
- Interaction Handling :-Handling user interaction, which will change the model, and therefore the display typically an event driven loop

### Hardcopy Devices



All the **output devices** can be categorized into two categories

- ☐ Hard Copy Devices
- ☐ Soft Copy Devices

**Hard copy devices** are those that give the output in the tangible form. Printers and Plotters are two common hard copy devices.

**Soft copy devices** give output in the intangible form or the virtual form, e.g. something displayed on a screen. All the computer monitors are covered under this category.

### **Printers**

All the printers irrespective of the technology used can be categorized as

- ☐ Impact Printers
- ☐ Non Impact Printers

**Impact printers** are those printers in which there is a direct contact between the printing head and the paper on which the print is produced.

- ☐ They work by striking a head or a needle against an inked ribbon which leaves a mark on the paper.
- ☐ These printers produce a lot of noise when printing, because of the head striking the paper.
- ☐ Examples are *Dot Matrix*, *Daisy Wheel* and *Line printers*.

In the case of **non-impact printers** the printing head never comes in direct contact with the paper.

- ☐ These printers work by spraying ink on the paper.
- ☐ Electrostatic or electromagnetic charge is used in these printers.
- ☐ Examples are *Ink-Jet* and *Laser* printers.

### **Dot-Matrix Printers :**

- ☐ Dot Matrix is an impact printer.

- ☐ These printer forms characters from individual dots.
- ☐ These printers have a print head which runs back and forth on a paper.
- ☐ The print head has a two-dimensional array of pins called dot matrix. There may be 9 to 24 pins in the dot matrix.

From this array of pins some pins are drawn out (or driven forward) to form the shape of a character.

- ☐ The drawn out pins strike an ink soaked cloth ribbon against a paper. This forms that particular character on the paper.
- ☐ Thus dot matrix printers can be used to print different fonts of characters.
- ☐ Since mechanical force is used, carbon copies of documents can be taken.
- ☐ 40 to 250 characters can be printed per second.

### **Daisy Wheel Printers :**

- ☐ This is an impact printer.
- ☐ Only preformed fonts of characters can be printed.
- ☐ This printer contains a daisy wheel. Daisy wheel is made of plastic or metal. This holds an entire character set as raised characters molded on each "petal".
- ☐ A motor rotates the daisy wheel to position the required character between the hammer and the ribbon.
- ☐ A small hammer then strikes the petal, which in turn strikes the inked ribbon to leave the character mark on the paper.
- ☐ The daisy wheel and hammer are mounted on a sliding carriage similar to that used by dot matrix printers.
- ☐ Different fonts cannot be printed using this technology.

### **Line Printers :**

- ☐ The line printer is a high speed impact printer in which one line is printed at a time.

- ☐ 600-1200 lines can be printed per minute.
- ☐ Drum printer is an example of line printers.
- ☐ These printers are very expensive.
- ☐ These kind of printers were popular in the early days of computers, but the technology is still in use.

### **Drum Printers**

- ☐ In a drum printer, a fixed font character set is engraved onto a number of print wheels.
- ☐ There are as many print wheels as the number of columns (letters in a line) the printer could print.
- ☐ The print wheels are joined to form a large drum (cylinder),
- ☐ This drum spins at high speed and paper and an inked ribbon is moved past the print position.
- ☐ As the desired character for each column passes the print position, a hammer strikes the paper from the rear and presses the paper against the ribbon and the drum, causing the desired character to be printed on the paper.

### **Ink-Jet Printers :**

- ☐ Inkjet printer is a non impact printer, Core of an inkjet printer is the print head.
- ☐ The print head contains an ink cartridge which has a series of nozzles that are used to spray tiny drops of ink on to the paper.
- ☐ Ink cartridges come in various combinations, such as separate black and colour cartridges, colour and black in a single cartridge or even a cartridge for each ink colour.
- ☐ A motor moves the print head back and forth across the paper.
- ☐ Different types of inkjet printers form their droplets of ink in different ways. There are two main inkjet technologies currently used by printer manufacturers
  - o **Thermal bubble** - This method is commonly referred to as **bubble jet**. In a thermal inkjet printer, tiny resistors create heat, and this heat vaporizes ink to create a bubble. As the bubble expands, some

of the ink is pushed out of a nozzle onto the paper. When the bubble "pops" (collapses), a vacuum is created. This pulls more ink into the print head from the cartridge. A typical bubble jet print head has **300 or 600 tiny nozzles**, and all of them can fire a droplet simultaneously.

- o **Piezoelectric** - This technology uses **piezo crystals**. A crystal is located at the back of the ink reservoir of each nozzle. The crystal receives a tiny electric charge that causes it to vibrate. When the crystal vibrates inward, it forces a tiny amount of ink out of the nozzle. When it vibrates out, it pulls some more ink into the reservoir to replace the ink sprayed out.
- ☐ The ink droplets are subjected to an electrostatic field created by a charging electrode as they form. Charged droplets are separated by one or more uncharged "guard droplets" to minimize electrostatic repulsion between neighbouring droplets. The charged droplets pass through an electrostatic field and are directed (deflected) by electrostatic deflection plates to print on the Paper.

### **Laser Printers :**

- ☐ A laser printer is a **non impact** printer, which produces a page of text at a time.
- ☐ Laser printer uses the principle of **Static Electricity** to print.
- ☐ This printer has revolving cylinder called **Drum**.
- ☐ Drum is given a **positive charge**.
- ☐ A **Laser beam** is used to draw the image to be printed, on the drum with negative charge. This discharges some portion of the charge on the drum. This creates electrostatic image of the print on the drum with no charge, and the background is left positively charged.
- ☐ The drum is then exposed to **toner** from which positively charged toner particles mixed with carbon black are released. Since positive charge repels positive charge, the toner particles settle on the discharged areas of the drum, this is exactly the image to be printed.
- ☐ The paper is then pressed against the drum, this transfers the toner particles on to the paper.
- ☐ Paper is then passed through a **fuser**, which is a set of heated rollers, this melts the carbon black on the paper to form the desired print.

### **Plotters :**

Another hard copy output device is plotter. Plotter is a printing device which can draw continuous lines. This is useful to print vector graphics rather than raster graphics unlike normal printers. Plotters are widely used in applications like CAD.

- ☐ Plotters print by moving one or more pen across the surface of a piece of paper. This means that plotters are restricted to line art, rather than raster graphics as with other printers.
- ☐ Pen plotters can draw complex line art, including text, but do so slowly because of the mechanical movement of the pens. They are often incapable of efficiently creating a solid region of colour, but can draw an area by drawing a number of close, regular lines.
- ☐ Plotters offered the fastest way to efficiently produce very large drawings or colour high-resolution vector-based artwork when computer memory was very expensive and processor power was very limited.
- ☐ There are a number of different types of plotters:
  - ☐ A **drum plotter** draws on paper wrapped around a drum which turns to produce one direction of the plot, while the pens move to provide the other direction.
  - ☐ A **flatbed plotter** draws on paper placed on a flat surface; and an electrostatic plotter draws on negatively charged paper with positively charged toner.
- ☐ Pen plotters have essentially become obsolete, and have been replaced by large-format inkjet printers and toner based printers.
- ☐ They are most frequently used for CAE (computer-aided engineering) applications, such as CAD (computer-aided design) and CAM (computer-aided manufacturing)

### **Difference between Raster Scan Display and Random Scan Display**

Raster Scan methods have increasingly become the dominant technology since about 1975. These methods use the TV type raster scan. The growth in the use of such methods has been dependent on rapidly decreasing memory prices and on the availability of cheap scan generating hardware from the TV industry.

## **DVST - Direct View Storage Tube**

Conceptually the Direct View Storage Tube (DVST) behaves like a CRT with highly persistent phosphor. Pictures drawn on there will be seen for several minutes (40-50 minutes) before fading. It is similar to CRT as far as the electronic gun and phosphor-coated mechanisms are concerned. But instead of the electron beam directly writing the pictures on the phosphor coated CRT screen, the writing is done with the help of a fine-mesh wire grid.

## **What is Color CRT Display? Explain Beam-penetration and Shadow-mask method**

This was one of the earlier CRTs to produce color displays. Coating phosphors of different compounds can produce different colored pictures. But the basic problem of graphics is not to produce a picture of a predetermined color, but to produce color pictures, with the color characteristics chosen at run time.

## **Plasma displays, Thin film electro-luminescent display, Light-emitting diode ( LED ), Liquid Crystal Display (LCD)**

Plasma displays are bright, have a wide color gamut, and can be produced in fairly large sizes, up to 262 cm (103 inches) diagonally. They have a very low-luminance "dark-room" black level, creating a black some find more desirable for watching movies. The display panel is only about 6 cm (2½ inches) thick, while the total thickness, including electronics, is less than 10 cm (4 inches).

## **How are refresh rates calculated? And What is the Importance of Refresh Rates**

Factors in determining refresh rates. A refresh rate is dependent upon a monitor's horizontal scanning frequency and the number of horizontal lines displayed. The horizontal scanning frequency is the number of times the electron beam sweeps one line and returns to the beginning of the next in one second. Horizontal scanning frequency is measured in kilohertz (kHz).

## **Cathode Ray Tube (CRT/Monitor)**

One of the basic and commonly used display devices is Cathode Ray Tube (CRT). A cathode ray tube is based on the simple concept that an electronic beam, when hits a phosphorescent surface, produces a beam of light (momentarily - though we later describe surfaces that produce light intensities lashing over a period of time).

## **What is pixel? Explain Pixel resolution**

A pixel (short for picture element, using the common abbreviation "pix" for "picture") is one of the many tiny dots that make up the representation of a picture in a computer's memory. Each such information element is not really a dot, nor a square, but an abstract sample.

### **Shadow Mask CRT (Cathode Ray Tube)**

In Shadow Mask CRT tiny holes in a metal plate separate the colored phosphors in the layer behind the front glass of the screen. The holes are placed in a manner ensuring that electrons from each of the tube's three cathode guns reach only the appropriately-colored phosphors on the display. All three beams pass through the same holes in the mask, but the angle of approach is different for each gun.

### **What is LCD (Liquid Crystal Display)?**

**LCD** stands for liquid crystal display. Your *digital* watch uses an LCD to show you the time, and most portable computers use an LCD to display the screen. There is actually a liquid compound, liquid crystals, sandwiched between two grids of electrodes. The electrodes can selectively turn on the different cells or *pixels* in the grid to create the image you see.

### **Magnet-Optical Storage Media**

There are used for erasable disks. MO system includes basic principles of both magnetic & optical storage systems. MO systems write magnetically & read optically. It has two standard forms: **5.25 inches & 3.5 inches**.

### **What is Monitor:-**

**Monitor** is another word for the computer *screen*. But "monitor" encompasses the whole piece of equipment, rather than just the screen part that you look at. You also might hear a monitor called a *display*, as in "Oooh, I got a new two-page display," or *VDT*(*video* display terminal), as in newspaper journalism, or *CRT* (cathode ray tube), which is the technical term for a picture tube. However, flat panel screens like *LCDS* are not referred to as monitors, even if they're housed externally from a computer.

### **What is CGA (Color Graphics Adapter)?**

**CGA** stands for color graphics adapter, the first IBM video *card* to permit graphics on the screen. We're lucky they've come out with better models, because CGA graphics are gawdawful crude. With a CGA, your screen can show up to 640 dots across by 200 dots up and down, with only one color. Even at that maximum *resolution*, pictures look really blocky and out of proportion. Pictures will look even more blocky if you want 4 colors on the screen at once, since you're then limited to 320 dots across and 200 down. If you can tolerate a totally chunky display of 160 by 200 dots, you can get a maximum of 16 colors on a CGA. Wow!

### **Explain vector vs. raster graphics.**

**Vector graphics** are stored in the computer as a set of mathematical formulas describing the shapes that make up each image. When you display a vector graphic on the screen or print it, these formulas are converted into the patterns of dots you can see. Because the dots are not specified unit! you display or print the graphic, you can change the size of the image without any loss of quality, and the image will always appear at the highest *resolution* of whatever screen or printer you're using. The term vector graphics means exactly the same thing as *object-oriented* (or just object) graphics.

### **What is interlaced or Non-Interlaced Monitors?**

In a standard television-like computer *monitor*, an image is produced on the screen by a beam of electrons sweeping rapidly across the surface of the picture tube, lighting up the screen as it passes. Starting at the top, the beam traces one horizontal row across the screen, shifts down a bit and does another row, and so on, until the full height of the screen has been covered.

### **What is LED (light emitting diode)?**

**LED** stands for light emitting diode. You know those little lights on your computer, usually near the hard disk, that flash while the computer is working? Those are LEDs. They work on the principle of electroluminescence, which refers to substances that glow when you apply electricity. LEDs were used in digital watches, but now all digital watches use *LCDs* because LCD stakes less power.

### **What is VGA (video graphics array)?**

**VGA**, which stands for video graphics array, is currently the most popular standard for PC screen display equipment. Technically, a VGA is a type of *video adapter* (circuitry in the computer that controls the screen). IBM developed the VGA for its PS/2 line of computers (the name "Video Graphics Array" is an IBM trademark), but loads of other manufacturers make *VGA add-in boards* (that plug into a slot in the pc) and *VGA chips* (in some pcs, these VGA chips are built right into the main part of the computer, the *motherboard*). A *VGA monitor* is a monitor that works with a VGA adapter.

### **What is Multiscan**

**Multiscan** refers to a type of computer *monitor* that automatically matches the synchronizing signals sent from the computer's video adapter (the video circuitry). On a standard television-type monitor, the image you see is formed by a single beam of electrons scanning lickety-split across the picture tube. The beam starts at one corner, traces a narrow horizontal line, then moves down a bit and traces the next line. The speed with which the beam travels horizontally and vertically (the horizontal and



vertical "scan frequencies"), must match the synchronizing signals from the computer's video circuits.

### What is Dot Pitch?

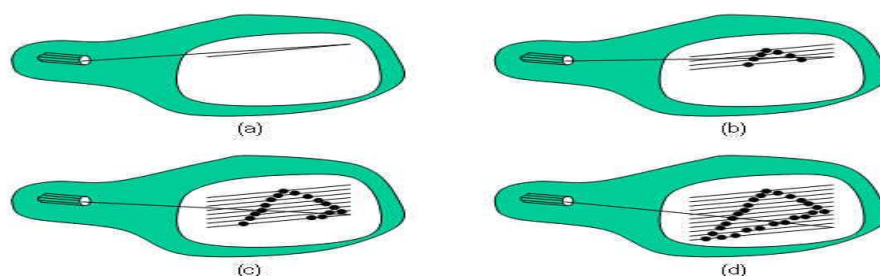
The dot pitch of a color monitor measures the size of the tiny individual dots of phosphorescent material that coat the back side of the picture tube's face. The dot pitch helps determine how sharp the image looks, independent of the *resolution* (which is measured in *pixels*). A smaller dot pitch is better.

### What is Low Resolution?

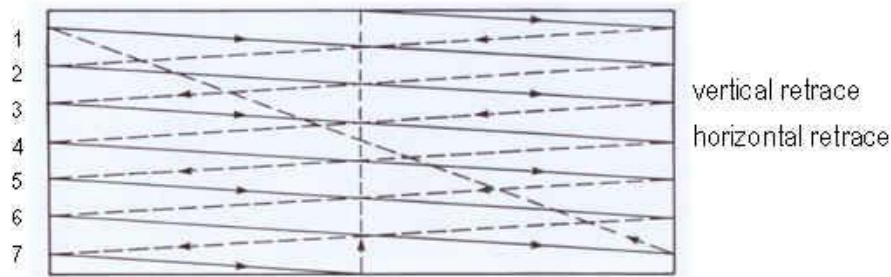
If an image is displayed on your screen or printed on the page in **low-res** (short for **low resolution**), that means you are seeing a low-grade quality. Some graphics are just low-resolution to begin with, such as graphics made in the *paint file format* at 72 dots per inch. Some graphics are created as complex, *high-resolution* images, but you may choose to display them on the screen or print them in low-res just to save time, since it takes longer for a screen or a printer to create the high- **resolution** version.

### Raster Scan Systems:-

It is the most common type of graphics monitor based on television technology. In a raster scan system, the electron beam is swept across the screen, one row at a time from top to bottom. When electron beam moves across each row the beam intensity is turned ON and OFF to create a pattern of illuminated spots. Picture definition is stored in a memory called frame buffer which holds the set of intensity values, which are then retrieved from the frame buffer and pointed on the screen one row at a time as shown in figure below:



At the end of each line the beam must be turned off and redirect to the left hand side of the CRT, this is called Horizontal Retrace. At the end of each frame, the electron beam return to top left corner of the screen to begin the next frame called Vertical Retrace as shown in figure below:



### Advantages

- produce realistic images
- also produced different colors
- and shadows scenes.

### Disadvantages

- low resolution
- expensive
- electron beam directed to whole screen

A video controller, often referred to as a video or graphics card, is a key hardware component that allows computers to generate graphic information to any video display devices, such as a monitor or projector. They are also known as graphics or video adapters. Some modern computers do not include video cards, but rather have graphics processing units directly integrated into the computer's motherboard.

### Display Processors:-

Interactive graphics systems typically employ several processing units. In addition to CPU, a special purpose display processor is used to interact with the CPU and to control the operation of the display device as shown in figure below:

The display processor is used to convert digital information from the CPU into analog value needed by the display device. This digital-analog conversion depends on the type of display devices used and the particular graphics functions that are to be hardware implemented. For many graphics monitors, the coordinate origin is defined at the lower left screen corner as shown in figure below: In the figure, the X and Y are used to store the coordinates of the screen pixels. First, set X to 0 and Y to

Ymax. This value is used to set intensity of the CRT beam. Then X is incremented by 1 and process is repeated for next pixel along the scan line. When the last pixel has been processed, then X is reset to 0 and Y is decremented by 1. The screen must be refreshed at the rate of sixty frames per second. Following figure shows one way to set up the organization of a raster scan system containing a separate display processor sometimes called a display processor. The purpose of display processor is to free the CPU from the graphics chores. In this, a separate display processor memory area can also be available. A major task of display processor is to perform a process called scan conversion. It is the process of separating contiguous graphics objects as a collection of ellipses, rectangles and polygons.

Example: Characters can be defined with rectangle grids or with curved outlines as shown in figure below:

The display processor also designed to perform a number of additional operations. These functions include various line style (dashed, dotted, solid), displaying color area and so on. Also designed to interface with interactive input devices such as mouse. Sometimes the display processor in a random graphics system is referred to as a display processing unit or a graphic controller. The following figure shows the set up of organization of random graphics systems.

Computer graphics tells us that what the actual workings of graphics are. Computer Graphics remains one of the most existing and repladly growing computer field. Computer Graphics as the pictorial representation or graphical representation of a computer

---

### **Application of computer graphics:**

Computer – Aided Design

Presentation Graphics

Computer Art

Entertainment

Education and Training

Graphics provides one of the most natural means of communicating with a computer.

Interactive computer graphics permits extensive, high-bandwidth user-computer interaction.

---

## **Representative Uses of Computer Graphics**

Computer graphics is used today in many different areas of industry, business, government, education, and entertainment.

- User interfaces: GUI, etc.
- Business, science and technology: histograms, bar and pie charts, etc.
- Office automation and electronic publishing: text, tables, graphs, hypermedia systems, etc.
- Computer-aided design (CAD): structures of building, automobile bodies, etc.
- Simulation and animation for scientific visualization and entertainment: flight simulation, games, movies, virtual reality, etc.
- Art and commerce : terminals in public places such as museums, etc.
- Cartography: map making
- Simulation and animation for scientific visualization and entertainment.
- Multimedia textbooks

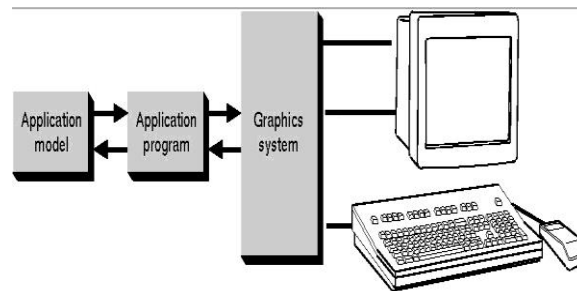
## **Classification of Applications**

- **Paint programs:** Allow you to create rough freehand drawings.
- **Animation software:** Enables you to chain and sequence a series of images to simulate movement. Each image is like a frame in a movie
- **CAD software:** Enables architects and engineers to draft designs. It is the acronym for computer-aided design. A CAD system is a combination of hardware and software that enables engineers and architects to design everything from furniture to airplanes.
- **Desktop publishing:** Provides a full set of word-processing features as well as fine control over placement of text and graphics, so that you can create newsletters, advertisements, books, and other types of documents
- **business software:** enables users to create highly stylized images for slide shows and reports. The software includes functions for creating various types of charts and graphs and for inserting text in a variety of fonts.

---

## **Conceptual Framework for Interactive Graphics.**

The high-level conceptual framework shown here can be used to describe almost any interactive graphics system



The three major parts of the framework are:

Application Modeling

Calculating what is to be displayed

Displaying the Model

Calling the graphics API routines

Interaction Handling

Handling user interaction, which will change the model, and therefore the display.

Typically an event driven loop

### Scan Converting Lines

Converting the geometric definition of a primitive form into a set of pixels that make up the primitive in the image space. This conversion task is scan conversion.

### Types of Scan Conversion

1. Digital Differential (DDA) Algorithm
2. Bresenham's Line Algorithm

**DDA algorithm** is an incremental scan conversion method.

- Incremental scan-conversion method

- Faster than the direct use of the line equation
- a floating point operation is still required
- The line drifts away from the original line when the line is relatively long

### AN ALGORITHM TO DRAW A LINE

Compute

$dx = x_2 - x_1$      $dy = y_2 - y_1$

If  $\text{abs}(dx) > \text{abs}(dy)$  then  $\text{steps} = \text{abs}(dx)$

Else  $\text{steps} = \text{abs}(dy)$

4        Plot a point at  $(x, y)$

$x_{inc} = dx / \text{steps};$

$y_{inc} = dy / \text{steps};$

$x = x_1$  and  $y = y_1$

Plot a point at  $(x, y)$

$k=1$

if  $k = \text{steps}$  , stop

$x = x + x_{inc}$

$y = y + y_{inc}$

Plot a point at  $(x, y)$

$k = k+1$

Go to step 7

### BRESENHAM LINE ALGORITHM

An accurate and efficient raster line generating algorithm, the **Bresenham's line-drawing algorithm**. This algorithm was developed by Jack E. Bresenham in 1962 at IBM.

1. Highly efficient incremental method

2. Produces mathematically correct results using simple calculations

*Bresenham's Line Drawing Algorithm for  $m < 1$  :*

*(1) Input the two line endpoints & store the left end point in  $(x_0, y_0)$ .*

*(2) Load  $(x_0, y_0)$  into frame buffer that is plot the first point.*

*(3) Calculate constants  $\Delta x$ ,  $\Delta y$ ,  $2\Delta y$  and  $2\Delta y - 2\Delta x$  and obtain the*

*starting value for the decision parameter as :  $P_0 = 2\Delta y - \Delta x$ .*

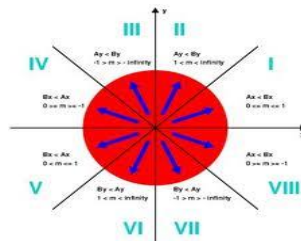
*(4) At each  $x_k$  along the line starting at  $k = 0$ , perform the following test if  $P_k < 0$  the next point to plot is  $(x_{k+1}, y_k)$  and  $P_{k+1} = P_k + 2\Delta y$*

*Otherwise the next point to plot is  $(x_{k+1}, y_{k+1})$  and  $P_{k+1} = P_k + 2\Delta y - 2\Delta x$ .*

*(5) Repeat step 4  $\Delta x$  times.*

### Scan Converting Circles

circle is a symmetrical figure , eight points can be plotted for each value that the algorithm calculates



A circle is a set of points that are at a given distance  $r$  from the center position  $(x_c, y_c)$ . This distance relationship is given as :

$(x - x_c)^2 + (y - y_c)^2 - r^2 = 0$  This equation is used to calculate the position of points along the circle path by moving in the  $x$  direction from  $(x_c - r)$  to  $(x_c + r)$  and determining the corresponding  $y$  values as :

$$y = y_c \pm \sqrt{r^2 - (x - x_c)^2}$$

**Algorithm :**

(1) Input radius  $r$  and circle center  $(x_c, y_c)$  and obtain the first point on

circumference of a circle centered on origin  $(x_0, y_0) = (0, r)$

(2) Calculate the initial value of the decision parameter as :  $P_0 = 5/4 - r$

(3) At each  $x_k$  position, starting at  $k = 0$  if  $P_k < 0$  the next point along the circle is  $(x_{k+1}, y_k)$  and  $P_{k+1} = P_k + 2x_{k+1} + 1$ , otherwise the next point along the circle is  $(x_k + 1, y_k - 1)$  and  $P_{k+1} = P_k + 2x_{k+1} + 1 -$

$2y_{k+1}$  where  $2x_{k+1} = 2x_k + 2$  &  $2y_{k+1} = 2y_k - 2$ .

(4) Determine symmetry points in other seven octants.

(5) Move each calculated pixel position  $(x, y)$  onto the circular path centered on  $(x_c, y_c)$  & plot coordinate values  $x = x + x_c$  &  $y = y + y_c$ .

(6) Repeat step (3) through (5) until  $x \geq y$ .

### Scan Converting Ellipses

An ellipse is defined as a set of points such that the sum of distances from two fixed points (foci) is same for all points given a point  $P = (x, y)$ , distances are  $d_1$  &  $d_2$ , equation is :

$$d_1 + d_2 = \text{constant}$$

In terms of local coordinates

$$F_1 = (x_1, y_1) \text{ \& \& } F_2 (x_2, y_2)$$

### Display Technologies

In some graphics systems a separate processor is used to interpret the commands in the display file.

Such a

#### Raster Display System:

- Interactive raster graphics employs several processing units
- Apart from CPU, a special purpose processor called video controller or display controller -> control operation of the display device
- Simple raster graphics system

#### Random Scan Display:

- The random scan display system with display processor.



- It except the frame buffer.
- In random scan display no local memory is provided for scan conversion algorithm.

### Video Controller:

- **Fixed area of system memory reserved for frame buffer**
- **Video controller given direct access to frame buffer to refresh the screen**
- **Coordinator origin is defined at lower left corner**
- **Scan lines labeled from  $y_{\max}$  at the top of the screen to 0 at the bottom**

### 1. Cohen- Sutherland Algorithm\_( PPT)

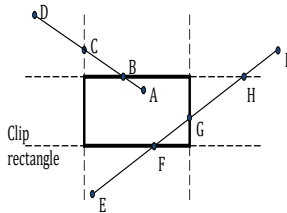
### Cohen-Sutherland Line Clipping in 2D

- ▶ Divide plane into 9 regions
- ▶ Compute the sign bit of 4 comparisons between a vertex and a **clip edge**
  - ▶  $(y_{\max} - y, y - y_{\min}, x_{\max} - x, x - x_{\min})$ , cast the results to 0 or 1
  - ▶ Point lines inside the region if all four bits are 0
- ▶ 4 bit **outcode** records results of four bounds tests:
  - ▶ 1<sup>st</sup> bit: outside halfplane of top edge (above top edge)
  - ▶ 2<sup>nd</sup> bit: outside halfplane of bottom edge
  - ▶ 3<sup>rd</sup> bit: outside halfplane of right edge
  - ▶ 4<sup>th</sup> bit: outside halfplane of left edge
- ▶ Compute outcodes for both vertices of the input edge, denoted  $OC_0$  and  $OC_1$
- ▶ If  $OC_0 = 0$  and  $OC_1 = 0$ , (i.e., outcode: 0000), then the input edge is trivially accepted
- ▶ Lines lying entirely in a particular halfplane can e trivially rejected. That is  $(OC_0 \text{ AND } OC_1) \neq 0$  (i.e., they share an "outside bit")

The diagram illustrates a 2D coordinate system with a central 'Clip Rectangle' defined by dashed lines. The plane is divided into nine regions by these lines. Each region is associated with a 4-bit outcode. The outcodes are: 1001 (top-left), 1000 (top), 1010 (top-right), 0001 (left), 0000 (inside), 0010 (right), 0101 (bottom-left), 0100 (bottom), and 0110 (bottom-right). Arrows point from the text 'Clip Rectangle' to the central rectangle.

15 - Clipping
6/22

## Cohen-Sutherland Algorithm

- ▶ If we can neither trivially accept or reject, then we do divide-and-conquer
  - ▶ Subdivide line into two segments and test again
- 
- ▶ Use a clip edge to cut line
  - ▶ Use outcodes to choose which edge is crossed
    - ▶ The bits that are different between outcodes will tell us which edge to examine
  - ▶ Pick an order for checking edges: top – bottom – right – left
  - ▶ Compute the intersection point
    - ▶ Clip edge will be axis-aligned, so we can fix either the x or the y
    - ▶ Can substitute into the line equation
  - ▶ Iterate for the newly created line segment, might need multiple passes (e.g., E-I at H)

15 - Clipping

7/22

## Cyrus-Beck Algorithm (PPT)

### Cyrus-Beck / Liang-Barsky Parametric Line Clipping

- ▶ Use the parametric line formulation:
  - ▶  $P(t) = P_0 + t(P_1 - P_0)$
- ▶ Determine if the line intersects with a clip line (both extended to infinity)
- ▶ Decide if the intersection actually occurs on the polygon
- ▶ This is a very similar strategy for intersection tests in ray-tracing

15 - Clipping

11/22

## Image Scanners

Image Processing is any form of signal, processing for which Input is an Image such as photographs or frames of video, the output of Image processing can be either an Image or a set of characteristics or parameters related to the Image.

## 2D Transformations

Moving of an object to one place in Window area to another place is called a Transformation.

Transformation is to change the object's

- Position (translation)
- Size (scaling)
- Orientation
- rotation)
- Shapes (shear)

**Rotation:**

- Rotation is applied to an object by repositioning it along a circular path in the XY plane
- Positive values of theta for counter clockwise rotation
- Negative values of theta for Clockwise rotation
- To generate a rotation , we specify

Rotation angle theta

Pivot point ( $X_r, Y_r$ )

**Scaling:**

- Scaling alters the size of an object .
- Uniform scaling means this scalar is the same for all components.
- Non –Uniform scaling different per component
- Operation can be carried out by multiplying each of it component by a scalar

**Reflection:** A reflection is a transformation that produces a mirror image of an object

- Reflection along x axis
- Reflection along y axis
- Reflection relative to an axis perpendicular to the xy plane and passing through the coordinate origin
- Reflection of an object relative to

**Shearing:** A transformation that distorts the shape of an object such that the transformed object appears as if the object were composed of internal layers that had been caused to slide over each other.

**PPT**

## 2D Transformation

### • Translation

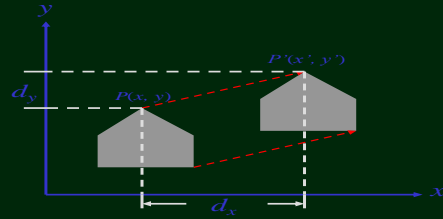
$P(x, y)$  move to  $P'(x', y')$

$$x' = x + d_x$$

$$y' = y + d_y$$

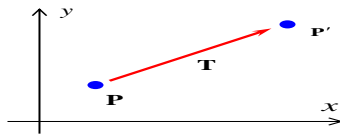
$$P = \begin{bmatrix} x \\ y \end{bmatrix}, P' = \begin{bmatrix} x' \\ y' \end{bmatrix}, T = \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

$$P' = P + T(d_x, d_y)$$



41

### 2D Translation



$$x' = x + t_x, y' = y + t_y$$

$$P = \begin{bmatrix} x \\ y \end{bmatrix}, P' = \begin{bmatrix} x' \\ y' \end{bmatrix}, T = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$$P' = P + T$$

April 2010

2

## 2D Transformation

### • Scaling

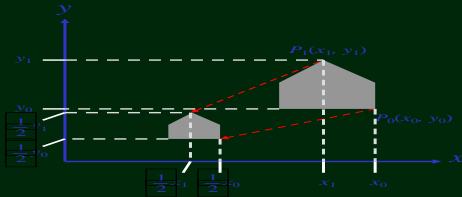
Scale by  $s_x$  along the  $x$  axis  
and by  $s_y$  along the  $y$  axis

$$x' = s_x \cdot x$$

$$y' = s_y \cdot y$$

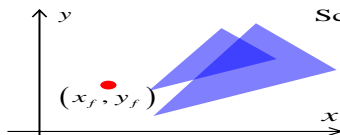
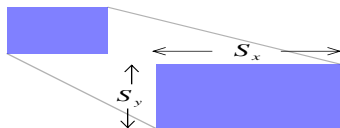
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

$$P' = S(s_x, s_y) \cdot P$$



42

### 2D Scaling



Scaling about a fixed point  $(x_f, y_f)$

$$x' = x \cdot s_x, y' = y \cdot s_y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$P' = S \cdot P$$

$$x' = x \cdot s_x + x_f (1 - s_x)$$

$$y' = y \cdot s_y + y_f (1 - s_y)$$

$$P' = P \cdot S + P_f \cdot (1 - S)$$

April 2010

4

## 2D Transformation

### • Rotation

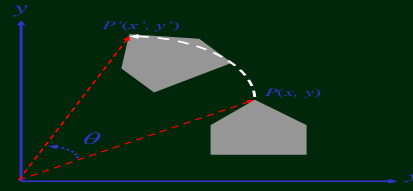
Rotate through an angle  $\theta$  about the origin

$$x' = x \cdot \cos \theta - y \cdot \sin \theta$$

$$y' = x \cdot \sin \theta + y \cdot \cos \theta$$

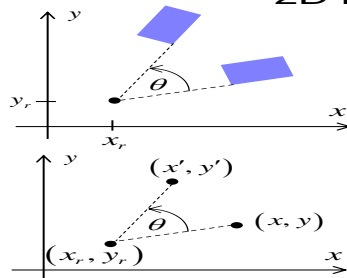
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\mathbf{P}' = \mathbf{R}(\theta) \cdot \mathbf{P}$$



4.3

### 2D Rotation



April 2010

Rotation in angle  $\theta$  about a pivot (rotation) point  $(x_r, y_r)$ .

$$x' = x_r + (x - x_r) \cos \theta - (y - y_r) \sin \theta$$

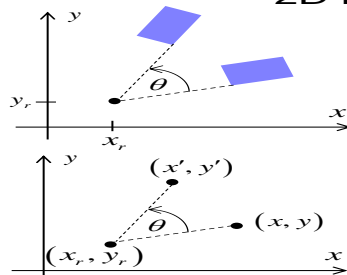
$$y' = y_r + (x - x_r) \sin \theta + (y - y_r) \cos \theta$$

$$\mathbf{P}' = \mathbf{P}_r + \mathbf{R} \cdot (\mathbf{P} - \mathbf{P}_r)$$

$$\mathbf{R} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$$

3

### 2D Rotation



April 2010

Rotation in angle  $\theta$  about a pivot (rotation) point  $(x_r, y_r)$ .

$$x' = x_r + (x - x_r) \cos \theta - (y - y_r) \sin \theta$$

$$y' = y_r + (x - x_r) \sin \theta + (y - y_r) \cos \theta$$

$$\mathbf{P}' = \mathbf{P}_r + \mathbf{R} \cdot (\mathbf{P} - \mathbf{P}_r)$$

$$\mathbf{R} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$$

3

## 2D Transformation

### • Derivation of the rotation equation

$$x = r \cdot \cos \phi$$

$$y = r \cdot \sin \phi$$

$$x' = r \cdot \cos(\phi + \theta)$$

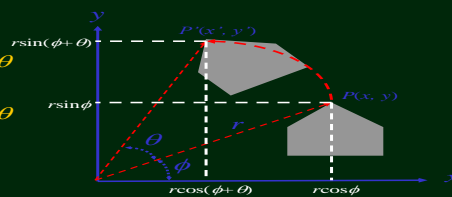
$$= r \cdot \cos \phi \cdot \cos \theta - r \cdot \sin \phi \cdot \sin \theta$$

$$y' = r \cdot \sin(\phi + \theta)$$

$$= r \cdot \cos \phi \cdot \sin \theta + r \cdot \sin \phi \cdot \cos \theta$$

$$x' = x \cdot \cos \theta - y \cdot \sin \theta$$

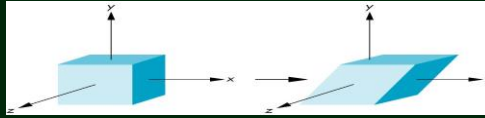
$$y' = x \cdot \sin \theta + y \cdot \cos \theta$$



4.4

## Shear

- Helpful to add one more basic transformation
- Equivalent to pulling faces in opposite directions



Angel: Interactive Computer Graphics 3E © Addison-Wesley 2002

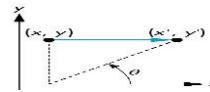
60

## Shear Matrix

Consider simple shear along  $x$  axis

$$\begin{aligned}x' &= x + y \cot \theta \\y' &= y \\z' &= z\end{aligned}$$

$$\mathbf{H}(\theta) = \begin{bmatrix} 1 & \cot \theta & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Angel: Interactive Computer Graphics 3E © Addison-Wesley 2002

61

## Homogeneous Coordinates and Matrix Representation of 2D Transformations, PPT

### Homogeneous Coordinates

The general form of four dimensional homogeneous coordinates is

$$\mathbf{p} = [x \ y \ z \ w]^T$$

We return to a three dimensional point (for  $w \neq 0$ ) by

$$x \leftarrow x/w$$

$$y \leftarrow y/w$$

$$z \leftarrow z/w$$

If  $w=0$ , the representation is that of a vector

Note that homogeneous coordinates replaces points in three dimensions by lines through the origin in four dimensions

Angel: Interactive Computer Graphics 3E © Addison-Wesley 2002

28

## Homogeneous Coordinates and Computer Graphics

- Homogeneous coordinates are key to all computer graphics systems
  - All standard transformations (rotation, translation, scaling) can be implemented by matrix multiplications with 4 x 4 matrices
  - Hardware pipeline works with 4 dimensional representations
  - For orthographic viewing, we can maintain  $w=0$  for vectors and  $w=1$  for points
  - For perspective we need a *perspective division*

Angel: Interactive Computer Graphics 3E © Addison-Wesley 2002

29

### Composite Transformations

1. Sequence of composite transformation matrix and transformations could be setup by the matrix product of the individual transformations
2. Also as Concatenation or Composition of Matrices

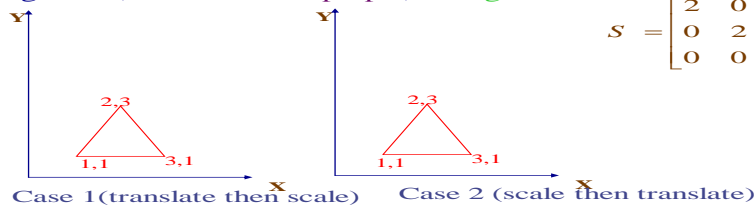
#### Compositing Transformations

- Does order matter?

- Case 1: translate by  $(-2, 0)$ , scale by  $(2, 2)$
- Case 2: scale by  $(2, 2)$ , translate by  $(-2, 0)$
- Begin: red, 1st transform: purple, 2nd: green

$$T = \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$S = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



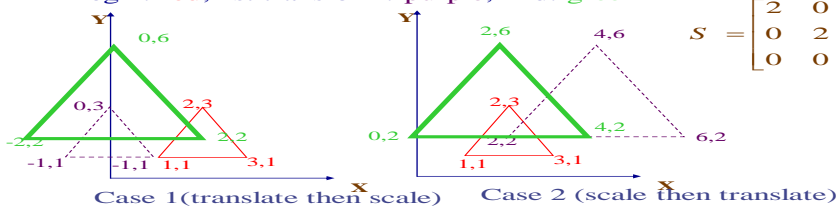
#### Compositing Transformations

- Does order matter?

- Case 1: translate by  $(-2, 0)$ , scale by  $(2, 2)$
- Case 2: scale by  $(2, 2)$ , translate by  $(-2, 0)$
- Begin: red, 1st transform: purple, 2nd: green

$$T = \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$S = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



## Composition Example

$$P' = STP \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & -4 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Scale(2.0,2.0);  
Translate(-2.0,0.0);  
drawTriangle();

$$P' = TSP \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & -2 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Translate(-2.0,0.0);  
Scale(2.0,2.0);  
drawTriangle();

In general, transformations are not commutative

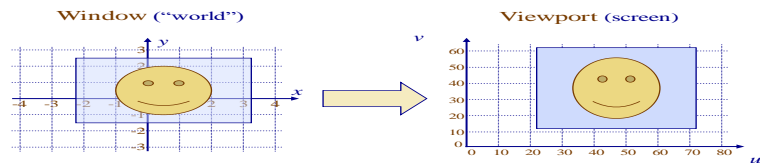
## Window-to-Viewport Transformation

Window area on which object to be display and view port is generated in the window area and finally display in window area after selection of a particular object that should be consider as in view port.

### Window-to-Viewport Transform

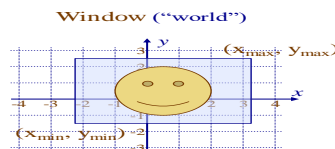
Need to transform points from “world” view (*window*) to the screen view (*viewport*)

- Maintain relative placement of points (usually)
- Can be done with a translate-scale-translate sequence



### Window

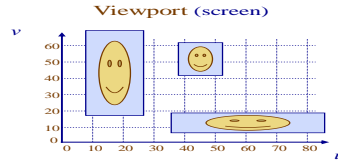
- “Window” refers to the area in “world space” or “world coordinates” that you wish to project onto the screen
- Location, units, size, etc. are all determined by the application, and are convenient for that application
- Units could be inches, feet, meters, kilometers, light years, etc.
- The window is often centered around the origin, but need not be
- Specified as (x,y) coordinates





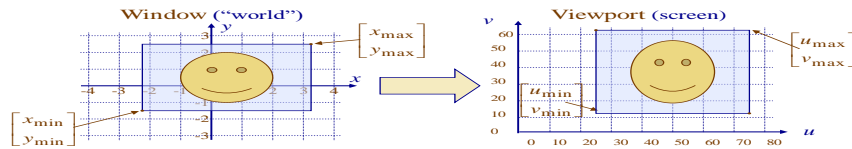
## Viewport (cont)

- You can have multiple viewports
  - They can contain the same view of a window, different views of the same window, or different views of different windows



## Window-to-Viewport Transform (cont.)

- The window-to-viewport transform is:
  - Translate lower-left corner of window to origin  
 $t_x = -x_{\min}$  and  $t_y = -y_{\min}$
  - Scale width and height of window to match viewport's  
 $s_x = \frac{u_{\max} - u_{\min}}{x_{\max} - x_{\min}}$  and  $s_y = \frac{v_{\max} - v_{\min}}{y_{\max} - y_{\min}}$
  - Translate corner at origin to lower-left corner in viewport  
 $t_x = u_{\min}$  and  $t_y = v_{\min}$



## Window-to-Viewport Transform (cont.)

- The final window-to-viewport transform is:

$$M_{WV} = T(u_{\min}, v_{\min}) \bullet S \left( \frac{u_{\max} - u_{\min}}{x_{\max} - x_{\min}}, \frac{v_{\max} - v_{\min}}{y_{\max} - y_{\min}} \right) \bullet T(-x_{\min}, -y_{\min})$$

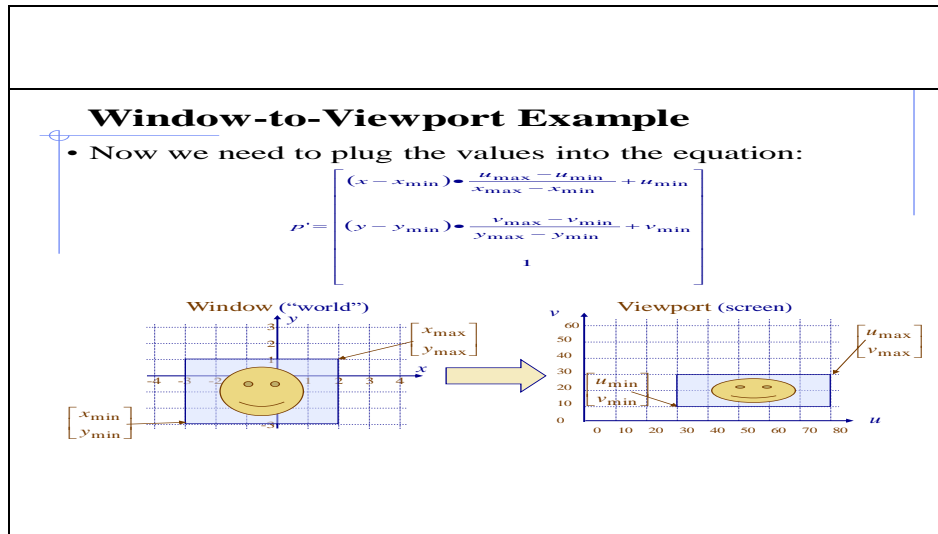
$$= \begin{bmatrix} 1 & 0 & u_{\min} \\ 0 & 1 & v_{\min} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{u_{\max} - u_{\min}}{x_{\max} - x_{\min}} & 0 & 0 \\ 0 & \frac{v_{\max} - v_{\min}}{y_{\max} - y_{\min}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_{\min} \\ 0 & 1 & -y_{\min} \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{u_{\max} - u_{\min}}{x_{\max} - x_{\min}} & 0 & -x_{\min} \bullet \frac{u_{\max} - u_{\min}}{x_{\max} - x_{\min}} + u_{\min} \\ 0 & \frac{v_{\max} - v_{\min}}{y_{\max} - y_{\min}} & -y_{\min} \bullet \frac{v_{\max} - v_{\min}}{y_{\max} - y_{\min}} + v_{\min} \\ 0 & 0 & 1 \end{bmatrix}$$

## Window-to-Viewport Transform (cont.)

- Multiplying the matrix  $M_{WV}$  by the point  $p$  gives:

$$p' = \begin{bmatrix} (x - x_{\min}) \bullet \frac{u_{\max} - u_{\min}}{x_{\max} - x_{\min}} + u_{\min} \\ (y - y_{\min}) \bullet \frac{v_{\max} - v_{\min}}{y_{\max} - y_{\min}} + v_{\min} \\ 1 \end{bmatrix}$$



## UNIT – III

### Representing Curves & Surfaces

Polygon Meshes, Parametric Cubic Curves

### Solid Modeling

Representing Solids, Regularized Boolean Set Operations, Primitive Instancing, Sweep Representations, Boundary Representations, Spatial Partitioning Representations, Constructive Solid Geometry, Comparison of Representations, User Interfaces for Solid Modeling.

### polygon mesh

- it is a collection of vertices
- it is collection of edges and
- collection of faces
- it is a large sub-field of computer graphics and geometric modeling
- operations performed on meshes may include Boolean logic, smoothing, simplification, and many others

### Parametric Cubic Curves

Curves and surfaces can have explicit, implicit, and parametric representations. Parametric representations are the most common in computer graphics.

- A parametric cubic curve is to be fitted to interpolate four points. The first and last points are to be at  $u=0$  and  $u=1$ . The other two points are at  $u=1/3$  and  $u=2/3$ , respectively. Find the equation of the curve in the form  $\mathbf{P}(u)=\mathbf{U}^T [\mathbf{M}_p] \mathbf{B}$ .
- The geometric matrix  $G$  of a parametric cubic curve defines a straight line
- A Bezier cubic curve obtained by a set of points.
- A set of control points explain what happen to a Bezier segment when two of the control points are coincident.

---

### **Solid Modeling**

- a) Primitive Instancing
  - It is set of Primitive 2D/3D solid shapes
  - Similar to parameterized object
  - A family with few difference in members
  - Relatively complex object
  - Without combining object
- b) Sweep Representations
  - Sweeping a object in 2D and 3D
  - Translational sweep
  - Rotational sweep
  - General sweep
- c) Boundary Representations
  - Object description in terms of vertices, faces and edges
  - Some b-reps are restricted to planer , polygon etc...
- d) Spatial Partitioning Representations
  - A solid is decomposed into a collection of adjoin nonintersecting solids
  - 1-Cell decomposition
  - 2-spatial occupancy enumeration
  - 3-Octress
  - 4- Binary space –portioning trees
  - Unambiguous but not necessary unique
- e) Constructive Solid Geometry

- Operators at the internal nodes and easy primitive at the leaves
- Not Unique
- Deleting and adding replacing –modifying subtree etc..

#### f) Comparison of Representations

- Accuracy
- Domain
- Uniqueness
- Closure
- Compactness and efficiency

---

### Projections

Projection is 'formed' on the **view plane** (planar geometric projection) rays (projectors) projected from the **center of projection** pass through each point of the models and intersect projection plane. Since everything is synthetic, the projection plane can be in front of the models, inside the models, or behind the models..

#### Parallel:

- center of projection infinitely far from view plane
- projectors will be parallel to each other
- need to define the **direction of projection** (vector)
- 2 sub-types
- orthographic - direction of projection is normal to view plane
- oblique - direction of projection not normal to view plane
- better for drafting / CAD applications

#### Orthographic projection (or orthogonal projection)

- Means of representing a three-dimensional object in two dimensions.
- It is a form of parallel projection, where all the projection lines are orthogonal to the projection plane, resulting in every plane of the scene appearing in affine transformation on the viewing surface.
- It is further divided into *multiview orthographic projections* and *axonometric projections* which type of projection is used depends on the needs of the user - whether the goal is the mathematically correct depiction of length and angles, or a realistic looking image of the object

#### Perspective:

- center of projection finitely far from view plane

- projectors will not be parallel to each other
- need to define the location of the **center of projection** (point)
- classified into 1, 2, or 3-point perspective
- more visually realistic - has perspective foreshortening (objects further away appear smaller)

---

### **Oblique projection**

- It is a simple type of technical drawing of graphical projection used for producing pictorial, two-dimensional images of three-dimensional objects.
- Types
- it projects an image by intersecting parallel rays (projectors)
- From the three-dimensional source object with the drawing surface (projection plane).

### **Hidden-Surface Removal**

- We now know which pixels contain which objects, however since some pixels may contain two or more objects we must calculate which of these objects is visible and which are hidden
  - Hidden surface removal is generally accomplished using the Z-buffer algorithm
  - In this algorithm, we set aside a two-dimensional array of memory (the Z-buffer) of the same size as the screen (#rows x #columns)
  - This is in addition to the buffer we will use to store the values of pixels which will be displayed (color values)
  - The Z-buffer will hold values which are depths (or z-values)
  - The buffer is initialized so that each element has the value of the far clipping plane (the largest possible z-value after clipping has been performed)
  - The other buffer is initialized so that each element contains a value which is the background color
  - Now for each polygon we have a set of pixel values which that polygon covers
  - For each one of these pixels, we compare its interpolated depth (z-value) with the value of the corresponding element already stored in the Z-buffer
  - If this value is less than the previously stored value, the pixel is nearer the viewer than previously encountered pixels
  - Replace the old value of the Z- buffer with the new, interpolated value and replace the old value of the other buffer with the value (color) of the pixel
  - Repeat for all polygons in the image
-

## **Z Buffer**

The easiest way to achieve hidden-surface removal is to use the depth buffer (sometimes called a z-buffer). A depth buffer works by associating a depth, or distance from the viewpoint, with each pixel on the window. Initially, the depth values for all pixels are set to the largest possible distance, and then the objects in the scene are drawn in any order. Graphical calculations in hardware or software convert each surface that's drawn to a set of pixels on the window where the surface will appear if it isn't obscured by something else. In addition, the distance from the eye is computed. With depth buffering enabled, before each pixel is drawn, a comparison is done with the depth value already stored at the pixel. If the new pixel is closer to the eye than what's there, the new pixel's colour and depth values replace those that are currently written into the pixel. If the new pixel's depth is greater than what's currently there, the new pixel would be obscured, and the colour and depth information for the incoming pixel is discarded.

## **Z Buffer in OpenGL**

To use depth buffering in OpenGL, you need to enable depth buffering. This has to be done only once. Each time you draw the scene, before drawing you need to clear the depth buffer and then draw the objects in the scene in any order.

## **Scan-Line Algorithm**

The scan-line algorithm is another image-space algorithm. It processes the image one scan-line at a time rather than one pixel at a time. By using area coherence of the polygon, the processing efficiency is improved over the pixel oriented method

---

## **Painter's algorithm**

The idea behind the Painter's algorithm is to draw polygons far away from the eye first, followed by drawing those that are close to the eye. Hidden surfaces will be written over in the image as the surfaces that obscure them are drawn. The concept is to map the objects of our scene from the world model to the screen somewhat like an artist creating an oil painting. First she paints the entire canvas with a background colour. Next, she adds the more distant objects such as mountains, fields, and trees. Finally, she creates the foreground with "near" objects to complete the painting. Our approach will be identical. First we sort the polygons according to their z-depth and then paint them to the screen, starting with the far faces and finishing with the near faces.

The algorithm initially sorts the faces in the object into back to front order. The faces are then scan converted in this order onto the screen. Thus a face near the front will obscure a face at the back by overwriting it at any points where their projections overlap. This accomplishes hidden-surface removal without any complex intersection calculations between the two projected faces. The algorithm is a hybrid algorithm in that it sorts in object space and does the final rendering in image space.

The basic algorithm :

## **E-COMMERCE (305)**

### **Unit - I**

The term **commerce** is define as trading of good & services or if **‘e’** for **‘electronic’** is added to this, the definition of **e – commerce** is defined as trading of goods, services, information or anything else of value between two entities over the internet.

Following are some definitions of **e – commerce**:-

1. It is the ability to conduct business electronically over the internet.
2. It means managing transactions using networking and electronic means.
3. It is a platform for selling products & services via internet.

#### **Characteristics of e – commerce:-**

1. Establishment of B to B relationship.
2. Electronic payment.
3. e – distribution of products & services.
4. Exchange of information.
5. Pre and post – sales support.
6. Customer relationship management.

#### **Advantage of e – commerce:-**

- 1. Facilitates the globalization of business:-** e – commerce facilitates the globalization of business by providing some economical access to distant markets and by supporting new opportunities for firms to increase economies by distributing their products internationally.
- 2. Provides increased purchasing opportunities for the buyer:-** As e – commerce increases sales opportunities for the seller, it also increases purchasing opportunities for buyer.

**3. Lowering staffing cost:-** As in e – commerce, the selling & purchasing process is outline, the amount of interaction with staff is minimized

**4. Market based expansion:** - An e – commerce is open to entirely new group of users, which include employees, customers, suppliers & business partners.

**5. Increased profits:-**With e – commerce, companies reach more & more customers where physical commerce cannot reach, thus increasing profits.

**6. Increased customer service & loyalty:** - e – commerce enables a company to be open for business wherever a customer needs it.

**7. Increase speed & accuracy:-** E – commerce see the speed and accuracy with which business can exchange information, which reduces cost on both sides of transactions. It is available 24 hours a day & 7 days a week.

**8. Reduction of paper storage.**

**9. Increased response times:-** In e – commerce, the interaction with the system take place in real time & therefore allows customer or bidder to respond more Quickly & thus reduces the time of discussion between then as in traditional commerce.

**Limitations of e – commerce:** -

**1. Security:** - the security risk in e – commerce can be- client / server risk data transfer and transaction risk virus risk

**2. High start up cost:-**

The various components of cost involved with e – commerce are:-

Connection: - connection cost to the internet.

Hardware / software: - this includes cost of sophisticated computer, modem, routers, etc.

Maintenance: - this includes cost involve in training of employees and maintenance of web-pages.

**3. Legal issues:** - these issues arise when the customer data is fall in the hands of strangers.

**4. Lack of skilled personnel:** - there is difficulty in finding skilled www developers and knowledgeable professionals to manage and a maintain customer on line.

**5. Loss of contact with customers:** - Sometimes customers feel that they have not received sufficient personal attention.

**6. Uncertainty and lack of information:-** most of the companies has never used any electronic means of communication with its customers as the internet is an unknown mode for them.

**7. Some business process may never be available to e – commerce:-**Some items such as foods, high cost items such as jewelry may be impossible to be available on the internet.



## **Electronic Commerce Model**

### **Types of e – commerce:-**

#### **1. Business to customer (B to C):-It means the consumer is motivated by business.**

Customer identifies a need

Searches for the product or services to satisfy the need

Select a vendor and negotiates a price

Receives the product or service

Makes payment

Gets service & warranty claims

### **B to C working**

1. Visiting the virtual mall- customer visits the mall by browsing the outline catalogue.
2. Customer registers- customer has to register to become part of the site's shopper registry
3. Customer buys product.
4. Merchant processes the order- the merchant then processes the order that is received from the previous stage & fills up the necessary forms
5. Credit card is processed: - credit card of the customer is authenticated through a payment gateway or a bank.
6. Shipment & delivery: - the product is then shipped to customer.
7. Customer receives: - the product is received by customer and is verified.
8. After sales service: - after sale, the firm wants to maintains a good relationship with its customers. It is called CRM customer relationship management.

**Business to business (B to B):-** this is called as a business motivated by another business.

### **B2B is classified as:-**

1. Market place: - a digital electronic market place where suppliers and commercial purchasers can conduct transactions.
2. E – distributors: - a company that suppliers products and services directly to individual business.
3. B2B service provider: - it is a company that aells access to internet based software application to another companies.

4. Infomediary: - a company whose business model is premised upon gathering information about customers & selling it to other businesses.

**Consumer to business (C to B):-** a business motivated by a customer.

**The various C2B classified into:-**

1. Idea collectors:- consumers generally have a great idea about how to improve the existing products and what new features can be added to new products.

2. Reverse auctions:- it allow prospective airline travelers to visit the website and name their price for travel between only pair of city. If an airline is willing to issue a ticket at there price, the passenger is obligated to buy.

3. Consumer to consumer (C to C):-

In this type, a consumer is motivated by another consumer. Consumers sells directly to other consumers via online classified ads and auctions, or by selling personal services or expertise online.

<b>Traditional Commerce</b>	<b>E- Commerce</b>
Customer can easily identify & authenticate a merchant by seeing directly to him.	It is not easy in this case.
Customers can directly talk to merchant. Communication in the hands of a third party.	Customer can only see the representation & can only is not see the webpages
Customers can interact with other customers and gain feed back about merchant from other customers	Customer cannot interact with other customers.
It is not available all the time.	It is always available 24* 7*365 hours.
It is slow method.	It is fast method.
Customers just give cash to merchant & there is no need to give their name or address. So there is no worry about personal information.	Customer have to give their personal information to purchase the product.

## **Unit – II**

### **HTML**

The Hypertext Mark-up Language (or HTML) is the language used to create documents for the World Wide Web. As the name implies it is a mark-up language - the original (ASCII) text is edited

and new (text) codes i.e. tags are added to indicate how (and where) the text should appear.

**HTML** is short for Hyper Text Markup Language.

**Hypertext:** Hypertext is simply a piece of text that works as a link.

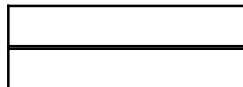
### **Features of HTML**

- Simple: It is simplest web developer tools. All the tags are self explanatory i.e. their meaning is clear from their name. For, example ‘B’ means bold, ‘I’ means italics and U means underline.
- Supports all types of documents: It can support all types of documents like text documents, sound and multimedia.
- Need no compilation: The languages like C/C++ requires compilation before doing anything with programs but HTML codes are automatically complied and results are displayed on screen.
- Supported by browser: All most all types of browsers supports the HTML code like Microsoft internet explorer ‘and Netscape navigation’ gold.
- Compatibility with other web tools: The code written in HTML can be pasted in other web tools and run efficiently without any change. For example HTML code can be pasted in Front page 2000, flash, dream weaver etc.
- Extendibility: adding features in it can enhance The HTML code. We can extend it using DHTML etc.
- Internet Base: It is the first web tool and all the Internet tools are based upon the idea of HTML.
- Developing intranet or extranet sites: HTML is commonly used to develop intranet and extranet sites, which are accessed by company people from one or more locations.
- Developing help files: It is used to develop on-line help files and help the developers to produce help documentation that is accessible from any computer and platform.
- **Developing network applications:** It is used to develop all types of applications such as training programs interactive charts or databases that are available,” from web.

### **Structure of an HTML Document**

An HTML document has two main parts: the head and the body. But firstly every HTML document should start by declaring that it is an HTML document. All normal web pages consist of a head and a body.

Head



## Body

The head is used for text and tags that do not show directly on the page. The body is used for text and tags that are shown directly on the page.

Finally, all web pages have an <html> tag at the beginning and the end, telling the browser where the document starts and where it stops.

<html>

<head> <!-- This section is for the title and technical info of the page--> </head> <body>

<!-- This section is for all that you want to show on the page. --> </body> </html>

These tags are of the form:

<html>: Should appear at the beginning of your document. </html>: Should appear at the end of your document.

A document consists of a head and a body. The head begins with <HEAD> and ends with </HEAD>. For the time being, the only thing that will appear in the head is the title of the document, which begins with <TITLE> and ends with </TITLE>. The title of the document appears at the top of the browser window.

The body, which begins with <BODY> and ends with </ BODY>, contains the text of the document that is displayed by the browser.

## Document Structure Format

<HTML> <HEAD>

<TITLE> Title Text Goes Here </TITLE> </HEAD> <BODY>

Body text goes here. </ BODY> </HTML> Explanation <HTML> <HEAD>

Every document begins with <HTML> and ends with </HTML> <TITLE>

<BODY>

## **HTML Tags:**

HTML codes are referred to as tags. Tags open with a less-than sign and close with a greater-than sign. The less-than and greater-than signs are called brackets.

<Tag>

- An HTML file is basically just a text file, which may contain references to other files. Angle brackets (<, >) are used to set off tags, which give information about the structure of the page. For example, the tag <p> indicates the beginning of a new paragraph.
- Tags are instructions that are embedded directly into the text of a HTML document.
- Each HTML tag describes that the browser should do something instead of simply displaying

Types of Tag: There are two kinds of HTML tags; paired and unpaired.

Paired Tag / Container Tag: A tag is said to be a paired tag if the text is placed between a tag and its companion tag. In paired tags, the first tag is referred to as **Opening Tag** and the second tag is referred to as **Closing Tag**.

Paired tags require an opening tag that turns a formatting feature on and a closing tag that turns the feature off. Paired tags must surround the text you want formatted with that feature.

For example, <title> and </title> define a page's title. You must include the slash in the closing tag in order for the pair to work. Always remember to close paired tags.

Example: <i> This text is in italics. </i>

Here <i> is called opening tag and </i> is called closing tag.

Paired / Container Tags	Function
<html>, </html>	Indicates the page is coded in HTML.
<head>, </head>,	Indicates part of the document that contains information about the page.
<title>, </title>	Always located in the head, the title provides the name of the document to be displayed by search engines and on browser title bars.

<body>, </body>	Indicates part of the document that contains formatted text, images, and links.
<h1>, </h1>	Level I heading; of the six heading levels, <h1> is the largest,

	<h2> is the next largest, etc.
<em>, </em>	Emphasis accentuates text, usually by italicizing it.
<strong>, </strong>	Strong stresses text, usually by making it bold.
<a href>, </a>	Links to another Web page or another location in the current Web page.
<a name>, </a>	Serves as an anchor for the <a href=> tag to link to. For more about <a name> and <a href=>
<blockquote>,</blockquote>	Indented margins; used for material quoted from an outside source.
<ol>,</ol>	Ordered list (1, 2, 3, 4....); used with the <li> tag.
<ul>, </ul>	Unordered list (bullet points); used with the <li> tag.
<li>,</li>	List item; indicates individual items in an ordered list (<ol>) or an unordered list (<ul>).
<p>, </p>	Paragraphs break; creates a blank line before subsequent text.

All paired tags in HTML must be properly nested. This means that every closing tag must match up with the nearest preceding unmatched paired tag. Unpaired Tag / Empty Tag: Unpaired tags work alone, and are usually placed before the text you want formatted. An unpaired tag does not have a companion tag. Unpaired tags are also known as Singular or Stand-Alone Tags. An unpaired tag sometimes also called empty — that is, they do not affect a block of the document in some way. An example is the <HR> element, which draws a horizontal line across the page. This element would simply be entered as

Example: <br> , <hr> etc. This tag does not require any companion tag.

The paired and unpaired tags are also called <tag\_on> that switches the tag sequence on. For example, to bold some text add a <strong> at the beginning of the text and </tagoff> that switches the tag sequence off. The tag\_on and tag\_off tags are the same except the off tag has an / character in front of it. For example, to switch off the bolding add a </strong> character sequence at the end of the text that is to be given the attribute of bolding.

Elements: Material between the tags is referred to as an element. An element is a fundamental component of the structure of a text document. Some examples of elements are heads, tables, paragraphs<sup>1</sup> and lists. Elements can contain plain text, other elements, or both.

The HTML instructions, along with the text to which the instructions apply, are called HTML elements.

Some elements may include an attribute, which is additional information that is included inside the start tag. For example, we can specify the alignment of images (top, middle, or bottom) by including the appropriate attribute with the image source HTML code.

### Upper and Lower Case

HTML is not case sensitive <title> is equivalent to <TITLE> or <TiTlE>. Element names are case insensitive. Thus, the the horizontal rule element can be written as any of <hr>, <Hr> or <HR>

Unpaired Tag / Empty Tag: Unpaired tags work alone, and are usually placed before the text you want formatted. An unpaired tag does not have a companion tag. Unpaired tags are also known as Singular or Stand-Alone Tags. An unpaired tag sometimes also called empty — that is, they do not affect a block of the document in some way. An example is the <HR> element, which draws a horizontal line across the page. This element would simply be entered as

Unpaired /Empty Tag	Function
<img>	Inserts an image into your text.
 	Lines break; moves subsequent text to the beginning of the line.
<hr>	Horizontal rule; draws a line across the width of the page.

The <BR> tag causes the browser to insert a line break (but not a blank line)

### Kinds of Text

<B> boldface text goes here </B> <I> italicized text goes here </I> <TT> typewritten text goes here </TT> <B>

The <B> and </B> tags surround text that is to be displayed using boldface <I>: The <I> and </I> tags surround text that is to be italicized.

#### Text Size

<BIG> big text goes here </BIG> <SMALL> small text goes here </SMALL> <BIG> <BIG> and </BIG> surround text that is to be made bigger than usual.

<SMALL>

<SMALL> and </SMALL> surround text that is to be made smaller than usual.

To create a superscript, we put a <SUP> tag at the beginning of the superscript and a </SUP> tag at the end. Similarly, to create a subscript put a <SUB> tag at the beginning of the subscript and a </SUB> tag at the end.

#### Superscript is Subscript

<SUP> superscript goes here </SUP> <SUB> subscript goes here </SUB>

#### Explanation

<SUP>

<SUP> and </SUP> surround text that is to be made into a superscript. <SUB>

<SUB> and </SUB> surround text that is to be made into a subscript.

### TEXT EFFECT TAGS

EFFECT	TAG	EFFECT	TAG
Bold	<B>Bold</B>	Strong emphasis	<STRONG>Strong emphasis</STRONG>
Italic	<I>Italic</I>	Emphasis	<EM>Emphasis</EM>
Keyboard text	<KBD>Keyboard text</BD>	Typewriter text	<TT>Typewriter text</TT>
A Subscript	A<SUB>Subscript</SUB>	ASUPLPT	A<SUP>Superscript</SUP>
Blink	<BLINK>Blink</BLINK>		



## Headings

HTML supports 6 levels of headings (numbered, surprisingly, 1 to 6). A heading level is declared by using the following tags:

`<number>` Should appear at the end of your heading.

Where number is the number of the required heading level.

### Examples of heading settings

Text marked as follows:

`<h1>`Heading level 1`</h1>` `<h2>`Heading level 2`</h2>` `<h3>`Heading level 3`</h3>` `<h4>`Heading level 4`</h4>` `<h5>`Heading level 5`</h5>` `<h6>`Heading level 6`</h6>`

Would appear in the following styles:

Heading level 1

Heading level 2

Heading level 3

Heading level 4

Heading level 5

Heading level 6

Tags and Attributes	Description
<code>&lt;DIV&gt;...&lt;/DIV&gt;</code>	A region of text to be formatted.
<code>ALIGN="..."</code>	Align text to CENTER, LEFT, or used with <code>&lt;P&gt;</code> , <code>&lt;H1&gt;</code> , <code>&lt;H2&gt;</code> , <code>&lt;H3&gt;</code> , etc.)
<code>&lt;OL&gt;...&lt;/OL&gt;</code>	An ordered (numbered) list.
<code>TYPE="..."</code>	The type of numerals used to label the list. Possible values are A, a, I, i, 1.
<code>START="..."</code>	The value with which to start this list.
<code>&lt;UL&gt;...&lt;/UL&gt;</code>	An unordered (bulleted) list.

TYPE="..."	The bullet dingbat used to mark list items. Possible values are DISC, CIRCLE, and SQUARE.
<LI>	A list item for use with <OL> or <UL>.
TYPE="..."	The type of bullet or number used to label this item. Possible values are DISC, CIRCLE, SQUARE, A, a, I, i, 1
VALUE="..."	The numeric value this list item should have (affects this item and all below it in <OL> lists).
<DL>...</DL>	A definition list. (May also be used for any kind of indentation.)
<dt>	A definition term, as part of a definition list.
<DD>	The corresponding definition to a definition term, as part of a definition list.

### Advanced Text, Formatting and Links

Tags and Attributes	Description
<EM>...</EM>	Emphasis (usually italic).
<STRONG>...</STRONG>	Stronger emphasis (usually bold).
<B>...</B>	Boldface text.
<I>.. </I>	Italic text.
<TT> </TT>	Typewriter (mono spaced) font

<PRE>...</PRE>	Preformatted text (Exact line endings and spacing will be preserved. Usually rendered in a mono spaced font)
<BIG>...</BIG>	Text is slightly larger than normal.
<SMALL>...</SMALL>	Text is slightly smaller than normal.
<SUB>...</SUB>	Superscript.
<SUP>...</SUP>	Subscript.
<STRIKE>..</STRIKE>	Puts a strikethrough line in text.
<FONT>...</FONT>	Controls the appearance of the enclosed text.
SIZE="..."	The Can size Also of Be the font, Specified from 1 to 7. Default is 3. As a value relative to the current size; for example, +2 Or -1.
COLOR="..."	Changes the color of the text.
FACE"..."	Name of font to use if it can be found on user's system. Multiple font names can separate by commas, and the first font on list that can be found will be used.
<BASEFONT>	Sets the default size of the font for the current page.
SIZE"..."	The default size of the font, from 1 to 7.
<A>...</A>	With the HREF attribute, creates a link to another document or anchor; with the NAME attribute, creates an anchor that can be linked to.
...HREF="..."	The address of the document and/or anchor point to link to
NAME="..."	The name for this anchor point in the document.

--	--

## Paragraphs

HTML will treat all text as one paragraph unless paragraph markers are added around each physical paragraph markers. The following codes are used for paragraph markers

`<p>` ----- this marker should be type at the beginning of every paragraph

`</p>` ----- this marker should be typed at the end of every paragraph

## ADDRESS Element

The ADDRESS element is used for address information, signatures statements of authorship, etc. It is often placed at the bottom (or top) of a document. The rendering of the contents of the ADDRESS is left up to the browser — most browsers render the ADDRESS in italics. It may also be right justified<sup>1</sup> or indented.

The BLOCK QUOTATION element defines a block quotation of text. A typical browser will render this in an appropriate way, for example by slightly indenting the text, or by italicizing it. BLOCKQUOTE also causes a paragraph break, and typically forces white space both before and after the quotation.

BLOCKQUOTE can contain paragraphs and most standard markup.

In HTML 3.2 and up, BLOCKQUOTE can take the ALIGN attribute to specify how text inside BLOCKQUOTE should be aligned. The supported values are

1. ALIGN="center" (center alignment)
2. ALIGN="left" (left alignment — the default)
3. ALIGN="right" (right alignment)
4. ALIGN="justify" (left-right justification). Justify alignment is not widely supported.

## Preformatted Text

The earlier versions of HTML did not support tables and , HTML viewers will disregard any layout that we placed in an HTML document. If an HTML document requires the presentation of text say, in a tabular format, then the only way that it can be viewed is to use HTML's preformatted codes. To keep the layout of text in an HTML document use the following codes: Must be added before the first character of the preformatted text.

`<pre>`

`</pre>`

HTML supports 3 types of lists

### 1. An Unordered List:

The term “unordered list” may be a bit unfamiliar to us, but odds are we’ve heard of the “bullet list.” That’s exactly what an unordered list is — a list of items, each one preceded by a “bullet” (a distinctive character; typically, a small black circle).

The list begins and ends with the tags `<UL>` and `</UL>` respectively. Each item in the list is marked using the `<LI>` tag, which stands for “List Item.” `<LI>` has a corresponding `</LI>`, but this closing tag is not required to end the list item (although you could use one if you really wanted to). You can use as many list items as you like, up to your browser’s built-in maximum, if any.

Here’s the markup for a simple list: `<UL>`

```
<LI> Monday <LI>Tuesday <LI> Wednesday <LI>Thursday <LI>Friday </ul>
```

If we loaded an HTML page containing the markup above, we would see the days of the week, each one preceded by a “bullet.”

### 2. An Ordered List

In order list instead of writing `<ul>`. We will write `<OL>` for order list and then the days of the week comes in numeric format.

### 3. A Definition List (or Glossary List)

Definition list `<dl>` and `</dl>`

Definition Lists

Ordered lists are as nestable as unordered lists, and we can nest unordered lists in ordered lists, as well as the other way around.

```
<DL>
```

Definition lists begin and end with the tags `<DL>` and `</DL>`. However, unlike the unordered and ordered lists, definition lists are not based on the items. They are instead based on term-definition pairs.

Example of a Definition List

- A **Definition List** has two parts: an element part and an element definition part. These are distinguished by the following HTML codes:

```
<dt>
```

This must appear at the beginning of the definition list’s elements and a corresponding `</ dt>` must be added after the last character of the list element.

This must appear at the beginning of each element’s definition and a corresponding `</dd>` must be added after the last character of the definition.

For example, the following HTML code will produce a definition list: <dl>

<dt>Item 1</dt>

<dd>Text relating to item 1</dd> <dd>

<dt>Item 2</dt>

<Dd>Text relating to item 2</dd>

This will have the following appearance: Item I

Text relating to item 1 Item 2

Text relating to item 2

### HTML Table

**HTML table tag** is used to display data in tabular form (row \* column). There can be many columns in a row.

HTML tables are used to manage the layout of the page e.g. header section, navigation bar, body content, footer section etc. But it is recommended to use div tag over table to manage the layout of the page .

### HTML Table Tags

Tag	Description
<table>	It defines a table.
<tr>	It defines a row in a table.
<th>	It defines a header cell in a table.
<td>	It defines a cell in a table.
<caption>	It defines the table caption.
<colgroup>	It specifies a group of one or more columns in a table for formatting.
<col>	It is used with <colgroup> element to specify column properties for each column.
<tbody>	It is used to group the body content in a table.
<thead>	It is used to group the header content in a table.
<tfooter>	It is used to group the footer content in a table.

### HTML Table Example

Let's see the example of HTML table tag. It output is shown above.

<table>

```

<tr><th>First_Name</th><th>Last_Name</th><th>Marks</th></tr>
<tr><td>Sonoo</td><td>Jaiswal</td><td>60</td></tr>
<tr><td>James</td><td>William</td><td>80</td></tr>
<tr><td>Swati</td><td>Sironi</td><td>82</td></tr>
<tr><td>Chetna</td><td>Singh</td><td>72</td></tr>
</table>

```

Output:

First_Name	Last_Name	Marks
Sonoo	Jaiswal	60
James	William	80
Swati	Sironi	82
Chetna	Singh	72

In the above html table, there are 5 rows and 3 columns =  $5 * 3 = 15$  values.

### HTML Table with Border

There are two ways to specify border for HTML tables.

1. By border attribute of table in HTML
2. By border property in CSS

#### 1) HTML Border attribute

You can use border attribute of table tag in HTML to specify border. But it is not recommended now.

```

<table border="1">
<tr><th>First_Name</th><th>Last_Name</th><th>Marks</th></tr>
<tr><td>Sonoo</td><td>Jaiswal</td><td>60</td></tr>
<tr><td>James</td><td>William</td><td>80</td></tr>
<tr><td>Swati</td><td>Sironi</td><td>82</td></tr>
<tr><td>Chetna</td><td>Singh</td><td>72</td></tr>
</table>

```

Output:

First_Name	Last_Name	Marks
Sonoo	Jaiswal	60
James	William	80

Swati	Sironi	82
Chetna	Singh	72

### Unit – III

**Electronic Payment** is a financial exchange that takes place online between buyers and sellers. The content of this exchange is usually some form of digital financial instrument (such as encrypted credit card numbers, **electronic** cheques or digital cash) that is backed by a bank or an intermediary, or by a legal tender. An **electronic payment** is any kind of non-cash **payment** that doesn't involve a paper check. **Methods of electronic payments** include credit cards, debit cards and the ACH (Automated Clearing House) network. ... On some Web sites, you can use an e-check instead of a credit card.

After selecting to **pay** by credit or debit card, the user passes on information such as name, credit or debit card details, and billing address, and then submits **payment**. ... The card-issuing bank approves/denies the transaction based on card status and whether the transaction is within the cardholder's credit limit or not. Helping You Navigate Credit Card **Processing**. so you can have great results. **Electronic Payment Systems (EPS)** is committed to making your credit card **payment processing** and merchant services painless, stress-free, and easy to understand.

The definition of an **electronic payment system** is a way of paying for a goods or services **electronically**, instead of using cash or a check, in person or by mail. An example of an **electronic payment system** is Pay Pal. An example of an **electronic payment system** is the use of a credit card.

An e-commerce **payment system** facilitates the acceptance of **electronic payment** for online transactions. Also known as a sample of **Electronic Data Interchange (EDI)**, e-commerce **payment systems** have become increasingly popular due to the widespread use of the internet-based shopping and banking. **E-payment system** is a way of making transactions or paying for goods and services through an **electronic** medium without the use of check or cash. ... The **electronic payment system** has grown increasingly over the last decades due to the widely spread of internet-based banking and shopping. **Electronic money (e-money)** is broadly defined as an **electronic** store of monetary value on a technical device that may be widely used for making payments to entities other than the **e-money** issuer. The device acts as a prepaid bearer instrument which does not necessarily involve bank **accounts** in transactions. **Online payment** refers to money that is exchanged



electronically. Typically, this involves use of computer networks, the **internet** and digital stored value **systems**. When you collect a **payment** over the **internet**, you are accepting an **online payment**.

By far, an online **payment** solution is the easiest and fastest method of **payment** to you and your customers. With **ePay**, it's easy for you to accept **payments** from your customers via credit card. You avoid a lot of programming work and the very high security requirements imposed by the PCI standard. Digital **cash** is a system of purchasing **cash** credits in relatively small amounts, storing the credits in your computer, and then spending them when making **electronic** purchases over the Internet. **Payment gateway** are server based transaction processing system which enclose business to authorize, process, and manage credit card transaction securely in a real time. It act as an intermediate between merchant shopping cart and all financial network involved with transaction.

#### **The areas of e-commerce are**

- 1) EDI
- 2) E-market
- 3) Internet commerce

#### **Unit - IV**

##### **E-Commerce Applications:**

##### **Online Banking:**

**Online banking or e-banking** is an electronic payment system that enables customers of a bank to conduct a range of financial transactions through the secured internet application. With the rapid digitalization and Internet becoming an essential part of human lives, E-Banking has increased tremendously not only among young Internet-savvy people but for other age groups also. The banks provide its customers, 24 hours online banking services - anytime, anywhere with its Internet Banking. All the banking transactions can be performed by the customer from the comfort of their home or office.

All banking internet service requires security of the highest nature and complete privacy protection. In India banks use 128-bit encryption SSL (Secure Sockets Layer) which is digitally certified by

VeriSign for Internet and e-commerce transactions. The large, established "brick and mortar" banks introduced Internet Banking services in the mid-1990s. E-banking makes use of electronic currency. Credit cards, Debit cards, Smart cards or stored-value cards, digital cash and digital checks are the different types of electronic currency

Banks provide the following services:

- ☐ Account balance
- ☐ Account related enquiries and status
- ☐ Transaction tracking and history
- ☐ Loan Instalments and funds flow details
- ☐ Statements
- ☐ Cheque status
- ☐ Demat Account Details displays the name, correspondence address, account numbers and bank account numbers associated with the account.
- ☐ Statement of Transaction lists the transactions for a period, with details of security and balances.
- ☐ Cheque book
- ☐ Stop payment
- ☐ Fixed and Recurring deposit opening / renewal
- ☐ Mobile / DTH recharge

IPO application

- ☐ Bill Payment

In addition e-banking may offer an array of international transactions such as:

- ☐ Foreign Currency Account Transfer
- ☐ Telegraphic Transfers
- ☐ International Cheques (Bank Drafts)
- ☐ Letter of Credit Application and Amendment
- ☐ Guarantee and Shipping Company Guarantee

### **Benefits of e-Banking:**

e-Banking provides flexibility in banking because the customer can visit bank anytime, anywhere. It saves time and money by dealing with the day-to-day banking business using e-Banking. Since all the banking operations are done online, it helps to reduce paper consumption. E-banking provides a number of benefits of for the banking institution as well as the consumers which can be discussed as

follows:

### **Benefits of e-banking for the Banks:**

- ☐ **Price:** In the long run a bank can save money by engaging lesser staff for managing branches. Also, it is cheaper to make transactions over the Internet.
- ☐ **Wider Customer Base:** The Internet allows banks to reach a whole new markets and wider customer base because there are no geographical boundaries with the Internet. The Internet also provides a level playing field to small banks.
- ☐ **Enhancing Efficiency of Operations:** The efficiency of banks improves by providing Internet access for their customers. The Internet provides the bank with an almost paper less system.
- ☐ **Customer Service and Satisfaction:** The banking on the Internet not only allows the customer to have a full range of services available to them but it also allows them some services not offered at any of the branches. With more better and faster options banks can provide better customer relations and satisfaction.
- ☐ **Building better corporate image:** Internet banking adds to the image and goodwill of the bank. A bank seems more state of the art to a customer if they offer Internet access. A person may not want to use Internet banking but having the service available gives a person the feeling that their bank is on the cutting image.

### **Benefits of e-banking for the Customers:**

- ☐ **Bill Pay:** Customers can make payments of bills online. The bank will withdraw the money from customer account and send the payee a paper check or an electronic payment
- ☐ **Buy and Sell Securities:** With the onset of dematerialization, many banks have started providing the online stock trading services to its customers. The customers can check stock market information, currency rates, etc and trade in them online through e-banking.
- ☐ **Check Balances:** The customers can check the balances in their accounts on the go.
- ☐ **Saves Time, Money and effort:** The customers save time, because they do not have to stand in a queue at their bank for availing services. Online banking is free of charge which saves their money.
- ☐ **24X7 access sitting at the comfort of work or home:** The customers can have access to banking all day long from anywhere.
- ☐ **Quick service:** The online service is instantaneous therefore the customers do not have to wait and there is no delay as in the physical set up.

### **Concerns with e-Banking:**

Although the trend of internet banking is on the rise but still there are some issues involved in e banking. They are:

□ **Customer Support:**

The banks need to create a whole new customer relations department to help e-banking customers. Banks have to ensure quick and timely assistance to the customers. Any problem can destroy the banks reputation quickly.

□ **Complying with the Laws:**

While Internet banking does not have national or state boundaries but there are different laws in different countries.

□ **Privacy and Security:**

The customer always has a doubt on the privacy, security and accuracy of their transactions.

**Online Insurance:**

E-insurance involves the advertisement, recommendation, negotiation, and purchase and claim settlement of insurance policies through the Internet. The consumers can search from different insurance companies for insurance product. They can evaluate the products from different companies to determine the one which best suits their needs. The terms of the insurance policy are then conveyed to the customer by the insurance company and the customer responds with details including a description of the entity being insured, the terms and the duration of the insurance policy. The buyer then pays the initial premium to the insurance company and the policy certificate is sent to the buyer. At present, most of these processes are not automated. Some e-insurance sites offer web-based policies but these are little more than passive catalogs of alternatives available to buyers. The conventional insurance differs from e insurance in the following ways:

1. In the conventional insurance industry, the insurance company informs its customers through advertisements. Advertisements are made either through passive channels like newspapers, magazines, billboards, radio and television, or through active channels like human insurance agents. E-insurance employs the Internet to reach customers through advertisements more effectively since the Internet integrates the traditional passive and active channels of advertisements into one. Advertisement banners, e-mail notifications and coupons are used to replace passive media, while software agents replace their active human counterparts. E-insurance accesses the buyer more effectively through multiple channels, which increases the market penetration of e-insurance over traditional insurance.

2. The buyer can pay the insurance through a credit card or authorize payment from a financial institution like a bank. Arrangements can also be made to ensure that later premiums are transferred automatically from the insurer's bank account without his intervention in the future.
3. Online claim settlement is difficult to automate completely because it requires coordination between parties outside the insurance company and physical examination of the evidence of damage.
4. Online policy purchase is faster, more users friendly and definitely more secure than the traditional processes. Therefore it is more attractive to the insurer. At the same time it incurs less cost and requires fewer resources than traditional insurance and is therefore more profitable for the insurance company.

### **Payment of Utility Bills**

The utility bills can be paid electronically by online, mobile and telephone banking. The customer can transfer money from their bank account to a creditor or vendor such as a public utility, department store or an individual to be credited against a specific account. These payments are typically executed electronically as a direct deposit through a national payment system, operated by the banks or in conjunction with the government. The banks also offer electronic payment system which enables the customer to schedule payments in advance to be made on a specified date (convenient for instalments such as mortgage and support payments), to save the information of the biller to be used in future and various options for searching the recent payment history.

### **Online Marketing**

Online marketing also known as web marketing, internet marketing or e-marketing, is referred to as the marketing of products or services over the Internet. It refers to a set of powerful internet tools and methodologies used for promoting products and services. It incorporates a wider range of marketing elements than traditional business marketing due to the extra channels and marketing mechanisms available on the Internet and connects organizations with qualified potential customers and takes business development to a much higher level than traditional marketing/advertising. Online marketing synergistically combines the Internet's creative and technical tools, including design, development, sales and advertising.

Online marketing has several advantages, including:

- ❑ **Reduced costs:** Large audiences can be catered with in a fraction of seconds as compared to traditional advertising budgets, allowing businesses to create good consumer ads.
- ❑ **Flexibility and convenience:** Consumers may research and purchase products and services at their leisure.
- ❑ **Multiple options:** Advertising tools include pay-per-click advertising, email marketing and local search integration (like Google Maps).
- ❑ **Demographic targeting:** Consumers can be demographically targeted much more effectively in an online rather than an offline process.

It includes Search Engine Marketing, Email marketing, Online Display Ads, Search Engine Optimisation, Social Media Marketing, Pay per Click, Internet Viral Marketing. Online marketing has overtaken traditional advertising in the recent years and continues to be a high-growth industry.

### **E-tailing:**

E-tailing is defined as the sale of goods and services through the Internet. Today is the Era of Globalization and the consumer is not confined to a particular place to access products. Electronic retailing or e-tailing include business-to-business and business-to-consumer sales. E-tailing revenue can come from the sale of products and services, through subscriptions to website content, or through advertising. E-tailing requires businesses to tailor traditional business models to the rapidly changing face of the Internet and its users.

Online retailing is classified into three main categories:

#### **1. Click:**

The businesses that operate only through the internet fall into this category. Prominent examples in this category include Amazon.com and e-Bay.

#### **Brick and Click:**

The businesses that use both the online as well as the offline channel fall into this category.

#### **3. Brick and Mortar:**

This is the conventional mode of retailing. The businesses that do not use the latest retailing channels and still rely upon the conventional mode belong to this category.

In India e-tailing comprises of online retail and marketplaces, has become the fastest-growing segment in the larger market growing at a CAGR of around 56% over 2009-2014. According to Google India the online shoppers in India are expected to cross 100 million by end of year 2016. By 2020, India is expected to generate \$100 billion online retail revenue. Books, apparel and accessories

and electronics are the largest selling products through e-Tailing, constituting around 80% of product distribution. The increasing use of smart phones, tablets and internet broadband, 3G and 4G has led to a strong consumer base. Popularity of cash-on-delivery (COD), increased women workforce, growing acceptability of online payments, favourable demographics, and lack of organized retail market are the other key factors driving the growth of online retail in India.

### **Factors behind the Growth of E-tailing in India**

#### **1. No rent or cost of purchasing land:**

E-retailers need not invest money in big showrooms at prime locations, and can simply operate through their websites or portals. This saves the costs of stores which are pretty high for physical store retailers.

#### **2. Increased communication with the client:**

E-Retailing enables personalized and customised interaction with customers.

#### **3. Universal reach:**

A supermarket has a limited geographical area of operation. It caters only to a limited number of customers to a particular area but a website can be accessed from any part of the globe.

#### **4. Convenience shopping:**

Online shopping saves time. Shopping in the comfort of home through the Internet is a huge attraction for customers. The online store is accessible 24×7,365 days and delivers the products at the required location which saves time and effort.

#### **5. Price and Selection:**

Online shoppers can easily compare the prices of the products they want to buy from different web sites, and then go for the purchase accordingly.

#### **6. Internet Boom:**

After 2005 there has been tremendous increase in the number of people using internet due to good broadband services in the country.

#### **7. Standard of living:**

The increased living standard of the people has made consumers inclined towards e-retail sites for shopping.

#### **8. Availability of wider range:**

Online market offers a much wider range of products than any retail shop. Consumers have wider options for the products online and even can buy those products that are not available at the retail shops.

### **9. Lifestyle:**

In today's busy lifestyle, lack of time for offline shopping and traffic congestion has led to the boom of online shopping market.

### **10. Competition in online market:**

Competition among the online shopping websites has attracted more customers to go for online shopping. Stiff competition has led to many offers and discounts by the websites. Hence products are available at very less prices to the consumers.

### **11. Promotional Tool:**

A website can be used as a medium to conduct promotional experiments, due to the wide reach of the internet, and the low cost. Therefore, it will be a great opportunity for Indian companies to promote their businesses.

### **12. Marketing Tool:**

A website is also an effective channel to communicate with customers. The internet provides a two-way communication channel. As a new communication channel, the internet can provide benefits to retailers, such as low costs, interactivity, personalization, and continuous communication. The growth of e-commerce is restricted due to the following reasons:

- ☐ Consumers can not touch and feel products.
- ☐ Delivery of Orders may take time.
- ☐ Shipping costs are often excessive
- ☐ After Sales service is poor
- ☐ Returns can be difficult and cumbersome

### **Advantages**

Online retailing has various advantages. These online retailers are redefining the retailing activities. Some of the advantages of online retailing are:

1. Lesser time and less space
2. Mass customization
3. Creation of supply chain
4. Identification of specific needs



5. Maintaining higher renewal rates
6. High usage rates
7. Low operating cost
8. Minimum postage expenses
9. Low paper transactions
10. Easier sales follow-up.

### **Risks Associated with E Tailing**

There are many risks of online business which may pose threat to the e tailing:

#### **1. Problems with the Payment System:**

People in India are not accustomed to the online shopping system. They are also dubious regarding the online payment system through the credit cards. Companies need to protect their system from hackers using technological and legal tools as customers often worry about theft of their personal information.

#### **2. Lack of Full Cost Disclosure:**

It is very easy to compare the basic price of an item online but the consumer may not be able to see the total cost as additional fees such as shipping costs are often not mentioned.

#### **3. Handling Returns:**

The customers must be able to return defective or unwanted product which he receives. At present, the online companies do exchange the goods but how long it will continue need to seen.

#### **4. Problems with Shipping Time:**

The customers using the online shopping should be assured that the products that they have ordered would reach them in due time.

#### **5. Offline Presence:**

The online retailers should have some offline presence as this gives customers the psychological comfort and trust.

#### **6. Language Problem:**

Most online retail shops use English as their mode of communication. English is not understood by the majority of the Indian population especially in remote and rural areas. To increase the customer base, content in the online retail shops should be provided in local language.

#### **7. Inferior product:**

Sometimes the description of the product might be different than the actual product. As a result the customer might end up with inferior quality product.

With the economic slowdown worldwide, many retailers who preferred having a presence are looking to go online at minimal costs since e-store is relatively small, convenient and has a low-cost start-up. The only costs involved in the e-tailing platform include the monthly hosting and ISP bills.

## **Online Services:**

### **6.1 Online Financial Services:**

There are many financial services which are transacted online. The common among them are:

- a. Internet Banking
- b. Bill Payment
- c. E-Brokerage
- d. E-insurance
- e. E-Delivery of Financial Services

### **6.2 Online Travel Industry:**

The travel industry is booming around the globe. In India e-commerce is being driven by the online travel industry and online travel bookings have increased substantially after the entry of low cost carriers. Currently, online travel industry is contributing 70% to the revenue generated by e-commerce in India. The companies that are involved in purchase of flights, booking hotel rooms, rental cars, and travel-related activity over the web comprise the online travel industry. There are well known travel websites such as **Expedia, Make my Trip, and Trip Advisor**.

The online travel industry is divided into three primary categories: suppliers, online travel agencies (OTA's) and aggregators. Suppliers include airlines, hotels, and rental car companies. They sell services directly to consumers via their own websites, but also widely utilize online Travel Agencies and aggregators to market their inventories. OTAs bridge the gap between suppliers and customers. They give suppliers quotes to consumers and fulfill online orders. Aggregators provide a platform where by the web users can compare the prices of OTAs and suppliers for specific travel queries. They route the users back to the organizations for purchases.

### **Drivers of the Online Travel Industry**

1. The global economic growth has led to the boom of travel industry. The increased income of the family has led to more and more spending on travel.
2. Perhaps the most noticeable trend driving the online travel industry is the shift from desktop computing to mobile phones and tablets. The general tilt in the population toward "mobile" usage is

having a marked impact on the online travel industry. The growing tendency for digital apps to foster consumer-to consumer transactions will also influence the online travel industry in the near future.

3. The long-term capacity trends in the airline industry will foster online travel opportunities in future. Expansion in the online industry has occurred in tandem with the falling cost per mile of air travel to consumers, as airlines have revamped their fleets with lighter, more fuel-efficient aircraft and focused on lowering fixed costs and increasing profitability.

4. Air travel is vital to the online travel industry, as healthy aviation traffic drives not only sales of flights, but hotel stays and rental car bookings as well.

### **Advantages of Online travel Industry**

The advantages of using an online travel services include:

- ☐ Reduced travel cost due to elimination of middleman
- ☐ Potential reductions in online marketing spend from travel agents.
- ☐ Ease of comparing accommodation costs and the services offered by individual travel providers.
- ☐ Impartial reviews on online travel sites may give new customers the confidence to book.
- ☐ Personalised and customised services available.
- ☐ Availability of substantial Discount
- ☐ The customer can make all travel plans on the Internet at their own convenience from anywhere.
- ☐ The customer can check the images and can compare the reviews of the hotels, airlines and places.
- ☐ Saves time and Money.

### **Disadvantages of Online Travel Agents**

However, there may be some disadvantages to using online travel services. These include:

- ☐ While booking Online travel there is nobody to give advice on the holiday like in a travel agents
- ☐ Possibility of Credit Card/ identity theft
- ☐ Site may not be secure enough

### **Online Career Services**

With the growth in technology, availability and access to internet online recruitment has become one of the important and primary sources of recruitment .The recruitment is done online by posting job advertisements and receiving applications online. Job Portals have displayed an exponential growth in the first decade of this millennium. It is an effective means of linking job seekers and employers. Internet is being increasingly used by both employees and employers to gather a more truthful picture about each other. Using World Wide Web over traditional sources like company brochures,

job fairs etc, provides benefits like greater quantity and variety of recruitment information at a much lower cost can be disseminated.

India's online recruitment industry took shape in 1997. The growth of the services sector, following the launch of economic reforms in 1991, resulted in the creation of additional jobs. In this background, internet proved to be an efficient medium that allowed employers and job seekers to connect. Prior to job portals, weekly government magazines such as Employment News and newspaper notifications were the primary means for employers and job seekers to interact. In India Naukri, JobsAhead (subsequently bought by Monster to start India operations), TimesJobs and Shine have been at the forefront in making online recruitment. The following are the major services available on the internet:

- a) Finding a Job
- b) Writing and Posting Resumes
- c) Career Planning
- d) NewsGroups

**Advantage:**

1. *Time Saving:* The process of filling a position through newspaper classifieds, headhunters or internal personnel departments is time-consuming against online process.
2. *Cost Effective:* Online applications reduce the cost of hiring considerably. Most online job classified sites are free or charge minimal fees for employers and job seekers.
3. *Ease of Storage:* Job applications received in digital format are easy to store, sort or screen for certain keywords thereby saving the time needed to scan the paper applications.
4. *Wide reach:* Web is a global data source. While newspapers and company announcements tend to reach only local markets, online applications are available to the whole world, including promising candidates who may be willing to relocate to for the right job.
5. *More Choices:* Job Openings are advertised on the Internet through job-posting sites, company websites, blogs and social networks, Thereby leading to wider opportunities for the job seekers.
6. *Convenience:* One of the major advantages of job hunting on the Internet is convenience. The candidate can access thousands of job openings from the comfort of the home. In case of a long-distance job search, employers can conduct video interviews, which is convenient for both employers and job seekers.

**Disadvantages:**

1. *Unqualified Applicants*: An employer may receive applications from unqualified applicants and applicants who are not seriously interested in the position. Screening thousands of applications can make selection process less efficient than traditional recruiting.
2. *Form Limitations*: A standardized online form makes sorting data and screening applications much simpler for employers, but it constrains job seekers and keeps them from uploading non traditional data associated with their resume. Particularly in creative industries, items like video and audio samples can be critical to considering an applicant. The limitations of file sizes and formats can keep an employer from receiving important application materials.
3. *Limited Job Pool*: Usually only about one-fourth of all job openings are advertised, online job-hunting is largely ineffective, compared to using professional networking as a job search tool for job seekers. Larger jobs can be tapped through professional network and interacting with colleagues and peers in the industry or career area.
4. *Security and Privacy Issues*: Online resumes may be misused for data mining and identity theft.

### **Online Auction**

An online auction is an auction done on the internet. There are four main types of auctions

#### **1. Ascending-bid auctions or English auctions:**

These auctions are carried out interactively in real time; with bidders present either physically or electronically. The seller gradually raises the price, bidders drop out until finally only one bidder remains, and that bidder wins the object at this final price. Oral auctions in which bidders shout out prices, or submit them electronically, are forms of ascending-bid auctions.

#### **2. Descending-bid auctions or Dutch auctions:**

This is also an interactive auction format, in which the seller gradually lowers the price from some high initial value until the first moment when some bidder accepts and pays the current price. These auctions are called Dutch auctions because flowers have long been sold in the Netherlands using this procedure.

#### **3. First-price sealed-bid auctions:**

In this kind of auction, bidders submit simultaneous “sealed bids” to the seller. The seller would then open them all together. The highest bidder wins the object and pays the value of her bid.

#### **4. Second-price sealed-bid auctions:**

Bidders submit simultaneous sealed bids to the sellers; the highest bidder wins the object and pays the value of the second-highest bid.

Online auctions have increased the quality and quantity of goods and services under the ambient of auction mechanisms. The auctions conduct has been expanded. The new and innovative products are being auctioned online. In the current web environment there are hundreds of websites dedicated to online auction practices.

### **Online Portal:**

**Portal** is a term, generally synonymous with *gateway*, for a World Wide Web site that is a starting site for users when they get connected to the Web or that users tend to visit as an anchor site.

A portal provides at least four essential services:

1. Search engine(s),
2. Email, 3. Links to other related sites, and
4. Personalized content.

It may also provide facilities such as chat, members list, free downloads and many more.

There are two types of Web portals Horizontal Web portals and Vertical Web portals.

### **Horizontal portals:**

They target the entire web community. These are general portals. These sites, often referred to as "mega portals", usually contain search engines and provide the ability for user to personalize the page by offering various channels (i.e. access to other information such as regional weather, stock quotes or news updates). Some major general portals include Yahoo, Excite, Netscape, Lycos, CNET, Microsoft Network, and America On line's AOL.com.

### **Vertical portals:**

These are niche portals. They offer information and services customized to niche audiences about a particular area of interest. Vertical industry portals, known as vortals, are sites that provide a gateway to information related to a particular industry, such as, insurance, automobiles, etc.

Other types of portals are:

### **Enterprise Portals:**

These are portals developed and maintained for use by members of the intranet or the enterprise network. The enterprise portals focus on providing employees with the information on a regularly updated manner along with document management system, availability of applications on demand, online training courses and web casts etc and communication in the form of emails, messaging, web meetings etc.

### **Knowledge Portals:**

Knowledge portals increase the effectiveness of knowledge workers by providing easy access to information that is necessary or helpful to them in one or more specific roles.

### **Corporate Portals:**

A corporate portal provides personalized access to an appropriate range of information about a particular company. Corporate portals aim at providing a virtual workplace for each individual using them - executives, employees, suppliers, customers, third-party service providers.

### **Market space Portals:**

They provide support to business-to-business and business-to-customer e-commerce, software support for e-commerce transactions and ability to find and access rich information about the products on sale also, ability to participate in discussion groups with other vendors and/or buyers

### **Online Learning:**

Online learning is the newest and most popular form of distance education today. Within the past decade it has had a major impact on education and the trend is only increasing. It is defined as use of Internet technologies to communicate and collaborate in teaching and learning. The online learning has the following advantages:

#### **1. Enhance student-to-student and faculty-to-student communication:**

Web-based education tools provide different ways to increase communication between students and faculty through discussion boards, chats, and emails.

#### **2. Enable student-centered teaching approaches:**

Every student has a unique learning style. Some students learn by viewing while others learn by doing. Web-based learning environment permit the instructor to present in many formats to accommodate different types of learning styles. For example instructor can put both lecture notes and slides online offering both visual and auditory learners benefit.

#### **3. Provides 24/7 accessibility to course materials:**

The online environment provide 24/7 access to the students. The students can have access to the learning material as per their own convenience.

#### **4. Provides just-in-time methods to assess and evaluate student:**

Online assessment tools provide instructors with many ways to build, distribute, and compile information quickly and easily.

#### **5. Saves Time, Money and Effort:**



Students can save the text and print as and when required. The direct result is a reduced institutional expense for both the cost and time associated with copying, collating, and distributing these materials. Instructors can also use E-mail to send messages directly to students or the Announcements feature to communicate with the entire class. Not only does this insure that students receive the materials, but it is also environmentally appealing, as it drastically reduces paper waste.

#### **6. Utilize time efficiently:**

Students benefit because they have immediate access to course materials at any location. They do not have to spend time walking across campus to the instructor's office or searching for a reading in the library.

#### **7. Students experience a sense of equality:**

Another benefit of using web-based communication tools is that it gives all students a reinforced sense of equality. Each individual has the same opportunity to "speak up" by posting messages without typical distractions such as seating arrangements, volume of student voices, and gender biases. Shy and anxious students feel more comfortable in expressing ideas and backing up facts when posting online instead of speaking in a lecture room.

#### **Disadvantages**

The most common disadvantages of online learning are outlined below:

1. Developing online courses requires a significant amount of time, money and resources. Institutions may be hesitant to implement an online program because it means that they will need to develop the online infrastructure, like course portals, and hire staff for issues related to tech support and student assistance.
2. Institutions may also need to invest in hiring or training faculty to teach in an online format. Many instructors are comfortable with teaching face-to-face, but need significant training in order to successfully teach an online course.
3. Instructor communication and availability is an important component of online learning. If the facilitator has difficulty being available to students or creating an environment in which participation is vital, this may impact the students' feeling of engagement.
4. Although instructors are available to students via e-mail, telephone, Web discussion boards and other online means, students may see the lack of face-to-face interaction and one-on-one instruction as a challenge.
5. A lack of communication or miscommunication between instructors and students may frustrate students who are struggling with course materials.



6. A lack of access to the appropriate technology may exclude some students, and technical problems may discourage others. This is particularly relevant to students in rural or lower socioeconomic areas.
7. Although online learning have grown in recent years, there are still many fields of study like healthcare and engineering, that require practical instruction cannot be successful in online environment.

### **Online Publishing:**

It refers to publishing the text and information online. The trend has been setting in for online publishing around the world. The readers are fast catching up with this trend. This is evident from the increased sale of devices like Kindle across the world.

### **Advantages of Online Publishing:**

1. Comfort: In this technological age the information available online is staggering, the person can access it anytime.
2. Cost: The costs of online desktop publishing are fairly low in consideration to those of print. The distribution itself is free. There are no printing costs, which are usually print publishers' biggest expense, nor the waste of large amounts of paper that go along with printing.
3. Most publications online right now are free to readers and are merely charging for ad space. Only few of them require subscriptions.
4. Editing is another plus involved in online publishing. Normally editing is done before the new issue goes online. In online publishing, there is no "final" product. Errors can be corrected in a matter of minutes (or seconds even).
5. The deadlines for online publication are merely self-imposed. For print, the editors have to take into consideration the printing and distribution time. Therefore, their deadlines are fairly rigid. However, for online publishing, deadlines are good to get the ball going, but the actual publishing can occur at any time without the dependence on the time-frame of another.
6. Since online desktop publishing is a fairly new field, there are no set standards for quality layout format leading to more room for experimentation.
7. In online publishing the audience is not dependent on distribution efforts and national boundaries. The online publication may be more widely available.

### **Disadvantages**

1. Readers are still not comfortable to sit down in front of a computer screen computer screen and read for great deal of time. Reading with a book in hand can never be replaced by e reading.
2. The costs that may come into play are those associated with online access. There are marketing costs of an online publication. Just because your publication is online, doesn't mean anyone out there knows where it is or is reading it.
3. It is very difficult to make any money from online publishing.
4. Audience is both a pro and a con for online publishing. Although publication may be more widely available, that doesn't mean that people are reading it. It's more difficult to determine the readership in online publications. Online, it's difficult to determine the quality and quantity of the audience.
5. It's still a volatile situation without any standards in online publishing. It is difficult to evaluate the readers like and dislikes. So while the content might be great, the layout could chase the readers away, and vice versa.
6. Submissions are another tough area to tackle in online publishing due to least Copyright laws.
7. Plagiarism is another threat for online publishing.

### **Online Entertainment:**

Online entertainment has become part of our everyday lives. The Internet has made a profound impact on entertainment, particularly since individuals have been able to gain access to it on the computers within their own homes, and more recently on portable devices such as mobile phones and iPods. It has provided entertainment creators with another avenue to explore, not just in still image and text but interactively with sound, motion and the live updating of content. The online entertainment industry enables the viewer to watch movies, videos, TV shows, listen to music, make videos, play video games, chat with friends in real time and many more. The following entertainment activities are done online

1. **Online Games:** There are a variety of games available on the internet to choose from and the majority of them are free to play. The different types of games include fighting, adventure, sport, and strategy.
2. **Live Chatting:** Chatting on the internet has been one of the popular activities to do. While chatting online the user need not worry about going over minutes like using a cell phone thereby saving money on phone bill.
3. **Memorabilia** – Pictures, life experiences, and recent activity can be labelled and posted online. This activity is one of the most popular things done online. For instance, Facebook.

**4. Music and Movies** – The entertainment of music and movies have always been important in people's lives. Music can now be bought online as well as previewing the music before buying. Movies can be rented or purchased at a low price online.

The internet has added great value for entertainment. Many activities are being done by people over the internet. However entertainment from the internet has captured the interest of almost everyone, regardless of age. In the future, the internet is going to continue to provide more forms of entertainment and open more opportunities to be amused.

### **Online Shopping:**

Online shopping is the process whereby consumers directly buy goods or services from a seller in real-time, without an intermediary service, over the Internet. It includes B2B and B2C transactions. The rapidly increasing popularity of online shopping is a truly global phenomenon. Online stores are usually available 24 hours a day, and many consumers have Internet access both at work and at home. In case consumer is not satisfied with the ordered product they can easily return an item for the correct one or for a refund. Online stores describe products for sale with text, photos, and videos, whereas in a physical retail store, the actual product and the manufacturer's packaging will be available for direct inspection. While shopping online it is easy for the consumer to find out deals for items or compare prices and services with different vendors. Search engines, online price comparison services and discovery shopping engines can be used to look up sellers of a particular product or service. Evolution and success of the online market place model with sites like Amazon, Jabong, Flipkart, Snapdeal etc., show the increasing trend of online shopping.

Online shoppers can make payment by following means

- ☐ Cash on delivery (C.O.D)
- ☐ Cheque
- ☐ Debit card
- ☐ Credit Card
- ☐ Gift cards
- ☐ Postal money order

In online shopping there is no scope of physical inspection of goods before purchase, consumers are at higher risk of fraud on the part of the merchant than in a physical store. Phishing is another danger, where consumers are fooled into thinking they are dealing with a reputable retailer, when they have actually been manipulated into feeding private information to a system operated by a malicious party. Privacy of personal information is a significant issue for some consumers. Different legal

jurisdictions have different laws concerning consumer privacy, and different levels of enforcement. Many websites keep track of consumers shopping habits in order to suggest items and other websites to view. An overview of few popular websites is as follows:

#### **Flipkart.com:**

It was started in 2007 by Sachin Bansal and Binny Bansal. The company is registered in Singapore and is having headquarters in Bangalore, Karnataka. Flipkart is also the biggest ecommerce brand of India. In 2014 they acquired competitor Myntra.com. The website covers all products from home appliances to gadgets, books to video CDs, and garments.

#### **Amazon:**

It is a US based ecommerce company and is world's no. 1 ecommerce site. Amazon holds the record of employing maximum number of employees in an Internet based business company. Amazon started its journey from books only but then soon it came up with movie DVDs, CDs, appliances etc. and now provides almost every product online.

#### **Jabong.com:**

It is an Indian ecommerce biggie which is really known for its good service and products. The website is selling apparel, footwear, fashion accessories, beauty products, fragrances, home accessories and other fashion and lifestyle products. The company is having headquarters in Gurgaon, NCR. Jabong was one of the most visited e-commerce sites during the Great Online Shopping Festival 2013.

#### **Snapdeal.com:**

It is an Indian ecommerce site which started in 2010 only but gained a good popularity among Indian buyers in a very less time. As compared to competitors, the growth rate of snapdeal is phenomenal. Snapdeal covers mainly most of the products like Flipkart and Amazon. This ecommerce site is widely used by people because of its best offers and good delivery service.

### **WEB BASED PROGRAMMING (313)**

#### **UNIT- 1**

##### **What is PHP?**

- ❑ PHP stands for **PHP: Hypertext Preprocessor**
- ❑ PHP is a server-side scripting language, like ASP
- ❑ PHP scripts are executed on the server
- ❑ PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid,

PostgreSQL, Generic ODBC, etc.)

- PHP is an open source software
- PHP is free to download and use

### What is a PHP File?

- PHP files can contain text, HTML tags and scripts
- PHP files are returned to the browser as plain HTML
- PHP files have a file extension of ".php", ".php3", or ".phtml"

### What is MySQL?

- MySQL is a database server
- MySQL is ideal for both small and large applications
- MySQL supports standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use

### PHP + MySQL

- PHP combined with MySQL are cross-platform (you can develop in Windows and serve on a Unix platform)

### Why PHP?

- PHP runs on different platforms (Windows, Linux, Unix, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP is FREE to download from the official PHP resource: [www.php.net](http://www.php.net)
- PHP is easy to learn and runs efficiently on the server side
- Your peers can help you through online

### Where to Start?

To get access to a web server with PHP support, you can:

- Install Apache (or IIS) on your own server, install PHP, and MySQL(For this module we install XAMMP which is a 3 in 1 software comprising of PHP, MySQL and Apache)
- Or find a web hosting plan with PHP and MySQL support

## What do you Need?

If your server supports PHP you don't need to do anything. Just create some .php files in your web directory, and the server will parse them for you. Because it is free, most web hosts offer PHP support.

However, if your server does not support PHP, you must install PHP.

Here is a link to a good tutorial from PHP.net on how to install

PHP5: <http://www.php.net/manual/en/install.php>

Download PHP

Download PHP for free here: <http://www.php.net/downloads.php>

Download MySQL Database

Download MySQL for free here: <http://www.mysql.com/downloads/index.html>

Download Apache Server

Download Apache for free here: <http://httpd.apache.org/download.cgi>

PHP code is executed on the server, and the plain HTML result is sent to the browser.

## Basic PHP Syntax

A PHP scripting block always starts with **<?php** and ends with **?>**. A PHP scripting block can be placed anywhere in the document.

On servers with shorthand support enabled you can start a scripting block with **<?** and end with **?>**.

For maximum compatibility, we recommend that you use the standard form (**<?php**) rather than the shorthand form.

```
<?php
```

```
?>
```

A PHP file normally contains HTML tags, just like an HTML file, and some PHP scripting code.

Below, we have an example of a simple PHP script which sends the text "Hello World" to the browser:

```
<html>

<body>

<?php
echo "Hello World";

?>
```

Each code line in PHP must end with a semicolon. The semicolon is a separator and is used to distinguish one set of instructions from another. There are two basic statements to output text with PHP: **echo** and **print**. In the example above we have used the echo statement to output the text "Hello World".

**Note:** The file must have a .php extension. If the file has a .html extension, the PHP code will not be executed.

### Comments in PHP

```
<html>

<body>

<?php

//This is a comment


/*
This is
a comment
block
```

In PHP, we use // to make a single-line comment or /\* and \*/ to make a large comment block.

A variable is used to store information.

## Variables in PHP

Variables are used for storing values, like text strings, numbers or arrays. When a variable is declared, it can be used over and over again in your script. All variables in PHP start with a \$ sign symbol.

The correct way of declaring a variable in PHP:

```
$var_name = value;
```

New PHP programmers often forget the \$ sign at the beginning of the variable. In that case it will not work.

Let's try creating a variable containing a string, and a variable containing a number:

```
<?php  
$txt="Hello World!";  
$x=16;
```

## PHP is a Loosely Typed Language

In PHP, a variable does not need to be declared before adding a value to it.

In the example above, you see that you do not have to tell PHP which data type the variable is. PHP automatically converts the variable to the correct data type, depending on its value.

In a strongly typed programming language, you have to declare (define) the type and name of the variable before using it.

In PHP, the variable is declared automatically when you use it.

## Naming Rules for Variables

- A variable name must start with a letter or an underscore "\_"
- A variable name can only contain alpha-numeric characters and underscores (a-z, A-Z, 0-9, and \_)
- A variable name should not contain spaces. If a variable name is more than one word, it should be separated with an underscore (\$my\_string), or with capitalization (\$myString)



- A string variable is used to store and manipulate text.

## String Variables in PHP

- String variables are used for values that contain characters.
- In this chapter we are going to look at the most common functions and operators used to manipulate strings in PHP.
- After we create a string we can manipulate it. A string can be used directly in a function or it can be stored in a variable.
- Below, the PHP script assigns the text "Hello World" to a string variable called \$txt:

```
<?php  
  
$txt="Hello World";  
echo $txt;  
  
?>
```

- The output of the code above will be:

```
Hello World
```

- Now, lets try to use some different functions and operators to manipulate the string.

## The Concatenation Operator

- There is only one string operator in PHP.
- The concatenation operator (.) is used to put two string values together.
- To concatenate two string variables together, use the concatenation operator:

```
<?php  
  
$txt1="Hello World!";  
  
$txt2="What a nice day!";  
echo $txt1 . " " . $txt2;  
  
?>
```

- The output of the code above will be:

```
Hello World! What a nice day!
```

- If we look at the code above you see that we used the concatenation operator two times. This is because we had to insert a third string (a space character), to separate the two strings.
- \_\_\_\_\_
- The strlen() function
- The strlen() function is used to return the length of a string.
- Let's find the length of a string:

```
<?php  
echo strlen("Hello world!");  
?>
```

The output of the code above will be:

```
12
```

- The length of a string is often used in loops or other functions, when it is important to know when the string ends. (i.e. in a loop, we would want to stop the loop after the last character in the string).

### The strpos() function

- The strpos() function is used to search for character within a string.
- If a match is found, this function will return the position of the first match. If no match is found, it will return FALSE.
- Let's see if we can find the string "world" in our string:

```
<?php  
echo strpos("Hello world!", "world");  
?>
```

- The output of the code above will be:

6

- The position of the string "world" in our string is position 6. The reason that it is 6 (and not 7), is that the first position in the string is 0, and not 1.

## PHP Operators

- This section lists the different operators used in PHP.

- **Arithmetic Operators**

Operator	Description	Example	Result
+	Addition	x= 2 x+ 2	4
-	Subtraction	x= 2 5-x	3
*	Multiplication	x= 4 x*5	20
/	Division	15/5 5/2	3 2.5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0

++	Increment	x= 5 x+ +	x=6
----	-----------	--------------------	-----

--	Decrement	x= 5 x--	x=4
----	-----------	----------------	-----

### Assignment Operators

Operator	Example	Is The Same As
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
.=	x.=y	x=x.y
%=	x%=y	x=x%y

### Comparison Operators

Operator	Description	Example
==	is equal to	5==8 returns false
!=	is not equal	5!=8 returns true
<>	is not equal	5<>8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true

>=	is greater than or equal to	5>=8 returns false
----	-----------------------------	--------------------

<=	is less than or equal to	5<=8 returns true
----	--------------------------	-------------------

## Logical Operators

Operator	Description	Example
&&	and	x= 6 y= 3 (x < 10 && y > 1) returns true
	or	x= 6 y= 3 (x==5    y==5) returns false
!	not	x= 6 y= 3 !(x==y) returns true

Conditional statements are used to perform different actions based on different conditions.

## Conditional Statements

Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.

In PHP we have the following conditional statements:

- ▢ **if statement** - use this statement to execute some code only if a specified condition is true
- ▢ **if...else statement** - use this statement to execute some code if a condition is true and another code if the condition is false
- ▢ **if...elseif ... else statement** - use this statement to select one of several blocks of code to

be executed

- switch statement - use this statement to select one of many blocks of code to be executed

## The if Statement

Use the if statement to execute some code only if a specified condition is true.

```
if (condition) code to be executed if condition is true;
```

### Syntax

```
<html>

<body>


<?php
$d=date("D");
if ($d=="Fri") echo "Have a nice weekend!";
?>
```

The following example will output "Have a nice weekend!" if the current day is Friday:

Notice that there is no ..else.. in this syntax. You tell the browser to execute some code **only if the specified condition is true**.

## The if...else Statement

Use the if... else statement to execute some code if a condition is true and another code if a condition is false.



```
if (condition)
```

```
    code to be executed if condition is true;
```

```
else
```

**Syntax**

## Example

```
<html>

<body>


<?php
$d=date("D");
if ($d=="Fri")
    echo "Have a nice weekend!";
else
```

The following example will output "Have a nice weekend!" if the current day is Friday, otherwise it will output "Have a nice day!":

If more than one line should be executed if a condition is true/false, the lines should be enclosed within curly braces:

```
<html>

<body>


<?php
$d=date("D");
if ($d=="Fri")
{
    echo "Hello!<br />";

    echo "Have a nice weekend!";
    echo "See you on Monday!";
```

## The if...elseif..... else Statement

Use the if...elseif ..else statement to select one of several blocks of code to be executed.

```
if (condition)

    code to be executed if condition is true;

elseif (condition)

    code to be executed if condition is true;

else
```

### Syntax

### Example

The following example will output "Have a nice weekend!" if the current day is Friday, and "Have a nice Sunday!" if the current day is Sunday. Otherwise it will output "Have a nice day!":

```
<html>

<body>

<?php

$d=date("D");

if ($d=="Fri")

    echo "Have a nice weekend!";

elseif ($d=="Sun")

    echo "Have a nice Sunday!";

else
```

Conditional statements are used to perform different actions based on different conditions.

## The PHP Switch Statement

Use the switch statement to select one of many blocks of code to be executed.

```
switch (n)
{
case label1:
    code to be executed if n=label1;
    break;
case label2:
    code to be executed if n=label2;
    break;
```

### Syntax

```
<html>
<body>
<?php
switch ($x)
{
case 1:
    echo "Number 1";
    break;
case 2:
    echo "Number 2";
```

This is how it works: First we have a single expression  $n$  (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use **break** to prevent the code from running into the next case automatically. The default statement is used if no match is found.

### Example

```
break;
default:

    echo "No number between 1 and 3";
}

?>
```

An array stores multiple values in one single variable.

### What is an Array?

A variable is a storage area holding a number or text. The problem is, a variable will hold only one value.

An array is a special variable, which can store multiple values in one single variable.

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
$cars1="Saab";

$cars2="Volvo";

$cars3="BMW".
```

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

The best solution here is to use an array!

An array can hold all your variable values under a single name. And you can access the values by referring to the array name.

Each element in the array has its own index so that it can be easily accessed.

In PHP, there are three kind of arrays:

- **Numeric array** - An array with a numeric index
- **Associative array** - An array where each ID key is associated with a value

## Associative Arrays

An associative array, each ID key is associated with a value. When storing data about specific named values, a numerical array is not always the best way to do it. With associative arrays we can use the values as keys and assign values to them.

### Example 1

In this example we use an array to assign ages to the different persons:

```
$ages = array("Peter"=>32, "Quagmire"=>30, "Joe"=>34);
```

### Example 2

```
$ages['Peter'] = "32";
```

```
$ages['Quagmire'] = "30";
```

```
$ages['Joe'] = "34";
```

This example is the same as example 1, but shows a different way of creating the array:

```
<?php
```

```
$ages['Peter'] = "32";
```

```
$ages['Quagmire'] = "30";
```

```
$ages['Joe'] = "34";
```

The ID keys can be used in a script:

```
Peter is 32 years old.
```

The code above will output:

## Multidimensional Arrays

In a multidimensional array, each element in the main array can also be an array. And each element in the sub-array can be an array, and so on.

### Example

In this example we create a multidimensional array, with automatically assigned ID keys:

```
$families = array
(
    "Griffin"=>array
    (
        "Peter",
        "Lois",
        "Megan"
    ),
    "Quagmire"=>array
    (
        "Glenn"
    ),
    "Brown"=>array
    (
        "Cleveland".
```

The array above would look like this if written to the output:

```
)  
[Quagmire] => Array  
(  
    [0] => Glenn  
)  
[Brown] => Array  
(  
    [0] => Cleveland  
    [1] => Loretta
```

## Example 2

Lets try displaying a single value from the array above:

```
echo "Is " . $families['Griffin'][2] .  
" a part of the Griffin family?";
```

```
Is Megan a part of the Griffin family?
```

The code above will output:

Loops execute a block of code a specified number of times, or while a specified condition is true.

## PHP Loops

Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal lines in a script we can use loops to perform a task like this.

In PHP, we have the following looping statements:

- **while** - loops through a block of code while a specified condition is true



- **do...while** - loops through a block of code once, and then repeats the loop as long as a specified condition is true
- **for** - loops through a block of code a specified number of times
- **foreach** - loops through a block of code for each element in an array

## The while Loop

The while loop executes a block of code while a condition is true.

```
while (condition)  
{  
    code to be executed;
```

### Syntax

### Example

The example below defines a loop that starts with i=1. The loop will continue to run as long as i is less than, or equal to 5. i will increase by 1 each time the loop runs:

```
<html>  
<body>  
<?php  
$i=1;  
while($i<=5)  
{  
    echo "The number is " . $i . "<br />";  
    $i++;
```

Output:

```
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
```

## The do...while Statement

The do...while statement will always execute the block of code once, it will then check the condition, and repeat the loop while the condition is true.

```
do
{
    code to be executed;
}
```

### Syntax

### Example

The example below defines a loop that starts with  $i=1$ . It will then increment  $i$  with 1, and write some output. Then the condition is checked, and the loop will continue to run as long as  $i$  is less than, or equal to 5:

```
<html>
```

```
<body>
```

```
<?php
```

```
$i=1;
```

```
do
```

```
}
```

```
while ($i<=5);
```

```
?>
```

Output:

```
The number is 2
```

```
The number is 3
```

```
The number is 4
```

```
The number is 5
```

```
The number is 6
```

The for loop and the foreach loop will be explained in the next chapter.

Loops execute a block of code a specified number of times, or while a specified condition is true.

### The for Loop

The for loop is used when you know in advance how many times the script should run.

```
for (init; condition; increment)
```

```
{
```

```
code to be executed;
```

## Syntax

### Parameters:

- *init*: Mostly used to set a counter (but can be any code to be executed once at the beginning of the loop)
- *condition*: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- *increment*: Mostly used to increment a counter (but can be any code to be executed at the end of the loop)

**Note:** Each of the parameters above can be empty, or have multiple expressions (separated by commas).

## Example

The example below defines a loop that starts with  $i=1$ . The loop will continue to run as long as  $i$  is less than, or equal to 5.  $i$  will increase by 1 each time the loop runs:

```
<html>
```

```
<body>
```

```
<?php
```

```
for ($i=1; $i<=5; $i++)
```

```
{
```

```
    echo "The number is " . $i . "<br />";
```

```
}
```

Output:

```
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
```

## The foreach Loop

The foreach loop is used to loop through arrays.

```
foreach ($array as $value)
{
    code to be executed;
```

### Syntax

For every loop iteration, the value of the current array element is assigned to \$value (and the array pointer is moved by one) - so on the next loop iteration, you'll be looking at the next array value.

### Example

The following example demonstrates a loop that will print the values of the given array:

```
<html>
<body>
<?php
$x=array("one","two","three");
foreach ($x as $value)
{
    echo $value . "<br />";
}
```

Output:

```
one  
two  
three
```

The real power of PHP comes from its functions. In PHP, there are more than 700 built-in functions.

## PHP Built-in Functions

### PHP Functions

In this chapter we will show you how to create your own functions.

To keep the browser from executing a script when the page loads, you can put your script into a function.

A function will be executed by a call to the function. You may call a function from anywhere within a page.

### Create a PHP Function

A function will be executed by a call to the function.

```
function functionName()  
{  
    code to be executed;
```

### Syntax

#### PHP function guidelines:

- Give the function a name that reflects what the function does
- The function name can start with a letter or underscore (not a number)

### Example

A simple function that writes my name when it is called:

```
<html>

<body>

<?php

function writeName()

{

echo "Kai Jim Refsnes";

}

echo "My name is ";
```

Output:

```
My name is Kai Jim Refsnes
```

## PHP Functions - Adding parameters

To add more functionality to a function, we can add parameters. A parameter is just like a variable.

Parameters are specified after the function name, inside the parentheses.

### Example 1

The following example will write different first names, but equal last name:

```
<html>

<body>
```

```
function writeName($fname)
{
    echo $fname . " Refsnes.<br />";
}

echo "My name is ";
writeName("Kai Jim");
echo "My sister's name is ";
writeName("Hege");

echo "My brother's name is ";
writeName("Stale");
```

My name is Kai Jim Refsnes.

My sister's name is Hege Refsnes.

My brother's name is Stale Refsnes.

Output:

## Example 2

The following function has two parameters:

```
<html>
<body>
<?php
function writeName($fname,$punctuation)
{
    echo $fname . " Refsnes" . $punctuation . "<br />";
```



```
echo "My sister's name is ";  
writeName("Hege","!");  
  
echo "My brother's name is ";  
writeName("Ståle","?");  
  
?>
```

Output:

```
My name is Kai Jim Refsnes.  
  
My sister's name is Hege Refsnes!  
My brother's name is Ståle Refsnes?
```

## PHP Functions - Return values

To let a function return a value, use the return statement.

```
<html>  
  
<body>  
  
<?php  
  
function add($x,$y)  
{  
  
$total=$x+$y;  
return $total;
```

### Example

```
</html>
```

Output:

1 + 16 = 17

## PHP Form Handling

The most important thing to notice when dealing with HTML forms and PHP is that any form element in an HTML page will **automatically** be available to your PHP scripts.

### Example

The example below contains an HTML form with two input fields and a submit button:

When a user fills out the form above and click on the submit button, the form data is sent to a PHP file, called "welcome.php":

"welcome.php" looks like this:

```
<html>
```

```
<body>
```

```
Welcome<?php echo $_POST["fname"]; ?>!  
You  
are<?php echo $_POST["age"]; ?>years old.
```

```
</body>
```

Welcome John!

You are 28 years old.

Output could be something like this:

### Form Validation

User input should be validated on the browser whenever possible (by client scripts). Browser validation is faster and reduces the server load.

You should consider server validation if the user input will be inserted into a database. A good way to validate a form on the server is to post the form to itself, instead of jumping to a different

page. The user will then get the error messages on the same page as the form. This makes it easier to discover the error.

---

The built-in `$_GET` function is used to collect values in a form with `method="get"`.

### The `$_GET` Function

The built-in `$_GET` function is used to collect values from a form sent with `method="get"`.

Information sent from a form with the GET method is visible to everyone (it will be displayed in the browser's address bar) and has limits on the amount of information to send (max. 100 characters).

```
<form action="welcome.php" method="get">
Name:<input type="text" name="fname"/> Age:
<input type="text" name="age" />

<input type="submit" />
```

### Example

When the user clicks the "Submit" button, the URL sent to the server could look something like this:

```
http://www.msu.ac.zw/welcome.php?fname=Peter&age=37
```

```
Welcome<?php echo $_GET["fname"];?>.<br/> You
are<?php echo $_GET["age"];?> years old!
```

The "welcome.php" file can now use the `$_GET` function to collect form data (the names of the form fields will automatically be the keys in the `$_GET` array):

### When to use `method="get"`?

When using `method="get"` in HTML forms, all variable names and values are displayed in the URL.

Note: This method should not be used when sending passwords or other sensitive information! However, because the variables are displayed in the URL, it is possible to bookmark the page.

This can be useful in some cases.

**Note:** The get method is not suitable for large variable values; the value cannot exceed 100 characters. The built-in \$\_POST function is used to collect values in a form with method="post".

## The \$\_POST Function

The built-in \$\_POST function is used to collect values from a form sent with method="post". Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send.

**Note:** However, there is an 8 Mb max size for the POST method, by default (can be changed by setting the post\_max\_size in the php.ini file).

```
<form action="welcome.php" method="post">
Name: <input type="text" name="fname" /> Age:
<input type="text" name="age" />

<input type="submit" />
```

## Example

```
http://www.msu.ac.zw/welcome.php
```

When the user clicks the "Submit" button, the URL will look like this:

The "welcome.php" file can now use the \$\_POST function to collect form data (the names of the form fields will automatically be the keys in the \$\_POST array):

```
Welcome<?php echo $_POST["fname"];?>!  
<br/> You  
are<?php echo $_POST["age"];?> years old.
```

## When to use method="post"?

Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send.

However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

## The PHP \$\_REQUEST Function

The PHP built-in \$\_REQUEST function contains the contents of both \$\_GET, \$\_POST, and \$\_COOKIE.

The \$\_REQUEST function can be used to collect form data sent with both the GET and POST methods.

```
Welcome <?php echo $_REQUEST["fname"]; ?>!<br /> You  
are <?php echo $_REQUEST["age"]; ?> years old.
```

## Example

### Server Side Includes (SSI)

You can insert the content of one PHP file into another PHP file before the server executes it, with the include () or require () function.

The two functions are identical in every way, except how they handle errors:

- include() generates a warning, but the script will continue execution
- require() generates a fatal error, and the script will stop

These two functions are used to create functions, headers, footers, or elements that will be reused on multiple pages.

Server side includes saves a lot of work. This means that you can create a standard header, footer, or menu file for all your webpages. When the header needs to be updated, you can only update the include file, or when you add a new page to your site, you can simply change the menu file (instead of updating the links on all your web pages).

## PHP include() Function

The include() function takes all the content in a specified file and includes it in the current file. If an error occurs, the include() function generates a warning, but the script will continue execution.

### Example 1

Assume that you have a standard header file, called "header.php". To include the header file in a page, use the include() function:

```
<html>

<body>

<?php include("header.php"); ?>

<h1>Welcome to my home page!</h1>

<p>Some text.</p>

</body>
```

### Example 2

Assume we have a standard menu file, called "menu.php", that should be used on all pages:

```
<a href="/default.php">Home</a>

<a href="/tutorials.php">Tutorials</a>

<a href="/references.php">References</a>

<a href="/examples.php">Examples</a>

<a href="/about.php">About Us</a>
```

All pages in the Web site should include this menu file. Here is how it can be done: If you look at

```
<html>

<body>

<div class="leftmenu">

<a href="/default.php">Home</a>

<a href="/tutorials.php">Tutorials</a>

<a href="/references.php">References</a>

<a href="/examples.php">Examples</a>

<a href="/about.php">About Us</a>

<a href="/contact.php">Contact Us</a>
```

the source code of the page above (in a browser), it will look like this:

### PHP require() Function

The require() function is identical to include(), except that it handles errors differently. If an error occurs, the include() function generates a warning, but the script will continue execution. The require() generates a fatal error, and the script will stop.

```
<html>

<body>

<?php
include("wrongFile.php");
echo "Hello World!";
```

### Error Example include() Function

**Warning:** include(wrongFile.php) [function.include]: failed  
to open stream:

No such file or directory in C:\home\website\test.php on line 5

**Warning:** include() [function.include]:  
Failed opening 'wrongFile.php' for inclusion  
(include\_path='.;C:\php5\pear')

</html>

Error message:

Notice that the echo statement is executed! This is because a Warning does not stop the script execution.

### Error Example require() Function

Now, let's run the same example with the require() function.

```
<html>

<body>

<?php
require("wrongFile.php");
echo "Hello World!";

?>
```

Error message:



**Warning:** require(wrongFile.php) [function.require]:

failed to open stream:

No such file or directory in C:\home\website\test.php on line 5

**Fatal error:** require() [function.require]:

Failed opening required 'wrongFile.php'

(include\_path='C:\php5\pear')

The echo statement is not executed, because the script execution stopped after the fatal error.

It is recommended to use the require() function instead of include(), because scripts should not continue after an error.

**The fopen() function is used to open files in PHP.**

## Opening a File

The fopen() function is used to open files in PHP.

The first parameter of this function contains the name of the file to be opened and the second parameter specifies in which mode the file should be opened:

```
<html>

<body>

<?php

$file=fopen("welcome.txt","r");

?>
```

The file may be opened in one of the following modes:

Modes	Description
R	Read only. Starts at the beginning of the file
r+	Read/Write. Starts at the beginning of the file
W	Write only. Opens and clears the contents of file; or creates a new file if it doesn't exist
w+	Read/Write. Opens and clears the contents of file; or creates a new file if it doesn't exist
A	Append. Opens and writes to the end of the file or creates a new file if it doesn't exist
a+	Read/Append. Preserves file content by writing to the end of the file
X	Write only. Creates a new file. Returns FALSE and an error if file already exists
x+	Read/Write. Creates a new file. Returns FALSE and an error if file already exists

**Note:** If the fopen() function is unable to open the specified file, it returns 0 (false).

### Example

The following example generates a message if the fopen() function is unable to open the specified file:

```
<html>

<body>

<?php

$file=fopen("welcome.txt","r") or exit("Unable to open file!");

?>
```

## Closing a File

```
<?php  
  
$file = fopen("test.txt","r");  
  
//somecodetobeexecuted  
  
fclose($file);
```

The fclose() function is used to close an open file:

## Check End-of-file

The feof() function checks if the "end-of-file" (EOF) has been reached. The feof() function is useful for looping through data of unknown length. **Note:** You cannot read from files opened in w, a, and x mode!

```
if (feof($file)) echo "End of file";
```

## Reading a File Line by Line

The fgets() function is used to read a single line from a file.

**Note:** After a call to this function the file pointer has moved to the next line.

## Example

The example below reads a file line by line, until the end of file is reached:

```
<?php  
  
$file = fopen("welcome.txt", "r") or exit("Unable to open file!");
```

```
//Output a line of the file until the end is reached
while(!feof($file))

{
    echo fgets($file). "<br />";
}

fclose($file);
```

### Reading a File Character by Character

The `fgetc()` function is used to read a single character from a file.

**Note:** After a call to this function the file pointer moves to the next character.

### Example

```
<?php

$file=fopen("welcome.txt","r") or exit("Unable to open file!");
while (!feof($file))

{
    echofgetc($file);
}


```

The example below reads a file character by character, until the end of file is reached:

```
<html>

<body>

<div class="leftmenu">

<?php include("menu.php"); ?>

</div>

<h1>Welcome to my home page.</h1>

<p>Some text.</p>

</body>
```

## Create an Upload-File Form

To allow users to upload files from a form can be very useful.

```
<html>
```

Look at the following HTML form for uploading files:

```
<form action="upload_file.php" method="post"
enctype="multipart/form-data">

<label for="file">Filename:</label>

<input type="file" name="file" id="file" />

<br />

<input type="submit" name="submit" value="Submit" />
```

Notice the following about the HTML form above:

- The `enctype` attribute of the `<form>` tag specifies which content-type to use when submitting the form. "multipart/form-data" is used when a form requires binary data, like the contents of a file, to be uploaded
- The `type="file"` attribute of the `<input>` tag specifies that the input should be processed as a file. For example, when viewed in a browser, there will be a browse-button next to the input field

Note: Allowing users to upload files is a big security risk. Only permit trusted users to perform file uploads.

## Create The Upload Script

```
<?php
if ($_FILES["file"]["error"] > 0)
{
    echo "Error: " . $_FILES["file"]["error"] . "<br />";
}
else
{
    echo "Upload: " . $_FILES["file"]["name"] . "<br />";
}
```

The "upload\_file.php" file contains the code for uploading a file:

By using the global PHP \$\_FILES array you can upload files from a client computer to the remote server.

The first parameter is the form's input name and the second index can be either "name", "type", "size", "tmp\_name" or "error". Like this:

- \$\_FILES["file"]["name"] - the name of the uploaded file
- \$\_FILES["file"]["type"] - the type of the uploaded file
- \$\_FILES["file"]["size"] - the size in bytes of the uploaded file
- \$\_FILES["file"]["tmp\_name"] - the name of the temporary copy of the file stored on the server
- \$\_FILES["file"]["error"] - the error code resulting from the file upload

This is a very simple way of uploading files. For security reasons, you should add restrictions on what the user is allowed to upload.

```
<?php
if ((($_FILES["file"]["type"] == "image/gif")
    || ($_FILES["file"]["type"] == "image/jpeg")
    || ($_FILES["file"]["type"] == "image/pjpeg"))
    && ($_FILES["file"]["size"] < 20000))
{
    if ($_FILES["file"]["error"] > 0)
    {
        echo "Error: " . $_FILES["file"]["error"] . "<br />";
    }
    else
    {
```

### Restrictions on Upload

In this script we add some restrictions to the file upload. The user may only upload .gif or .jpeg files and the file size must be under 20 kb:

```
}
```

**Note:** For IE to recognize jpg files the type must be pjpeg, for FireFox it must be jpeg.

### Saving the Uploaded File

The examples above create a temporary copy of the uploaded files in the PHP temp folder on the server.

The temporary copied files disappears when the script ends. To store the uploaded file we need

```
<?php
if ((($_FILES["file"]["type"] == "image/gif")
    || ($_FILES["file"]["type"] == "image/jpeg")
    || ($_FILES["file"]["type"] == "image/pjpeg"))
    && ($_FILES["file"]["size"] < 20000))
{
    if ($_FILES["file"]["error"] > 0)
    {
        echo "Return Code: " . $_FILES["file"]["error"] . "<br />";
    }
    else
    {
        echo "Upload: " . $_FILES["file"]["name"] . "<br />";
        echo "Type: " . $_FILES["file"]["type"] . "<br />";
        echo "Size: " . ($_FILES["file"]["size"] / 1024) . " Kb<br />";
        echo "Temp file: " . $_FILES["file"]["tmp_name"] . "<br />";
    }
}
```

to copy it to a different location:

```
    }
    else
    {
        echo "Invalid file":
    }
}
```

The script above checks if the file already exists, if it does not, it copies the file to the specified folder.



Note: This example saves the file to a new folder called "upload" A cookie is often used to identify a user.

---

## What is a Cookie?

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

### How to Create a Cookie?

The `setcookie()` function is used to set a cookie.

**Note:** The `setcookie()` function must appear BEFORE the `<html>` tag.

```
setcookie(name, value, expire, path, domain);
```

### Syntax

#### Example 1

In the example below, we will create a cookie named "user" and assign the value "Alex Porter" to it. We also specify that the cookie should expire after one hour:

```
<?php  
setcookie("user", "Alex Porter", time()+3600);
```

```
<html>
```

Note: The value of the cookie is automatically URLencoded when sending the cookie, and automatically decoded when received (to prevent URLencoding, use `setrawcookie()` instead).

## Example 2

You can also set the expiration time of the cookie in another way. It may be easier than using seconds.

```
<?php  
  
$expire=time()+60*60*24*30;  
setcookie("user", "Alex Porter", $expire);  
  
?>
```

In the example above the expiration time is set to a month (*60sec \* 60min \* 24hours \* 30 days*).

## How to Retrieve a Cookie Value?

The PHP `$_COOKIE` variable is used to retrieve a cookie value.

```
<?php  
  
// Print a cookie  
  
echo $_COOKIE["user"];  
  
// A way to view all cookies
```

In the example below, we retrieve the value of the cookie named "user" and display it on a page:

```
?>
```

In the following example we use the `isset()` function to find out if a cookie has been set:

```
<html>  
  
<body>  
  
<?php  
  
if (isset($_COOKIE["user"]))  
  
    echo "Welcome " . $_COOKIE["user"] . " !<br />";  
else  
  
    echo "Welcome guest!<br />";
```

## How to Delete a Cookie?

When deleting a cookie you should assure that the expiration date is in the past. Delete example:

```
<?php  
  
//set the expiration date to one hour ago  
setcookie("user", "", time()-3600);  
  
?>
```

## What if a Browser Does NOT Support Cookies?

If your application deals with browsers that do not support cookies, you will have to use other methods to pass information from one page to another in your application. One method is to pass the data through forms (forms and user input are described earlier in this tutorial).

The form below passes the user input to "welcome.php" when the user clicks on the "Submit" button:

```
<html>  
  
<body>  
  
<form action="welcome.php" method="post">  
Name: <input type="text" name="name" /> Age:  
<input type="text" name="age" />  
  
<input type="submit" />  
  
</form>
```

Retrieve the values in the "welcome.php" file like this:

```
<html>  
  
<body>  
  
Welcome <?php echo $_POST["name"]; ?>.<br /> You  
are<?php echo $_POST["age"]; ?>years old.  
  
</body>
```

A PHP session variable is used to store information about, or change settings for a user session. Session variables hold information about one single user, and are available to all pages in one application.

## PHP Session Variables

When you are working with an application, you open it, do some changes and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are and what you do because the HTTP address doesn't maintain state.

A PHP session solves this problem by allowing you to store user information on the server for later use (i.e. username, shopping items, etc). However, session information is temporary and will be deleted after the user has left the website. If you need a permanent storage you may want to store the data in a database.

Sessions work by creating a unique id (UID) for each visitor and store variables based on this UID. The UID is either stored in a cookie or is propagated in the URL.

## Starting a PHP Session

Before you can store user information in your PHP session, you must first start up the session.

**Note:** The `session_start()` function must appear BEFORE the `<html>` tag:

```
<?php session_start(); ?>

<html>

<body>

    ...
```

The code above will register the user's session with the server, allow you to start saving user information, and assign a UID for that user's session.

## Storing a Session Variable

The correct way to store and retrieve session variables is to use the PHP `$_SESSION` variable:

```
<?php
session_start();

// store session data

$_SESSION['views']=1;
```

```
<body>

<?php

//retrieve session data

echo "Pageviews=". $_SESSION['views'];

?>
```

```
Pageviews=1
```

Output:

In the example below, we create a simple page-views counter. The `isset()` function checks if the "views" variable has already been set. If "views" has been set, we can increment our counter. If "views" doesn't exist, we create a "views" variable, and set it to 1:

```
<?php
session_start();

if(isset($_SESSION['views']))

$_SESSION['views']=$_SESSION['views']+1; else

$_SESSION['views']=1;
```

## Destroying a Session

If you wish to delete some session data, you can use the `unset()` or the `session_destroy()` function. The `unset()` function is used to free the specified session variable:

```
unset($_SESSION['views']);

?>
```

You can also completely destroy the session by calling the `session_destroy()` function:

**Note:** session\_destroy() will reset your session and you will lose all your stored session data.

## The PHP mail() Function

The PHP mail() function is used to send emails from inside a script.

### Syntax

```
mail(to,subject,message,headers,parameters)
```

Parameter	Description
to	Required. Specifies the receiver / receivers of the email
subject	Required. Specifies the subject of the email. Note: This parameter cannot contain any newline characters
message	Required. Defines the message to be sent. Each line should be separated with a LF (\n). Lines should not exceed 70 characters
headers	Optional. Specifies additional headers, like From, Cc, and Bcc. The additional headers should be separated with a CRLF (\r\n)
parameters	Optional. Specifies an additional parameter to the sendmail program

### PHP Simple E-Mail

The simplest way to send an email with PHP is to send a text email.

In the example below we first declare the variables (\$to, \$subject, \$message, \$from, \$headers), then we use the variables in the mail() function to send an e-mail:

```
$to = "someone@example.com";  
  
$subject = "Test mail";  
  
$message = "Hello! This is a simple email message."  
  
$from = "someoneelse@example.com";  
  
$headers = "From: $from";
```

## PHP Mail Form

```
<html>

<body>

<?php
if (isset($_REQUEST['email']))
//if "email" is filled out, send email
{
//send email

$email = $_REQUEST['email'] ;
$subject = $_REQUEST['subject'] ;
$message = $_REQUEST['message'] ;
mail("someone@example.com", "Subject: $subject",
$message, "From: $email" );
echo "Thank you for using our mail form";
}
else
```

With PHP, you can create a feedback-form on your website. The example below sends a text message to a specified e-mail address:

```
</body>
```

This is how the example above works:

- First, check if the email input field is filled out
- If it is not set (like when the page is first visited); output the HTML form
- If it is set (after the form is filled out); send the email from the form

- When submit is pressed after the form is filled out, the page reloads, sees that the email input is set, and sends the email

**Note:** This is the simplest way to send e-mail, but it is not secure. In the next chapter of this tutorial you can read more about vulnerabilities in e-mail scripts, and how to validate user input to make it more secure.

There is a weakness in the PHP e-mail script in the previous chapter.

## PHP E-mail Injections

First, look at the PHP code from the previous chapter:

```
<html>

<body>


<?php
if (isset($_REQUEST['email']))
//if "email" is filled out, send email
{
    //send email

    $email = $_REQUEST['email'] ;

    $subject = $_REQUEST['subject'] ;

    $message = $_REQUEST['message'] ;
```



```
Email: <input name='email' type='text' /><br />
Subject: <input name='subject' type='text' /><br />
Message:<br />

<textarea name='message' rows='15' cols='40'>

</textarea><br />

<input type='submit' />

</form>";

}
```

```
someone@example.com%0ACc:person2@example.com
```

```
%0ABcc:person3@example.com,person3@example.com,
anotherperson4@example.com,person5@example.com
```

The problem with the code above is that unauthorized users can insert data into the mail headers via the input form. What happens if the user adds the following text to the email input field in the form?

The mail() function puts the text above into the mail headers as usual, and now the header has an extra Cc:, Bcc:, and To: field. When the user clicks the submit button, the e-mail will be sent to all of the addresses above!

## PHP Stopping E-mail Injections

The best way to stop e-mail injections is to validate the input.

The code below is the same as in the previous chapter, but now we have added an input validator that checks the email field in the form:

```
<html>
```

```
<body>
```

```
<?php
```

```
function spamcheck($field)
```

```
    $field=filter_var($field, FILTER_SANITIZE_EMAIL);
```

```
    //filter_var() validates the e-mail
```

```
    //address using FILTER_VALIDATE_EMAIL
```

```
    if(filter_var($field, FILTER_VALIDATE_EMAIL))
```

```
    {
```

```
        return TRUE;
```

```
    }
```

```
else
```

```
{
```

```
    return FALSE;
```

```
}
```

```
}
```

```
if (isset($_REQUEST['email']))
```

```
    { //if "email" is filled out, proceed
```

```
    //check if the email address is invalid
```

```
    $mailcheck = spamcheck($_REQUEST['email']); if
```

```
    ($mailcheck==FALSE)
```

```
    {
```

```
        echo "Invalid input";
```

```

    }
else
    { //send email

    $email = $_REQUEST['email'] ;

    $subject = $_REQUEST['subject'] ;

    $message      =      $_REQUEST['message']      ;
    mail("someone@example.com", "Subject: $subject",
    $message, "From: $email" );

    echo "Thank you for using our mail form";

    }
}

else

    { //if "email" is not filled out, display the form

    echo "<form method='post' action='mailform.php'> Email: <input name='email' type='text' /><br />
    Subject: <input name='subject' type='text' /><br /> Message:<br />

    <textarea name='message' rows='15' cols='40'>

    </textarea><br />

    <input type='submit' />

    </form>";

    }

?>

```

In the code above we use PHP filters to validate input:

- The FILTER\_SANITIZE\_EMAIL filter removes all illegal e-mail characters from a string

- The FILTER\_VALIDATE\_EMAIL filter validates value as an e-mail address
- The default error handling in PHP is very simple. An error message with filename, line number and a message describing the error is sent to the browser.

## PHP Error Handling

When creating scripts and web applications, error handling is an important part. If your code lacks error checking code, your program may look very unprofessional and you may be open to security risks.

We will show different error handling methods:

- Simple "die()" statements
- Custom errors and error triggers
- Error reporting

**Basic Error Handling:** Using the die() function

The first example shows a simple script that opens a text file:

```
<?php
$file=fopen("welcome.txt","r");
```

If the file does not exist you might get an error like this:

**Warning:** fopen(welcome.txt) [function.fopen]: failed to open stream:

No such file or directory in C:\webfolder\test.php on line 2

To avoid that the user gets an error message like the one above, we test if the file exist before we try to access it:

```
<?php
if(!file_exists("welcome.txt"))
{
    die("File not found");
}
else
```

Now if the file does not exist you get an error like this:

```
File not found
```

The code above is more efficient than the earlier code, because it uses a simple error handling mechanism to stop the script after the error.

However, simply stopping the script is not always the right way to go. Let's take a look at alternative PHP functions for handling errors.

### Creating a Custom Error Handler

Creating a custom error handler is quite simple. We simply create a special function that can be called when an error occurs in PHP.

This function must be able to handle a minimum of two parameters (error level and error message) but can accept up to five parameters (optionally: file, line-number, and the error context):

```
error_function(error_level,error_message,
error_file,error_line,error_context)
```

Syntax

Parameter	Description
-----------	-------------

error_level	Required. Specifies the error report level for the user-defined error. Must be a value number. See table below for possible error report levels
error_message	Required. Specifies the error message for the user-defined error
error_file	Optional. Specifies the filename in which the error occurred
error_line	Optional. Specifies the line number in which the error occurred
error_context	Optional. Specifies an array containing every variable, and their values, in use when the error occurred

### Error Report levels

These error report levels are the different types of error the user-defined error handler can be used for:

Value	Constant	Description
2	E_WARNING	Non-fatal run-time errors. Execution of the script is not halted
8	E_NOTICE	Run-time notices. The script found something that might be an error, but could also happen when running a script normally
256	E_USER_ERROR	Fatal user-generated error. This is like an E_ERROR set by the programmer using the PHP function trigger_error()
512	E_USER_WARNING	Non-fatal user-generated warning. This is like an E_WARNING set by the programmer using the PHP function trigger_error()
1024	E_USER_NOTICE	User-generated notice. This is like an E_NOTICE set by the programmer using the PHP function trigger_error()
4096	E_RECOVERABLE_ERROR	Catchable fatal error. This is like an E_ERROR but can be caught by a user defined handle (see also set_error_handler())
8191	E_ALL	All errors and warnings, except level E_STRICT (E_STRICT will be part of E_ALL as of PHP 6.0)

Now let's create a function to handle errors:

```
function customError($errno, $errstr)
{
    echo "<b>Error:</b> [$errno] $errstr<br />"; echo
    die();
}
```

The code above is a simple error handling function. When it is triggered, it gets the error level and an error message. It then outputs the error level and message and terminates the script.

### **Set Error Handler**

The default error handler for PHP is the built in error handler. We are going to make the function above the default error handler for the duration of the script. It is possible to change the error handler to apply for only some errors, that way the script can handle different errors in different ways. However, in this example we are going to use our custom error handler for all errors:

```
set_error_handler("customError");
```

Since we want our custom function to handle all errors, the `set_error_handler()` only needed one parameter, a second parameter could be added to specify an error level.

#### **Example**

Testing the error handler by trying to output variable that does not exist:

```
<?php

//error handler function

function customError($errno, $errstr)

{

    echo "<b>Error:</b> [$errno] $errstr";

}

//set error handler
set_error_handler("customError");
```

The output of the code above should be something like this:

```
Error: [8] Undefined variable: test
```

### Trigger an Error

In a script where users can input data it is useful to trigger errors when an illegal input occurs. In PHP, this is done by the `trigger_error()` function.

Example

In this example an error occurs if the "test" variable is bigger than "1":

```
<?php

$test=2;

if ($test>1)

{
```

The output of the code above should be something like this:



**Notice:** Value must be 1 or below in  
C:\webfolder\test.php on line 6

An error can be triggered anywhere you wish in a script, and by adding a second parameter, you can specify what error level is triggered.

### Possible error types:

- E\_USER\_ERROR - Fatal user-generated run-time error. Errors that can not be recovered from. Execution of the script is halted
- E\_USER\_WARNING - Non-fatal user-generated run-time warning. Execution of the script is not halted
- E\_USER\_NOTICE - Default. User-generated run-time notice. The script found something that might be an error, but could also happen when running a script normally

### 2. Example

3. In this example an E\_USER\_WARNING occurs if the "test" variable is bigger than "1". If an E\_USER\_WARNING occurs we will use our custom error handler and end

```
<?php

//error handler function

function customError($errno, $errstr)

{

    echo "<b>Error:</b> [$errno] $errstr<br />"; echo

    "Ending Script";

    die();

}

//set error handler

set_error_handler("customError",E_USER_WARNING);

//...
```

the script:

4.

5. The output of the code above should be something like this:

**Error: [512] Value must be 1 or below**  
Ending Script

6.

## 7. Error Logging

8. By default, PHP sends an error log to the servers logging system or a file, depending on how the error\_log configuration is set in the php.ini file. By using the error\_log() function you can send error logs to a specified file or a remote destination.

9. Sending errors messages to yourself by e-mail can be a good way of getting notified of specific errors.

## Send an Error Message by E-Mail

```
<?php

//error handler function

function customError($errno,$errstr)

{

    echo "<b>Error:</b> [$errno] $errstr<br/>"; echo
    "Webmaster has been notified";
    error_log("Error: [$errno]$errstr",1,
    "someone@example.com","From: webmaster@example.com");

}

//set error handler
set_error_handler("customError",E_USER_WARNING);

//trigger error
```

In the example below we will send an e-mail with an error message and end the script, if a specific error occurs:

Error: [512] Value must be 1 or below  
Webmaster has been notified

The output of the code above should be something like this:

Error: [512] Value must be 1 or below

And the mail received from the code above looks like this:

This should not be used with all errors. Regular errors should be logged on the server using the default PHP logging system.

Exceptions are used to change the normal flow of a script if a specified error occurs

### What is an Exception

With PHP 5 came a new object oriented way of dealing with errors.

Exception handling is used to change the normal flow of the code execution if a specified error (exceptional) condition occurs. This condition is called an exception.

This is what normally happens when an exception is triggered:

- The current code state is saved
- The code execution will switch to a predefined (custom) exception handler function
- Depending on the situation, the handler may then resume the execution from the saved code state, terminate the script execution or continue the script from a different location in the code

We will show different error handling methods:

- Basic use of Exceptions
- Creating a custom exception handler
- Multiple exceptions
- Re-throwing an exception
- Setting a top level exception handler

**Note:** Exceptions should only be used with error conditions, and should not be used to jump to another place in the code at a specified point.

## Basic Use of Exceptions

When an exception is thrown, the code following it will not be executed, and PHP will try to find the matching "catch" block.

If an exception is not caught, a fatal error will be issued with an "Uncaught Exception" message.

Lets try to throw an exception without catching it:

```
}  
  
return true;  
  
}
```

**Fatal error:** Uncaught exception 'Exception'

with message 'Value must be 1 or below' in C:\webfolder\test.php:6 Stack  
trace: #0 C:\webfolder\test.php(12):

The code above will get an error like this:

## Try, throw and catch

To avoid the error from the example above, we need to create the proper code to handle an exception.

Proper exception code should include:

1. Try - A function using an exception should be in a "try" block. If the exception does not trigger, the code will continue as normal. However if the exception triggers, an exception is "thrown"
2. Throw - This is how you trigger an exception. Each "throw" must have at least one "catch"
3. Catch - A "catch" block retrieves an exception and creates an object containing the exception information

```
<?php

//createfunctionwithanexception
function checkNum($number)
{
    if($number>1)
    {
        throw new Exception("Value must be 1 or below");
    }
}
```

Lets try to trigger an exception with valid code:

```
//trigger exception in a "try" block
try
{
    checkNum(2);

    //If the exception is thrown, this text will not be shown
    echo 'If you see this, the number is 1 or below';
}

//catch exception
```

The code above will get an error like this:

```
Message: Value must be 1 or below
```

Example explained:

The code above throws an exception and catches it:

1. The checkNum() function is created. It checks if a number is greater than 1. If it is, an exception is thrown

2. The checkNum() function is called in a "try" block
3. The exception within the checkNum() function is thrown
4. The "catch" block retrieves the exception and creates an object (\$e) containing the exception information
5. The error message from the exception is echoed by calling \$e->getMessage() from the exception object

However, one way to get around the "every throw must have a catch" rule is to set a top level exception handler to handle errors that slip through.

### **Creating a Custom Exception Class**

Creating a custom exception handler is quite simple. We simply create a special class with functions that can be called when an exception occurs in PHP. The class must be an extension of the exception class.

```

<?php

class customException extendsException

{

public function errorMessage()

{

//error message

$errorMsg = 'Error on line '.$this->getLine().' in '.$this->getFile()

.':<b>'.$this->getMessage().</b>is not a valid E-Mail address';

return $errorMsg;

}

}

$email = "someone@example...com"; try

{

//check if

if(filter_var($email, FILTER_VALIDATE_EMAIL) === FALSE)

{

//throw exception if email is not valid

```

### Lets create an exception class:

The new class is a copy of the old exception class with an addition of the errorMessage() function. Since it is a copy of the old class, and it inherits the properties and methods from the old class, we can use the exception class methods like getLine() and getFile() and getMessage().

Example explained:

The code above throws an exception and catches it with a custom exception class:

1. The customException() class is created as an extension of the old exception class. This way it inherits all methods and properties from the old exception class

2. The errorMessage() function is created. This function returns an error message if an e-mail address is invalid
3. The \$email variable is set to a string that is not a valid e-mail address
4. The "try" block is executed and an exception is thrown since the e-mail address is invalid
5. The "catch" block catches the exception and displays the error message

## Multiple Exceptions

It is possible for a script to use multiple exceptions to check for multiple conditions.

It is possible to use several if..else blocks, a switch, or nest multiple exceptions. These exceptions

```
<?php
class customException extends Exception
{
    public function errorMessage()
    {
        //error message
        $errorMsg = 'Error on line '.$this->getLine().' in '.$this->getFile()
        .':<b>'.$this->getMessage().</b>is not a valid E-Mail address';
        return $errorMsg;
    }
}

$email = "someone@example.com"; try
{
    //check if
    if(filter_var($email, FILTER_VALIDATE_EMAIL) === FALSE)
```

can use different exception classes and return different error



messag

```
}  
  
catch (customException $e)  
{  
    echo $e->errorMessage();  
}
```

Example explained:

The code above tests two conditions and throws an exception if any of the conditions are not met:

1. The customException() class is created as an extension of the old exception class. This way it inherits all methods and properties from the old exception class
2. The errorMessage() function is created. This function returns an error message if an e-mail address is invalid
3. The \$email variable is set to a string that is a valid e-mail address, but contains the string "example"
4. The "try" block is executed and an exception is not thrown on the first condition
5. The second condition triggers an exception since the e-mail contains the string "example"
6. The "catch" block catches the exception and displays the correct error message

If there was no customException catch, only the base exception catch, the exception would be handled there

## Re-throwing Exceptions

Sometimes, when an exception is thrown, you may wish to handle it differently than the standard way. It is possible to throw an exception a second time within a "catch" block.

A script should hide system errors from users. System errors may be important for the coder, but is of no interest to the user. To make things easier for the user you can re-throw the exception with a user friendly message

```
<?php

class customException extends Exception
{
    public function errorMessage()
    {
        //error message

        $errorMsg = 'Error on line '.$this->getLine().' in '.$this->getFile()
        .':<b>'.$this->getMessage().</b>is not a valid E-Mail address';
        return $errorMsg;
    }
}

$email = "someone@example.com"; try
{
    //check if
    if(filter_var($email, FILTER_VALIDATE_EMAIL) === FALSE)
    {
        //throw exception if email is not valid
        throw new customException($email);
    }

    //check for "example" in mail address
    if(strpos($email, "example") !== FALSE)
    {
        throw new Exception("$email is an example e-mail");
    }
}
```

Example explained:

The code above tests if the email-address contains the string "example" in it, if it does, the exception is re-thrown:

1. The `customException()` class is created as an extension of the old exception class. This way it inherits all methods and properties from the old exception class.
2. The `errorMessage()` function is created. This function returns an error message if an e-mail address is invalid
3. The `$email` variable is set to a string that is a valid e-mail address, but contains the string "example"
4. The "try" block contains another "try" block to make it possible to re-throw the exception
5. The exception is triggered since the e-mail contains the string "example"
6. The "catch" block catches the exception and re-throws a "customException"
7. The "customException" is caught and displays an error message

If the exception is not caught in its current "try" block, it will search for a catch block on "higher levels".

---

### Set a Top Level Exception Handler

The `set_exception_handler()` function sets a user-defined function to handle all uncaught exceptions.

```
<?php  
  
function myException($exception)  
{  
  
    echo "<b>Exception:</b> ", $exception->getMessage();  
  
}
```

The output of the code above should be something like this In the code above there was no "catch" block. Instead, the top level exception handler triggered. This function should be used to catch uncaught exceptions.

### Rules for exceptions

- Code may be surrounded in a try block, to help catch potential exceptions
- Each try block or "throw" must have at least one corresponding catch block
- Multiple catch blocks can be used to catch different classes of exceptions
- Exceptions can be thrown (or re-thrown) in a catch block within a try block A

simple rule: If you throw something, you have to catch it.

PHP filters are used to validate and filter data coming from insecure sources, like user input.

### What is a PHP Filter?

A PHP filter is used to validate and filter data coming from insecure sources. To test, validate and filter user input or custom data is an important part of any web application. The PHP filter extension is designed to make data filtering easier and quicker.

### Why use a Filter?

Almost all web applications depend on external input. Usually this comes from a user or another application (like a web service). By using filters you can be sure your application gets the correct input type.

### You should always filter all external data!

Input filtering is one of the most important application security issues.

## What is external data?

- Input data from a form
- Cookies
- Web services data
- Server variables
- Database query results

## Functions and Filters

To filter a variable, use one of the following filter functions:

- `filter_var()` - Filters a single variable with a specified filter
- `filter_var_array()` - Filter several variables with the same or different filters
- `filter_input` - Get one input variable and filter it
- `filter_input_array` - Get several input variables and filter them with the same or different filters

In the example below, we validate an integer using the `filter_var()` function:

```
<?php
$int = 123;

if(!filter_var($int, FILTER_VALIDATE_INT))
{
    echo("Integer is not valid");
}
```

The code above uses the "FILTER\_VALIDATE\_INT" filter to filter the variable. Since the integer is valid, the output of the code above will be: "Integer is valid".

If we try with a variable that is not an integer (like "123abc"), the output will be: "Integer is not valid".

## Validating and Sanitizing

There are two kinds of filters:

### Validating filters:

- Are used to validate user input
- Strict format rules (like URL or E-Mail validating)
- Returns the expected type on success or FALSE on failure

#### Sanitizing filters:

- Are used to allow or disallow specified characters in a string
- No data format rules
- Always return the string

#### Options and Flags

Options and flags are used to add additional filtering options to the specified filters.

Different filters have different options and flags.

In the example below, we validate an integer using the `filter_var()` and the "min\_range" and "max\_range" options:

```
<?php
$var=300;

$int_options = array(
    "options"=>array
    (
        "min_range"=>0,
        "max_range"=>256
    )
);

if(!filter_var($var, FILTER_VALIDATE_INT, $int_options))
{
    echo("Integer is not valid");
}
```

Like the code above, options must be put in an associative array with the name "options". If a flag is used it does not need to be in an array. Since the integer is "300" it is not in the specified range, and the output of the code above will be: "Integer is not valid".

## Validate Input

The first thing we need to do is to confirm that the input data we are looking for exists.

Then we filter the input data using the `filter_input()` function.

In the example below, the input variable "email" is sent to the PHP page:

```
<?php
if(!filter_has_var(INPUT_GET, "email"))
{
    echo("Input type does not exist");
}
else
{
    if (!filter_input(INPUT_GET, "email", FILTER_VALIDATE_EMAIL))
    {
        echo "E-Mail is not valid":
    }
}
```

### Example Explained

The example above has an input (email) sent to it using the "GET" method:

1. Check if an "email" input variable of the "GET" type exist
2. If the input variable exists, check if it is a valid e-mail address

## Sanitize Input

Let's try cleaning up an URL sent from a form.

First we confirm that the input data we are looking for exists. Then we sanitize the input data using the `filter_input()` function. In the example below, the input variable "url" is sent to the PHP page:

```
<?php if(!filter_has_var(INPUT_POST, "url"))
```

```
{
```



```
echo("Input type does not exist");  
  
}  
  
else  
  
{  
  
$url = filter_input(INPUT_POST,
```

### Example Explained

The example above has an input (url) sent to it using the "POST" method:

1. Check if the "url" input of the "POST" type exists
2. If the input variable exists, sanitize (take away invalid characters) and store it in the \$url variable

If the input variable is a string like this "http://www.W3ååSchøøools.com/", the \$url variable after the sanitizing will look like this:

### Filter Multiple Inputs

A form almost always consists of more than one input field. To avoid calling the filter\_var or filter\_input functions over and over, we can use the filter\_var\_array or the filter\_input\_array functions. In this example we use the filter\_input\_array() function to filter three GET variables. The received GET variables is a name, an age and an e-mail address:

```
<?php  
  
$filters = array (  
  
    "name" => array (  
  
        "filter"=>FILTER_SANITIZE_STRING  
  
    ),  
  
    "age" => array (  
  
        "filter"=>FILTER_SANITIZE_NUMBER_INT  
  
    ),  
  
    "email" => array (  
  
        "filter"=>FILTER_SANITIZE_EMAIL  
  
    )  
);
```

```

"options"=>array (
    "min_range"=>1,
    "max_range"=>120
)
),
"email"=> FILTER_VALIDATE_EMAIL,
);

$result = filter_input_array(INPUT_GET, $filters); if
(!$result["age"])
{
    echo("Age must be a number between 1 and 120.<br />");
}
elseif(!$result["email"])
{
    echo("Email must be a valid email address.");
}
}

```

### Example Explained

The example above has three inputs (name, age and email) sent to it using the "GET" method:

1. Set an array containing the name of input variables and the filters used on the specified input variables
2. Call the `filter_input_array()` function with the GET input variables and the array we just set
3. Check the "age" and "email" variables in the `$result` variable for invalid inputs. (If any of the input variables are invalid, that input variable will be FALSE after the `filter_input_array()` function)

The second parameter of the `filter_input_array()` function can be an array or a single filter ID.

If the parameter is a single filter ID all values in the input array are filtered by the specified filter.

If the parameter is an array it must follow these rules:

- Must be an associative array containing an input variable as an array key (like the "age" input variable)
- The array value must be a filter ID or an array specifying the filter, flags and options

### Using Filter Callback

It is possible to call a user defined function and use it as a filter using the FILTER\_CALLBACK filter. This way, we have full control of the data filtering.

You can create your own user defined function or use an existing PHP function

The function you wish to use to filter is specified the same way as an option is specified. In an associative array with the name "options"

In the example below, we use a user created function to convert all "\_" to whitespaces:

```
<?php

function convertSpace($string)
{
    return str_replace("_", " ", $string);
}

$string = "Peter_is_a_great_guy!";
```

The result from the code above should look like this:

```
Peter is a great guy!
```

### Example Explained

The example above converts all "\_" to whitespaces:

1. Create a function to replace "\_" to whitespaces
2. Call the filter\_var() function with the FILTER\_CALLBACK filter and an array containing our function

3. MySQL is the most popular open-source database system.
4. What is MySQL?
5. MySQL is a database.
6. The data in MySQL is stored in database objects called tables.
7. A table is a collections of related data entries and it consists of columns and rows.
8. Databases are useful when storing information categorically. A company may have a database with the following tables: "Employees", "Products", "Customers" and "Orders".
9. Database Tables
10. A database most often contains one or more tables. Each table is identified by a name (e.g. "Customers" or "Orders"). Tables contain records (rows) with data.
11. Below is an example of a table called "Persons":

LastName	FirstName	Address	City
Hansen	Ola	Timoteivn 10	Sandnes
Svendson	Tove	Borgvn 23	Sandnes
Pettersen	Kari	Storgt 20	Stavanger

12. The table above contains three records (one for each person) and four columns (LastName, FirstName, Address, and City).
13. Queries
14. A query is a question or a request.
15. With MySQL, we can query a database for specific information and have a recordset returned.
16. Look at the following query:

```
SELECT LastName FROM Persons
```

17. The query above selects all the data in the "LastName" column from the "Persons" table, and will return a recordset like this:

LastName
Hansen

Svendson

Pettersen

18. Download MySQL Database

19. If you don't have a PHP server with a MySQL Database, you can download MySQL for free here: <http://www.mysql.com/downloads/index.html>

20. Facts About MySQL Database

21. One great thing about MySQL is that it can be scaled down to support embedded database applications. Perhaps it is because of this reputation that many people believe that MySQL can only handle small to medium-sized systems.

22. The truth is that MySQL is the de-facto standard database for web sites that support huge volumes of both data and end users (like Friendster, Yahoo, Google).

23. Look at <http://www.mysql.com/customers/> for an overview of companies using MySQL.

24. The free MySQL database is very often used with PHP.

25. Create a Connection to a MySQL Database

26. Before you can access data in a database, you must create a connection to the database.

27. In PHP, this is done with the `mysql_connect()` function.

28.

```
mysql_connect(servername,username,password);
```

## Syntax

Parameter	Description
servername	Optional. Specifies the server to connect to. Default value is "localhost:3306"
username	Optional. Specifies the username to log in with. Default value is the name of the user that owns the server process
password	Optional. Specifies the password to log in with. Default is ""

Note: There are more available parameters, but the ones listed above are the most important.

## Example

In the following example we store the connection in a variable (\$con) for later use in the script. The "die" part will be executed if the connection fails:

```
<?php

$con = mysql_connect("localhost","peter","abc123"); if
(!$con)

{

    die('Could not connect: ' . mysql_error());

}
```

### Closing a Connection

The connection will be closed automatically when the script ends. To close the connection before, use the mysql\_close() function:

```
<?php

$con = mysql_connect("localhost","peter","abc123"); if
(!$con)

{

    die('Could not connect: ' . mysql_error());

}

// some code
```

A database holds one or multiple tables. Create a Database

The CREATE DATABASE statement is used to create a database in MySQL.

```
CREATE DATABASE database_name
```

### Syntax

To learn more about SQL, please visit our [SQL tutorial](#).

To get PHP to execute the statement above we must use the `mysql_query()` function.

This function is used to send a query or command to a MySQL connection.

### Example

```
<?php
$con = mysql_connect("localhost","peter","abc123"); if
(!$con)
{
    die('Could not connect: ' . mysql_error());
}

if (mysql_query("CREATE DATABASE my_db",$con))
{
    echo "Database created";
}
else
```

The following example creates a database called "my\_db":

#### Create a Table

The CREATE TABLE statement is used to create a table in MySQL.



```
CREATE TABLE table_name (  
  
column_name1    data_type,  
column_name2    data_type,  
column_name3    data_type,  
  
....
```

#### Syntax

We must add the CREATE TABLE statement to the mysql\_query() function to execute the command.

#### Example

The following example creates a table named "Persons", with three columns. The column names will be "FirstName", "LastName" a

```
<?php  
  
$con = mysql_connect("localhost","peter","abc123"); if  
(!$con)  
{  
    die('Could not connect: ' . mysql_error());  
}  
  
// Create database  
  
if (mysql_query("CREATE DATABASE my_db",$con))  
{  
    echo "Database created";  
}  
  
else
```

```
$sql = "CREATE TABLE Persons (
```

```
FirstName    varchar(15),
```

```
LastName     varchar(15),
```

```
Age int
```

```
);
```

```
// Execute query
```

```
mysql_query($sql,$con);
```

**Important:** A database must be selected before a table can be created. The database is selected with the `mysql_select_db()` function.

**Note:** When you create a database field of type `varchar`, you must specify the maximum length of the field, e.g. `varchar(15)`.

The data type specifies what type of data the column can hold.

---

### Primary Keys and Auto Increment Fields

Each table should have a primary key field.

A primary key is used to uniquely identify the rows in a table. Each primary key value must be unique within the table. Furthermore, the primary key field cannot be null because the database engine requires a value to locate the record.

The following example sets the `personID` field as the primary key field. The primary key field is often an ID number, and is often used with the `AUTO_INCREMENT` setting. `AUTO_INCREMENT` automatically increases the value of the field by 1 each time a new record is added. To ensure that the primary key field cannot be null, we must add the

```
$sql = "CREATE TABLE Persons (
```

NOT

---

```
personID int NOT NULL AUTO_INCREMENT, PRIMARY
KEY(personID),

FirstName varchar(15),
LastName varchar(15),
Age int

);

mysql_query($sql,$con);
```

The INSERT INTO statement is used to insert new records in a table.

### Insert Data Into a Database Table

The INSERT INTO statement is used to add new records to a database table.

#### Syntax

It is possible to write the INSERT INTO statement in two forms.

```
INSERT INTO table_name

VALUES (value1, value2,value3,...)
```

```
INSERT INTO table_name (column1, column2, column3,...) VALUES
(value1, value2, value3,...)
```

The first form doesn't specify the column names where the data will be inserted, only their values:

The second form specifies both the column names and the values to be inserted:

To get PHP to execute the statements above we must use the `mysql_query()` function. This function is used to send a query or command to a MySQL connection.

#### Example

In the previous chapter we created a table named "Persons", with three columns; "Firstname", "Lastname" and "Age". We will use the same table in this example. The following example adds two new records to the "Persons" table:

## Insert Data From a Form Into a Database

Now we will create an HTML form that can be used to add new records to the "Persons" table.

Here is the HTML form:

```
<html>

<body>

    <form action="insert.php" method="post"> Firstname:
```

```
Age: <input type="text" name="age" />
```

```
<input type="submit" />
```

```
</form>
```

When a user clicks the submit button in the HTML form in the example above, the form data is sent to "insert.php".

The "insert.php" file connects to a database, and retrieves the values from the form with the PHP \$\_POST variables.

Then, the mysql\_query() function executes the INSERT INTO statement, and a new record will be added to the "Persons" table.

तेजस्वि नावधीतमस्तु  
ISO 9001:2015 & 14001:2015

Here is the "insert.php" page:

```
<?php

$con = mysql_connect("localhost","peter","abc123"); if
(!$con)

{
    die('Could not connect: ' . mysql_error());
}

mysql_select_db("my_db", $con);

$sql="INSERT INTO Persons(FirstName,LastName,Age)
VALUES

('$_POST[firstname]','$_POST[lastname]','$_POST[age]')";
if (!mysql_query($sql,$con))

?>
```

The SELECT statement is used to select data from a database.

### Select Data From a Database Table

The SELECT statement is used to select data from a database.

```
SELECT column_name(s)
FROM table_name
```

#### Syntax

To get PHP to execute the statement above we must use the mysql\_query() function.

This function is used to send a query or command to a MySQL connection.

#### Example

The following example selects all the data stored in the "Persons" table (The \* character selects all the data in the table):

---

```
<?php
$con = mysql_connect("localhost","peter","abc123"); if
<?php
(!$con)
$con = mysql_connect("localhost","peter","abc123"); if
(!$con)
    die('Could not connect: ' . mysql_error());
    {
    }
    die('Could not connect: ' . mysql_error());
mysql_select_db("my_db", $con);
}

mysql_select_db("my_db", $con);
$result = mysql_query("SELECT * FROM Persons");
```

```
}
```

```
mysql_close($con);
```

The example above stores the data returned by the `mysql_query()` function in the `$result` variable.

Next, we use the `mysql_fetch_array()` function to return the first row from the recordset as an array. Each call to `mysql_fetch_array()` returns the next row in the recordset. The while loop loops through all the records in the record set. To print the value of each row, we use the PHP

`$row` variable (`$row['FirstName']` and `$row['LastName']`). The output of the code above will be:

```
Peter Griffin
Glenn Quagmire
```

### Display the Result in an HTML Table

The following example selects the same data as the example above, but will display the data in an HTML table:



```

<tr>

<th>Firstname</th>

<th>Lastname</th>

</tr>";

while($row = mysql_fetch_array($result))
{

    echo "<tr>";

    echo "<td>" . $row['FirstName'] . "</td>";
    echo "<td>" . $row['LastName'] . "</td>";
    echo "</tr>";

```

The output of the code above will be:

Firstname	Lastname
Glenn	Quagmire
Peter	Griffin

The WHERE clause is used to filter records.

### The WHERE clause

The WHERE clause is used to extract only those records that fulfill a specified criterion.

```

SELECT column_name(s)
FROM table_name
WHERE column_name operator value

```

### Syntax

To get PHP to execute the statement above we must use the `mysql_query()` function.

This function is used to send a query or command to a MySQL connection.

### Example

The following example selects all rows from the "Persons" table where "FirstName='Peter':

The output of the code above will be:

Peter Griffin

The ORDER BY keyword is used to sort the data in a recordset.

### **The ORDER BY Keyword**

The ORDER BY keyword is used to sort the data in a recordset.

The ORDER BY keyword sort the records in ascending order by default.

If you want to sort the records in a descending order, you can use the DESC keyword.

```
SELECT column_name(s)
FROM table_name

ORDER BY column_name(s) ASC|DESC
```

### **Syntax**

### **Example**

The following example selects all the data stored in the "Persons" table, and sorts the result by the "Age" column:

तेजस्वि नावधीतमस्तु  
ISO 9001:2015 & 14001:2015



```

<?php

$con = mysql_connect("localhost","peter","abc123"); if
(!$con)

{
    die('Could not connect: ' . mysql_error());
}

mysql_select_db("my_db", $con);

$result=mysql_query("SELECT * FROM Persons ORDER BY age");

while($row =mysql_fetch_array($result))
{
    echo $row['FirstName'];
    echo " " . $row['LastName'];
}

```

The output of the code above will be:

Glenn Quagmire 33

Peter Griffin 35

### Order by Two Columns

It is also possible to order by more than one column. When ordering by more than one column, the second column is only used if the values in the first column are equal:

```

SELECT column_name(s)
FROM table_name

ORDER BY column1, column2

```

```
<?php
$con = mysql_connect("localhost","peter","abc123"); if
(!$con)
{
    die('Could not connect: ' . mysql_error());
}

mysql_select_db("my_db", $con);

mysql_query("UPDATE Persons SET Age='36'")
```

The UPDATE statement is used to modify data in a table.

### Update Data In a Database

The UPDATE statement is used to update existing records in a table.

```
UPDATE table_name
SET column1=value, column2=value2,... WHERE
some_column=some_value
```

### Syntax

Note: Notice the WHERE clause in the UPDATE syntax. The WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated! To get PHP to execute the statement above we must use the mysql\_query() function. This function is used to send a query or command to a MySQL connection.

### Example

Earlier in the tutorial we created a table named "Persons". Here is how it looks:

FirstName	LastName	Age
Peter	Griffin	35
Glenn	Quagmire	33

The following example updates some data in the "Persons" table:

After the update, the "Persons" table will look like this:

FirstName	LastName	Age
Peter	Griffin	36
Glenn	Quagmire	33

The DELETE statement is used to delete records in a table.

### Delete Data In a Database

```
DELETE FROM table_name  
WHERE some_column =some_value
```

The DELETE FROM statement is used to delete records from a database table.

### Syntax

Note: Notice the WHERE clause in the DELETE syntax. The WHERE clause specifies which record or records that should be deleted. If you omit the WHERE clause, all records will be deleted! To get PHP to execute the statement above we must use the `mysql_query()` function. This function is used to send a query or command to a MySQL connection.

### Example

Look at the following "Persons" table:

FirstName	LastName	Age
Peter	Griffin	35
Glenn	Quagmire	33

The following example deletes all the records in the "Persons" table where LastName='Griffin':

```
<?php
$con = mysql_connect("localhost","peter","abc123"); if
(!$con)
{
die('Could not connect: ' . mysql_error());
}
```

```
mysql_close($con);
```

After the deletion, the table will look like this:

FirstName	LastName	Age
Glenn	Quagmire	33

ODBC is an Application Programming Interface (API) that allows you to connect to a data source (e.g. an MS Access database).

### Create an ODBC Connection

With an ODBC connection, you can connect to any database, on any computer in your network, as long as an ODBC connection is available.

Here is how to create an ODBC connection to a MS Access Database:

1. Open the **Administrative Tools** icon in your Control Panel.
2. Double-click on the **Data Sources (ODBC)** icon inside.
3. Choose the **System DSN** tab.
4. Click on **Add** in the System DSN tab.
5. Select the **Microsoft Access Driver**. Click **Finish**.
6. In the next screen, click **Select** to locate the database.
7. Give the database a **Data Source Name (DSN)**.
8. Click **OK**.

9. Note that this configuration has to be done on the computer where your web site is located. If you are running Internet Information Server (IIS) on your own computer, the instructions above will work, but if your web site is located on a remote server, you have to have physical access to that server, or ask your web host to set up a DSN for you to use.

### 10. Connecting to an ODBC

11. The `odbc_connect()` function is used to connect to an ODBC data source. The function takes four parameters: the data source name, username, password, and an optional cursor type.

12. The `odbc_exec()` function is used to execute an SQL statement.

### 13. Example

14. The following example creates a connection to a DSN called northwind, with no

```
$conn=odbc_connect('northwind','');  
$sql="SELECT * FROM customers";  
$rs=odbc_exec($conn,$sql);
```

username and no password. It then creates an SQL and executes it:

### 15. Retrieving Records

16. The `odbc_fetch_row()` function is used to return records from the result-set. This function returns true if it is able to return rows, otherwise false.

```
odbc_fetch_row($rs)
```

17. The function takes two parameters: the ODBC result identifier and an optional row number:

### 18. Retrieving Fields from a Record

19. The `odbc_result()` function is used to read fields from a record. This function takes two parameters: the ODBC result identifier and a field number or name.

20. The code line below returns the value of the first field from the record:

```
$compname=odbc_result($rs,1);
```

21.

22. The code line below returns the value of a field called "CompanyName":

```
$compname=odbc_result($rs,"CompanyName");
```

23.

#### **24. Closing an ODBC Connection**

```
odbc_close($conn);
```

25. The odbc\_close() function is used to close an ODBC connection.

#### **26. An ODBC Example**

27. The following example shows how to first create a database connection, then a result-set, and then display the data in an HTML table.

तेजस्वि नावधीतमस्तु  
ISO 9001:2015 & 14001:2015



```

<html>

<body>

<?php

$conn=odbc_connect('northwind',''); if
(!$conn)

    {exit("Connection Failed: " . $conn);}

$sql="SELECT * FROM customers";

$rs=odbc_exec($conn,$sql); if
(!$rs)

    {exit("ErrorinSQL");}
echo "<table><tr>";

echo "<th>Companyname</th>";
echo "<th>Contactname</th></tr>";
while (odbc_fetch_row($rs))

{

    $compname=odbc_result($rs,"CompanyName");

```

```
?>
```

```
</body>
```

28.

29.

### 30. PHP Array Introduction

31. The array functions allow you to manipulate arrays.

32. PHP supports both simple and multi-dimensional arrays. There are also specific functions for populating arrays from database queries.

### 33. PHP Array Functions

---

34. **PHP:** indicates the earliest version of PHP that supports the function.

Function	Description	PHP
<a href="#"><u>array()</u></a>	Creates an array	3
<a href="#"><u>array_change_key_case()</u></a>	Returns an array with all keys in lowercase or uppercase	4
<a href="#"><u>array_chunk()</u></a>	Splits an array into chunks of arrays	4
<a href="#"><u>array_combine()</u></a>	Creates an array by using one array for keys and another for its values	5
<a href="#"><u>array_count_values()</u></a>	Returns an array with the number of occurrences for each value	4
<a href="#"><u>array_diff()</u></a>	Compares array values, and returns the differences	4
<a href="#"><u>array_diff_assoc()</u></a>	Compares array keys and values, and returns the differences	4
<a href="#"><u>array_diff_key()</u></a>	Compares array keys, and returns the differences	5
<a href="#"><u>array_diff_uassoc()</u></a>	Compares array keys and values, with an additional user- made function check, and returns the differences	5
<a href="#"><u>array_diff_ukey()</u></a>	Compares array keys, with an additional user-made function check, and returns the differences	5
<a href="#"><u>array_fill()</u></a>	Fills an array with values	4
<a href="#"><u>array_filter()</u></a>	Filters elements of an array using a user-made function	4
<a href="#"><u>array_flip()</u></a>	Exchanges all keys with their associated values in an array	4
<a href="#"><u>array_intersect()</u></a>	Compares array values, and returns the matches	4
<a href="#"><u>array_intersect_assoc()</u></a>	Compares array keys and values, and returns the matches	4
<a href="#"><u>array_intersect_key()</u></a>	Compares array keys, and returns the matches	5
<a href="#"><u>array_intersect_uassoc()</u></a>	Compares array keys and values, with an additional user- made function check, and returns the matches	5



<a href="#"><u>array_intersect_ukey()</u></a>	Compares array keys, with an additional user-made function check, and returns the matches	5
<a href="#"><u>array_key_exists()</u></a>	Checks if the specified key exists in the array	4
<a href="#"><u>array_keys()</u></a>	Returns all the keys of an array	4
<a href="#"><u>array_map()</u></a>	Sends each value of an array to a user-made function, which returns new values	4
<a href="#"><u>array_merge()</u></a>	Merges one or more arrays into one array	4
<a href="#"><u>array_merge_recursive()</u></a>	Merges one or more arrays into one array	4
<a href="#"><u>array_multisort()</u></a>	Sorts multiple or multi-dimensional arrays	4
<a href="#"><u>array_pad()</u></a>	Inserts a specified number of items, with a specified value, to an array	4
<a href="#"><u>array_pop()</u></a>	Deletes the last element of an array	4
<a href="#"><u>array_product()</u></a>	Calculates the product of the values in an array	5
<a href="#"><u>array_push()</u></a>	Inserts one or more elements to the end of an array	4
<a href="#"><u>array_rand()</u></a>	Returns one or more random keys from an array	4
<a href="#"><u>array_reduce()</u></a>	Returns an array as a string, using a user-defined function	4
<a href="#"><u>array_reverse()</u></a>	Returns an array in the reverse order	4
<a href="#"><u>array_search()</u></a>	Searches an array for a given value and returns the key	4
<a href="#"><u>array_shift()</u></a>	Removes the first element from an array, and returns the value of the removed element	4
<a href="#"><u>array_slice()</u></a>	Returns selected parts of an array	4
<a href="#"><u>array_splice()</u></a>	Removes and replaces specified elements of an array	4
<a href="#"><u>array_sum()</u></a>	Returns the sum of the values in an array	4

<a href="#"><u>array_udiff()</u></a>	Compares array values in a user-made function and returns an array	5
<a href="#"><u>array_udiff_assoc()</u></a>	Compares array keys, and compares array values in a user-made function, and returns an array	5
<a href="#"><u>array_udiff_uassoc()</u></a>	Compares array keys and array values in user-made functions, and returns an array	5
<a href="#"><u>array_uintersect()</u></a>	Compares array values in a user-made function and returns an array	5
<a href="#"><u>array_uintersect_assoc()</u></a>	Compares array keys, and compares array values in a user-made function, and returns an array	5
<a href="#"><u>array_uintersect_uassoc()</u></a>	Compares array keys and array values in user-made functions, and returns an array	5
<a href="#"><u>array_unique()</u></a>	Removes duplicate values from an array	4
<a href="#"><u>array_unshift()</u></a>	Adds one or more elements to the beginning of an array	4
<a href="#"><u>array_values()</u></a>	Returns all the values of an array	4
<a href="#"><u>array_walk()</u></a>	Applies a user function to every member of an array	3
<a href="#"><u>array_walk_recursive()</u></a>	Applies a user function recursively to every member of an array	5
<a href="#"><u>arsort()</u></a>	Sorts an array in reverse order and maintain index association	3
<a href="#"><u>asort()</u></a>	Sorts an array and maintain index association	3
<a href="#"><u>compact()</u></a>	Create array containing variables and their values	4
<a href="#"><u>count()</u></a>	Counts elements in an array, or properties in an object	3
<a href="#"><u>current()</u></a>	Returns the current element in an array	3
<a href="#"><u>each()</u></a>	Returns the current key and value pair from an array	3

<u>end()</u>	Sets the internal pointer of an array to its last element	3
<u>extract()</u>	Imports variables into the current symbol table from an array	3
<u>in_array()</u>	Checks if a specified value exists in an array	4
<u>key()</u>	Fetches a key from an array	3
<u>krsort()</u>	Sorts an array by key in reverse order	3
<u>ksort()</u>	Sorts an array by key	3
<u>list()</u>	Assigns variables as if they were an array	3
<u>natcasesort()</u>	Sorts an array using a case insensitive "natural order" algorithm	4
<u>natsort()</u>	Sorts an array using a "natural order" algorithm	4
<u>next()</u>	Advance the internal array pointer of an array	3
<u>pos()</u>	Alias of current()	3
<u>prev()</u>	Rewinds the internal array pointer	3
<u>range()</u>	Creates an array containing a range of elements	3
<u>reset()</u>	Sets the internal pointer of an array to its first element	3
<u>rsort()</u>	Sorts an array in reverse order	3
<u>shuffle()</u>	Shuffles an array	3
<u>sizeof()</u>	Alias of count()	3
<u>sort()</u>	Sorts an array	3
<u>uasort()</u>	Sorts an array with a user-defined function and maintain index association	3
<u>uksort()</u>	Sorts an array by keys using a user-defined function	3

<u>usort()</u>	Sorts an array by values using a user-defined function	3
----------------	--	---

NAAC ACCREDITED



तेजस्वि नावधीतमस्तु

ISO 9001:2015 & 14001:2015