GUDLAVALLERU ENGINEERING COLLEGE

(An Autonomous Institute with Permanent Affiliation to JNTUK, Kakinada)
Seshadri Rao Knowledge Village, Gudlavalleru – 521 356.

Department of Computer Science and Engineering

HANDOUT

on

OPERATING SYSTEMS

Vision

To be a Centre of Excellence in computer science and engineering education and training to meet the challenging needs of the industry and society.

Mission

- To impart quality education through well-designed curriculum in tune with the growing software needs of the industry.
- To serve our students by inculcating in them problem solving, leadership, teamwork skills and the value of commitment to quality, ethical behavior & respect for others.
- To foster industry-academia relationship for mutual benefit and growth.

Program Educational Objectives

PEO1 : Identify, analyze, formulate and solve Computer Science and Engineering problems both independently and in a team environment by using the appropriate modern tools.

PEO2 : Manage software projects with significant technical, legal, ethical, social, environmental and economic considerations.

PEO3 : Demonstrate commitment and progress in lifelong learning, professional development, leadership and Communicate effectively with professional clients and the public.

HANDOUT ON OPERATING SYSTEMS

Class & Sem : II B.Tech. – II Semester Year : 2019-20

Branch : CSE Credits : 3

==========

1. Brief History and Scope of the Subject

- Computer operating systems (OS) provide a set of functions needed and used by most application programs on a computer, and the links needed to control and synchronize computer hardware. On the first computers, with no operating system, every program needed the full hardware specification to run correctly and perform standard tasks, and its own drivers for peripheral devices like printers and punched paper card readers.
- Operating systems can also be considered to be managers of the resources. An operating system determines which computer resources will be utilized for solving which problem and the order in which they will be used. In general, an operating system has three principal types of functions.
- Allocation and assignment of system resources such as input/output devices, software, central processing unit, etc.
- Scheduling: This function coordinates resources and jobs and follows certain given priority.
- Monitoring: This function monitors and keeps track of the activities in the computer system. It maintains logs of job operation, notifies end-users or computer operators of any abnormal terminations or error conditions. This function also contains security monitoring features such as any authorized attempt to access the system as well as ensures that all the security safeguards are in place .
- Throughout the history of computers, the operating system has continually evolved as the needs of the users and the capabilities of the computer systems have changed.

2. Pre-Requisites

- Basic knowledge of system programs and application programs

3. Course Objectives:

- To impart the concepts of process, memory and file management techniques.
- To familiarize with the deadlock handling techniques.

4. Course Outcomes:

Upon successful completion of the course, the students will be able to

- **describe** the role, functions and structures of operating systems.
- **evaluate** the performance of CPU scheduling algorithms by calculating average waiting time and turnaround time.
- **compare and contrast** memory management schemes for efficient utilization of memory.
- **apply** deadlock prevention, avoidance and recovery techniques to keep the system in safe state.
- **determine** seek time of disk scheduling algorithms.
- **develop** software or hardware based solutions for critical section problems.
- **analyze** files and directory structures and implementations.

5. Program Outcomes:

Graduates of the Computer Science and Engineering Program will have Engineering Graduates will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and wit society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

6. Mapping of Course Outcomes with Program Outcomes:

	1	2	3	4	5	6	7	8	9	10	11	12
CO1	H	L	M	L								
CO2	H	L	M	L								
CO3	M	L	M	L								
CO4	H	L	M	L								
CO5	H	L	M	L								
CO6	H	L	M	L								
CO7	H	L	M	L								

7. Prescribed Text Books

i. Abraham Silberschatz, Peter B, Galvin, Greg Gagne, Operating System Principles, John Wiley, 7th edition.

ii. Stallings, Operating Systems - Internal and Design Principles, Pearson education, 6th edition–2005.

8. Reference Text Books

i. D. M. Dhamdhere, Operating systems- A Concept based Approach, TMH, 2nd edition.

ii. Andrew S Tanenbaum, Modern Operating Systems, PHI, 3rd edition.

9. URLs and Other E-Learning Resources

http://www.nptel.iitm.ac.in/video.php?subjectId=112106134

http://www.preservearticles.com/2012051832397/5-important-limitation-of-operations-research.html

http://www.nptel.iitm.ac.in/video.php?subjectId=112106134

http://personal.maths.surrey.ac.uk/st/J.F/chapter7.pdf

http://nptel.iitm.ac.in/syllabus/syllabus_pdf/111107064.pdf

http://nptel.iitm.ac.in/courses/110106045/

http://nptel.iitm.ac.in/syllabus/109103021/

http://nptel.iitm.ac.in/video.php?subjectId=112106131

10. Digital Learning Materials:

http://www.scribd.com/doc/39223153/Replacement-Models-Operation-Research#download

http://www.nptel.iitm.ac.in/courses/Webcourse-contents/IIT-ROORKEE/INDUSTRIAL-ENGINERRING/part3/inventory/lecture2.htm

http://www.eolss.net/sample-chapters/c02/E6-05-05-05.pdf

11. Lecture Schedule / Lesson Plan(3+1*)

Topic	No. of Periods	
	Theory	Tutorial
UNIT- 1: INRODUCTION		
Operating system operations	1	2
Operating system services	2	
System calls	1	
Types of system calls	2	
Operating –system structure	2	

UNIT-II: Process Management		
Process, Process state, Process control block (PCB)	**1**	**3**
Process scheduling	**1**	
Scheduling queues	**1**	
Schedulers	**1**	
Context switch	**1**	
Scheduling criteria	**1**	
Scheduling algorithms	**3**	
Operations on processes	**2**	
Inter process communication	**2**	
UNIT – III: Memory Management Strategies		
Swapping	**1**	**2**
Contiguous memory allocation	**1**	
Paging	**3**	
Segmentation	**1**	
Virtual-Memory Management	**1**	
Demand paging	**1**	
Page replacement Algorithms	**2**	
Allocation of Frames	**1**	
Thrashing	**1**	
UNIT - IV : Deadlocks and Mass-storage structure		
System model, Deadlock characterization	**1**	**2**
Methods for handling deadlocks:		
deadlock- prevention, Avoidance	**3**	
Detection, recovery	**1**	
Mass-storage structure:		
Overview, Disk Scheduling	**2**	
Disk Management	**2**	

UNIT - V: Synchronization		
The critical section problem	1	2
Peterson's solution	1	
Synchronization hardware	1	
Semaphores	2	
Classic problems of synchronization	2	
Monitors	2	
UNIT-VI: File system Interface		
Concept of a file	1	2
Access methods	1	
Directory structure	1	
File system mounting	1	
Files sharing and protection	1	
Total No. of periods	**56**	**13**

12. Seminar Topics

CPU Scheduling
Deadlocks
Disk Scheduling

OPERATING SYSTEMS

Unit – 1

Introduction

Objectives:

- To introduce the basic concepts and functions of various operating systems

Syllabus: Introduction

Operating system operations, Operating system services, System calls, Types of system calls, Operating –system structure.

Outcomes:

Students will be able to

- Understand the structure of Operating System
- Know the Services provided by Operating System
- Identify various System calls

LEARNING MATERIAL
UNIT-I

Definitions:

- An Operating System is a program that acts as an intermediary between a user of a computer and the computer hardware.
- An operating system is software that manages the computer hardware.
- The operating system controls the hardware and coordinates its use among the various application programs for the various users.
- An operating system is similar to a ***government***. Like a government, it performs no useful function by itself. It simply provides an *environment* within which other programs can do useful work.
- An operating system is a control program. A manages the execution of user programs to prevent errors and improper use of the computer.
- An operating system is a control program. A manages the execution of user programs to prevent errors and improper use of the computer.
- It is a set of programs previously written and stored in the memory of the computer.

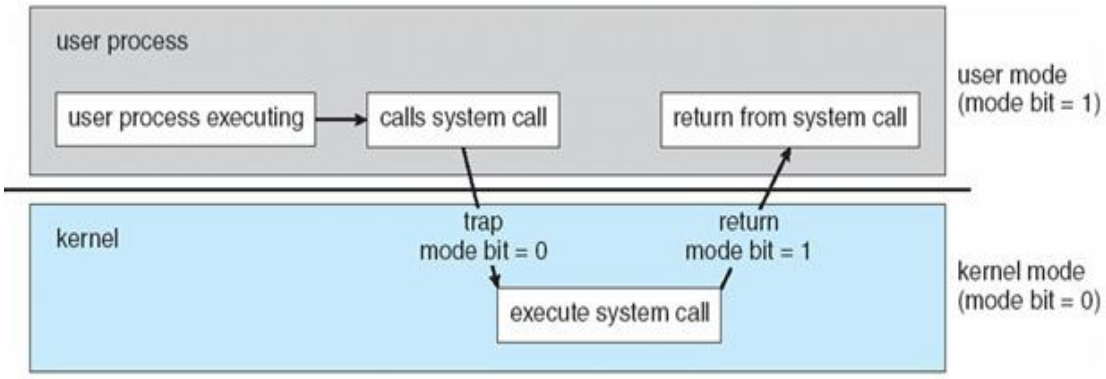
1.1 OPERATING SYSTEM OPERATIONS

- Modern operating systems are **interrupt driven.**
- If there are no processes to execute, no I/O devices to service, and no users to whom to respond, an operating system is waiting for something to happen.
- Events are almost always signaled by the occurrence of an interrupt or a trap.
- A **trap (or** an **exception)** is a software-generated interrupt caused either by an error or by a specific request from a user program.
- For each type of interrupt, separate segments of code in the operating system determine what action should be taken.

- If the operating system and the users share the hardware and software resources of the computer system.
    - With sharing, many processes could be adversely affected by a bug in one program.
    - For example, if a process gets stuck in an infinite loop, this loop could prevent the correct operation of many other processes.
    - One erroneous program might modify another program, the data of another program, or even the operating system itself.
- A properly designed operating system must ensure that an incorrect program cannot cause other programs to execute incorrectly.

❖ **Dual-Mode Operation:**

- To ensure the proper execution of the operating system, we must be able to distinguish between the execution of operating-system code and user defined code.
- There are two **modes** of operation:
    - **User mode** and
    - **Kernel mode (supervisor mode, system mode, or privileged mode).**
- A bit, called the **mode bit,** is added to the hardware of the computer to indicate the current mode: **kernel (0) or user (1)**.
- When the computer system is executing on behalf of a user application, the system is in user mode.
- When a user application requests a service from the operating system (via a system call), it must transition from user to kernel mode to fulfill the request.



- At system boot time, the hardware starts in **kernel mode**.
- The operating system is then loaded and starts user applications in user mode.
- Whenever a trap or interrupt occurs, the hardware switches from user mode to kernel mode
- Whenever the operating system gains control of the computer, it is in kernel mode.
- The dual mode of operation provides us with the means for protecting the operating system from errant users—and errant users from one another.
- We accomplish this protection by designating some of the machine instructions that may cause harm as **privileged instructions.**
- The hardware allows privileged instructions to be executed only in kernel mode.
- If an attempt is made to execute a privileged instruction in user mode, the hardware does not execute the instruction but rather treats it as illegal and traps it to the operating system.

❖ **Timer:**

- We must prevent a user program from getting stuck in an infinite loop or not calling system services and never returning control to the operating system.
- To accomplish this goal, we can use a timer.

- A timer can be set to interrupt the computer after a specified period.
- There are two types of timers
  - **Fixed timer**
  - **Variable timer**
- A **variable timer** is generally implemented by a fixed-rate clock and a counter.
- The operating system sets the counter.
  - Every time the clock ticks, the counter is decremented. When the counter reaches 0, an interrupt occurs
- Before turning over control to the user, the operating system ensures that the timer is set to interrupt.
- If the timer interrupts, control transfers automatically to the operating system.

1.2 OPERATING SYSTEM SERVICES

- An Operating System provides an environment for the execution of programs.
- It provides certain services to programs and to the users of those programs.
- The specific services provided are differ from one operating system to another.
- One set of operating system services provides functions that are helpful to the user.

❖ **User Interface**

- Almost all operating systems have a user interface. This interface can take several forms
  - **Command Line Interface:** Command Line Interface which uses text commands and a method for entering them.

- **Batch Interface:** Batch Interface in which commands and directives to control those commands are entered into files, and those are executed.
- **Graphical User Interface:** This interface is a window system with a pointing device to direct I/O, choose from menus, and make selections and a keyboard to enter text.

❖ **Program execution**

- The system must be able to load a program into memory and to run that program.
- The program must be able to end its execution, either normally or abnormally.

❖ **I/O operation**

- A running program may require I/O, which may involve file or an I/O device.
- For efficiency and protection, users usually cannot control I/O devices directly. Therefore, the operating system must provide a means to do I/O.

❖ **File System manipulation**

- The file System is of particular interest obviously, programs need to read and write files and directories.
- They also need to create and delete them by name, search for a given file, and list file information.
- Some program includes permissions management to allow or deny access to files or directories based on file ownership.

❖ **Communication**

- There are many circumstances in which one process needs to exchange information with another process.
- Such communication may occur between processes that are executing on the same computer or between processes that are

executing on different computer systems tied together by a computer network.

- Communication may be implemented via shared memory or through message passing

❖ **Error Detection:**

- The operating system needs to be constantly aware of possible errors.
- Error may occur in the CPU and memory hardware (such as a memory error or a power failure), in I/O devices.
- Debugging facilities can greatly enhance the user's and programmer's abilities to use the system efficiently.

❖ **Resource Allocation**

- When there are multiple users or multiple jobs running at a same time, resources must be allocated to each of them.
- Many different types of resources are managed by the operating system.
- Some may have special allocation code, whereas others may have much more general request and release code.

❖ **Accounting**

- We want to keep track of which users use how much and what kinds of computer resources.
- This record keeping may be used for accounting so that user billed or simply for accumulating usage statistics.

❖ **Protection and security**

- The owners of information stored in a multiuser or networked computer system may want to control use of that information.
- When several separate processes execute concurrently, it should not be possible for one process to interface with the others or with the operating system itself.

- Protection involves ensuring that all access to system resources is controlled. Security of the system from outsiders is also important.

1.3 SYSTEM CALLS

- System calls provide an interface to the services made available by an operating system.
- These system calls are generally available as routines written in C and C++.
- Certain low level tasks are written in assembly-language instructions.
- Mostly accessed by programs via a high-level Application Programming Interface (API) rather than direct system call use.
- Three 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)
  - Java API for the Java virtual machine (JVM)

Below is a sequence of system calls to copy the contents of one file to another file:

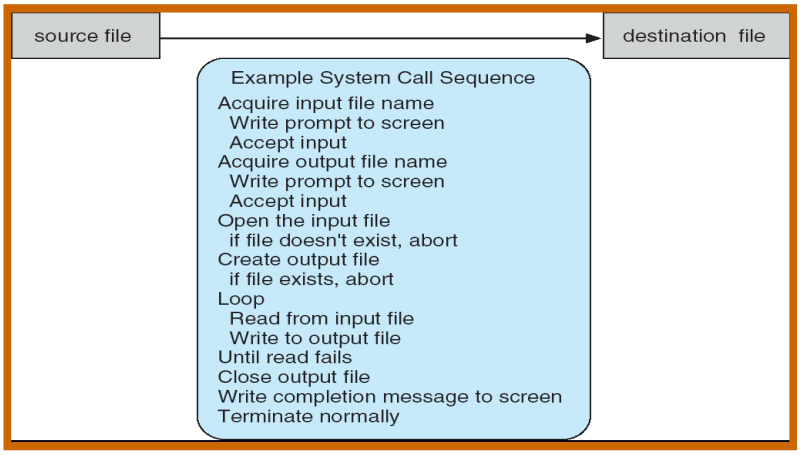


Fig: **Example of a System Call**

- Some system calls are generally in assembly language. The assembler converts assembly language to machine language and thus system calls are executed.
- Typically, there is 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)

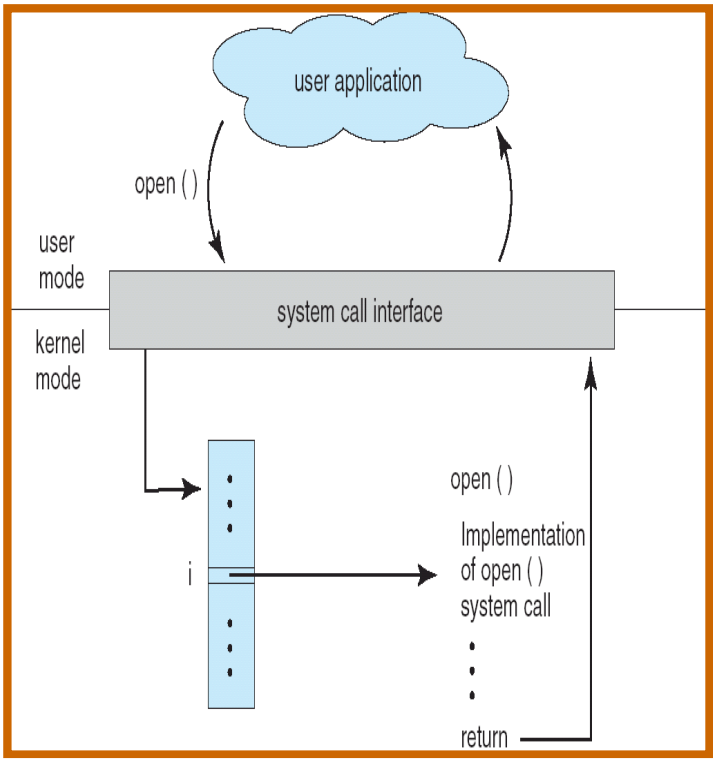


Fig: **The handling of a user application invoking the open() system call**

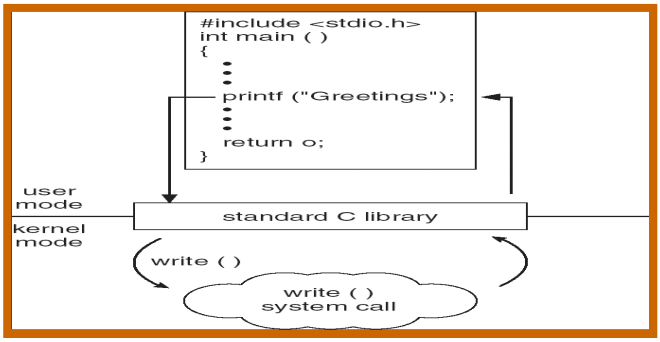


Fig: **C program invoking printf() library call, which calls the write() system call**

❖ **System Call Parameter Passing:**

- Three general methods used to pass parameters to the OS
  1. Simplest: pass the parameters in *registers*
     - In some cases, may be more parameters than registers
  2. 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
  3. 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

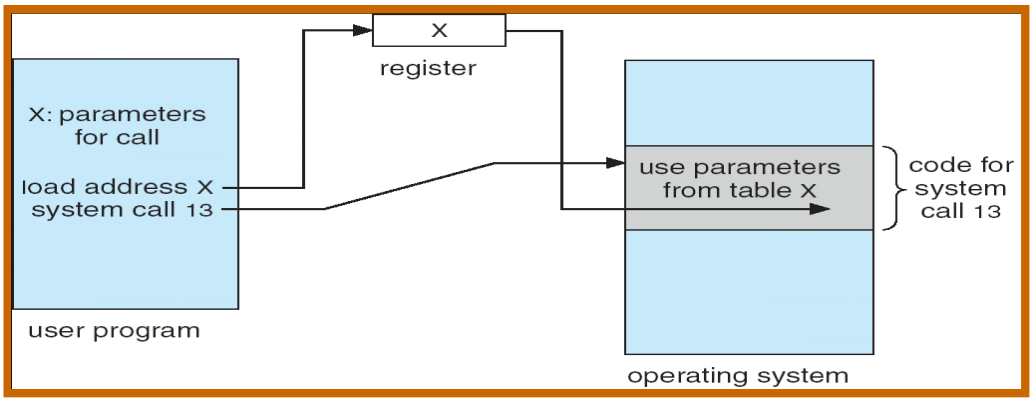


Fig: **Passing of parameters as a table**

1.4 Types of System Calls

- System calls can be grouped into five major categories:
  - ***Process control***
    - Load, execute, end, abort, create process, get/set process attributes, wait for time/signal, allocate/free memory
  - **end, abort:** The process end is used normally for the process to be end and abort is used when the errors occur in the process.
  - **load, execute:** To bring the job from secondary memory to main memory, the system call load is used and to execute particular job ,execute system call is used.
  - **create process, terminate process:** To create a process we use a system call called create process, after the complete execution of the parent process the system call terminate process is used.
  - **wait event, signal event:** Wait event makes the event wait for a while until a particular event done, and then the system call, signal event is used. Until the signal is not given, the operation is not done. Ex: Signal Clock
  - **allocate and free memory:** Allocates the resources whatever the process needs, when it is created and after its execution making the memory free.
  - ***File management (manipulation)***
    - Create/delete/open/close/read/write a file, get/set file attributes
  - **Create, delete:** Create means creating a file. Delete means deleting a file. These are the system calls given by the user, and the system process the work.
  - **Open, close:** Open means opening a file. Close means closing a file.
  - **Read, write, reposition:** read means reading the contents of a file. Write means writing the contents to a file. Reposition means

changing the position i.e., moving information from one drive to another.

- **get file attributes & set file attributes:** get file attributes means the details of the file i.e., when it is created, last modified, user name, type of file, date of creation etc., set file attributes means changing the current attributes.

❖ ***Device management***

- Request/release device, read/write data, get/set attributes

- **Request device, release device:** The request generated to do particular action, if the devices already engage and set them to release.
- **Read, write, reposition:** The system calls for read, write and reposition of a device.
- **get device attributes & set device attributes:** It is used for getting and setting the attributes **.**Ex: Settings in phone

❖ ***Information maintenance***

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

- **get time/date , set time/date**
- **get system date , set system date**
- **get process ,file ,device attributes**
- **set process ,file ,device attributes**

❖ ***Communications***

- Create/delete connection, send/receive messages, attach/detach devices

- **Create, delete:** For creating and deleting communication connection. We are having sharing on and off options between two systems.
- **Send, receive:** The send and receive system calls are used to send and receive messages based on the requirement we need.

	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() InitlializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

1.5 OPERATING SYSTEM STRUCTURE

- A System as large and complex as a modern operating system must be engineered carefully if it is to be function properly and to be modified easily.
- The common approach is to partition the task into small components.
- Each of these modules should have inputs, outputs and functions.

There are four types of operating systems structures.

1. **Simple Structure**
2. **Layered approach**
3. **Micro kernels**
4. **Modules**

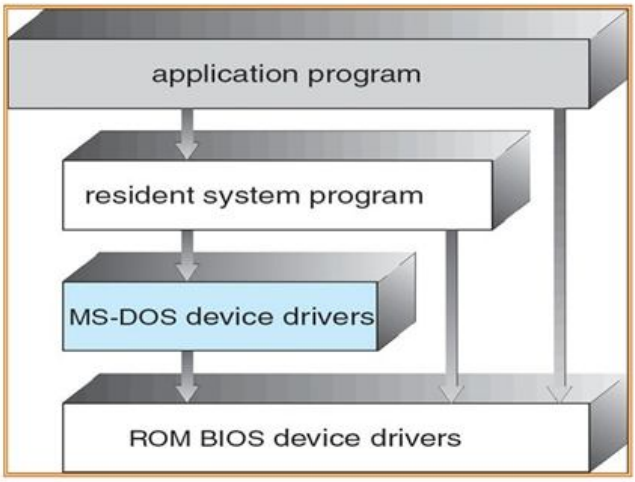
❖ Simple Structure:

- Many commercial operating systems do not have well-defined structures.

- Frequently, such systems started as small, simple, and limited systems and then grew beyond their original scope.

Example 1: MS-DOS operating system.

- It was written to provide the most functionality in the least space, so it was not divided into modules carefully.
- In MS-DOS, the interfaces and levels of functionality are not well separated.
- There is no CPU Execution Mode (user and kernel), and so errors in applications can cause the whole system to crash.

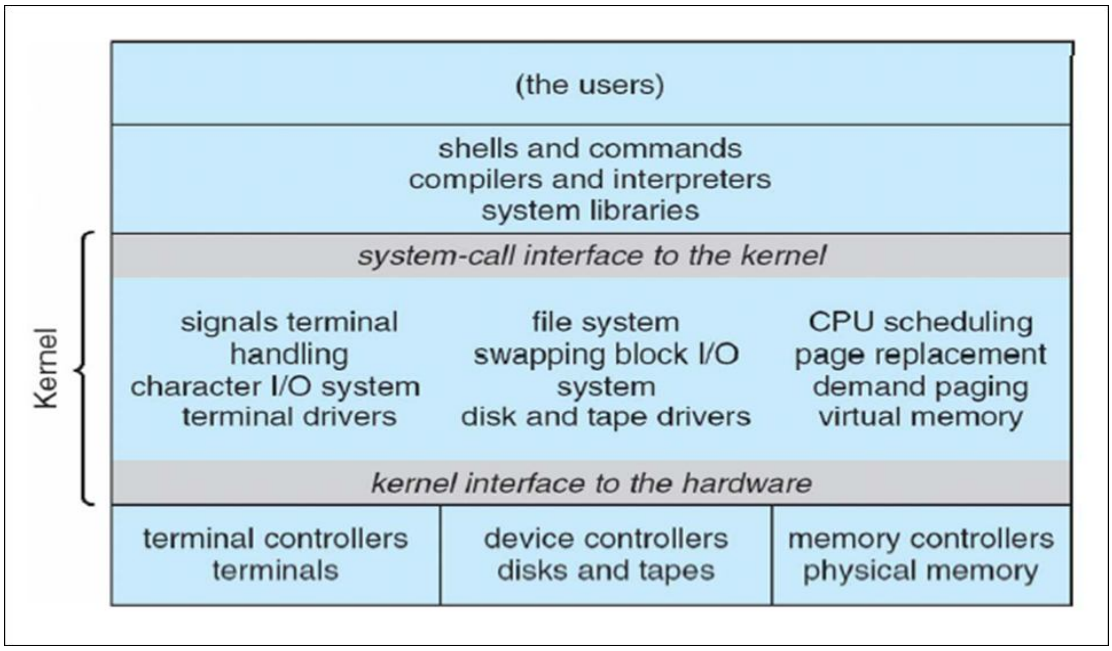


MS-DOS layer structure

Example 2: The original UNIX operating system.

- Like MS-DOS, UNIX initially was limited by hard ware functionality.
- It consists of **two separable parts**: The kernel and the system programs.
- The kernel:
  - It is further divided in to a series of interfaces and device drivers which have been added and expanded over the years as UNIX has evolved.

- Everything below the system call interface and above the physical hardware is the kernel.
- The Kernel provides the file system, CPU scheduling, Memory management and other operating system functions through system calls.
- This monolithic structure was difficult to implement and maintain.



Traditional UNIX Operating System

❖ **Layered Approach:**

- A system can be made modular in many ways.
- One method is layered approach.
  - Here, the operating system is broken into a number of layers or levels.
  - The bottom layer is an implementation of an abstract object made up of data and the operations that can manipulate those data.

- A layer of an operating system say layer M consists of data structures and a set of routines that can be invoked by higher level layers.
- Layer M in turn, can invoke operations on lower level layers.

Advantages:

1. Simplicity of construction and debugging
   - The layers are selected so that each uses functions and services of only lower-level layers.
   - This approach simplifies debugging and system verification.
   - The first layer can be debugged without any concern for the rest of the system, because it uses only the basic hardware to implement its functions.
   - Once the first layer is debugged, its correct functioning can be assumed while the second layer is debugged, and so on.
   - If an error is found during the debugging of a particular layer, the error must be on that layer, because the layers below it are already debugged.
   - Thus, the design and implementation of the system is simplified.
   - Each layer is implemented with only those operations provided by lower-level layers.
   - A layer does not need to know how these operations are implemented; it needs to know only what these operations do.
   - Each layer hides the existence of certain data structures, operations, and hardware from higher-lever layers.

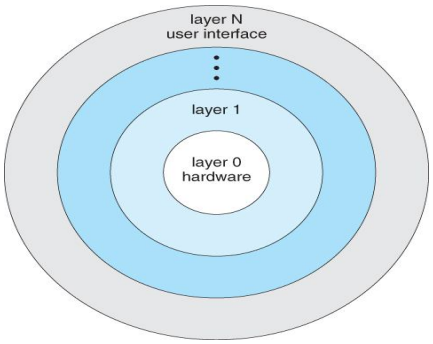
Disadvantages:

- The major difficulty with this approach involves appropriately defining the various layers. Because a layer can use only lower-level layers, careful planning is necessary.

- **Example**: The device driver for the backing store must be at a lower level than the memory-management routines. Because, memory management requires the ability to use the backing store.
- Final problem with layered implementations is that they tend to be less efficient than other types.

Example:

- When a user program executes an I/O operation, It executes a system call that is trapped to the I/O layer, which calls the memory-management layer, which in turn calls the CPU-scheduling layer, which is then passed to the hardware.
- At each layer, the parameters may be modified; data may need to be passed, and so on.
- Each layer adds overhead to the system call; the net result is a system call that takes longer than does one on a non layered system.



Layered Operating system

❖ **Micro kernels:**

- In the mid-1980s, researchers at Carnegie Mellon University developed an operating system called Mach that modularized the kernel using the Microkernel approach.

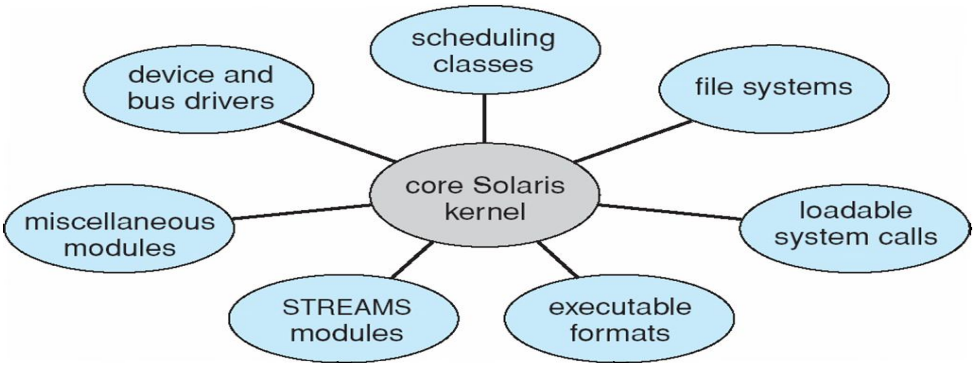
- This method structures the operating system by removing all nonessential components from the kernel and implementing them as system and user-level programs.
- The result is a smaller kernel.
- There is a little consensus that which services should remain in the kernel and which should be implemented in user space.
- The main function of the microkernel is to provide a **communication facility**.
- This communication facility is provided between the client program and the various services that are also running in user space.
- Communication is provided by message passing.
- Example: If the client program wishes to access a file, it must interact with file server. The client program and service never interact directly. Rather, they communicate indirectly by exchanging messages with the microkernel.
- **Advantages** :
    - One benefit of the microkernel approach is ease of extending the operating system.
    - All new services are added to user space and consequently do not require modification of the kernel.
    - The microkernel also provides more security and reliability.
- Some operating systems that have used micro kernel approach: Tru64UNIX, QNX.
- **Drawback:**
    - Micro kernels can suffer from performance decreases due to increased system function overhead.

❖ **Modules:**

- The best current methodology for operating system design involves using object-oriented programming techniques to create a modular kernel.
- The kernel has a set of core components and dynamically links in additional services either during boot time or run time.
- Such a strategy uses dynamically loadable modules and is common in modern implementations of UNIX such as Solaris, Linux and Mac OS X.
- Example 1: Solaris operating system structure:

  It is organized around a core kernel with seven types of loadable kernel modules.

  1. Scheduling classes.
  2. File systems.
  3. Loadable system calls.
  4. Executable formats.
  5. STREAMS modules.
  6. Miscellaneous.
  7. Device and Bus drivers.



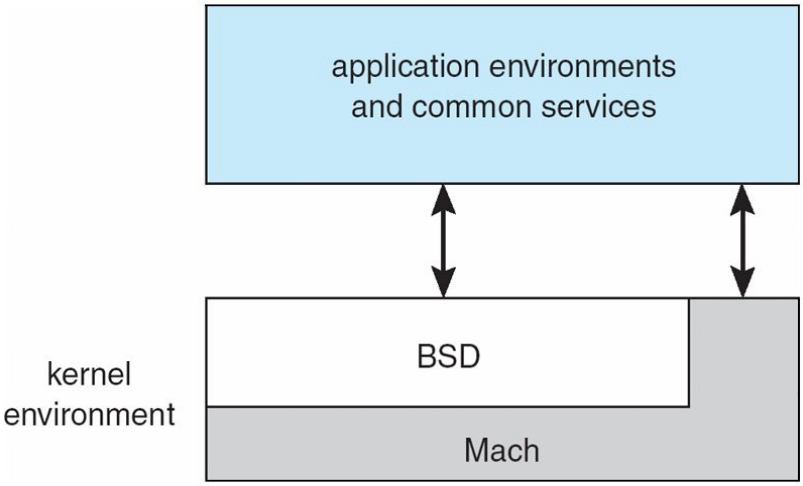
Solaris loadable module

Advantages:

1. It is more flexible than layered system in that any module can call any other module.
2. It is more efficient, because modules do not need to invoke message passing in order to communicate.

Example 2: Mac OS X Structure

- The Apple Macintosh Mac OS X operating system uses a hybrid structure.
- Mac OS X structures the operating system using a layered technique where one layer consists of the Mach microkernel.
- The top layer includes application environments and a set of services providing a graphical interface to applications.
- Below these layers is the kernel environment, which consists primarily of the Mach microkernel and the BSD kernel.
- Mach provides memory management:
    - Support for remote procedure calls.
    - Inter process communication.
    - Message passing.
    - Thread scheduling.
- BSD component provides
    - BSD command line interface
    - Support for networking and file systems.
    - Implementation of POSIX APIs, including Pthreads.
- In addition to Mach and BSD, the kernel environment provides an I/O kit for development of device drivers and dynamically loadable modules.


application environments
and common services
BSD
kernel
environment
Mach

****

UNIT-I
Assignment-Cum-Tutorial Questions
SECTION-A

Objective Questions

1. An _____ acts as an interface between the user and the computer system.
2. Which concept explains the working of an Operating System? [ ]
   a) It is event driven
   b) It is object oriented
   c) It is procedure based system software
   d) It is a collection of procedures that interact with each other
3. A kernel is an essential part of an operating system [True/False]
4. Which of these is/are the desirable features of an Operating system
   a) Extensible b) Portable c) Reliable d) All [ ]
5. Which one of the following is the mode bit associated for user mode and kernel mode respectively [ ]
   a) 1 and 0 b) 0 and 1 b)1 and 2 d) 2 and 1
6. CPU has two modes: privileged and non-privileged. In order to change the mode from privileged to non-privileged **(GATE-2001)**
   a) a hardware interrupt is needed. [ ]
   b) a software interrupt is needed.
   c) a privileged instruction (which does not generate an interrupt) is needed.
   d) a non-privileged instruction (which does not generate an interrupt) is needed.
7. _____ is a mechanism which involves in ensuring that all access to system resources is controlled.
8. Some of the important activities that an Operating System performs [ ]
   a) Job accounting b) Security
   c) Error detecting aids d) All of these

9. Which of the following system calls are used to maintain system information.
   a) Get/set time or date. [ ]
   b) request device, release device.
   c) send, receive messages.
   d) get process attributes, set process attributes.
10. _____ provides an interface to the services made available by an operating system.
11. In which of the following users do not interact with the computer directly [ ]
   a) Batch operating system b) DOS operating system
   c) Time-sharing Operating Systems d) None of these
12. Which of the following functionality is provided by micro kernel approach
   a) Communication. [ ]
   b) ease of extending of an operating system.
   c) reliability and security.
   d) All of the above.
13. _____ and _____ are two fundamental models of implementing communication.
14. One function of an operating system is to handle interrupts. Interrupts are [ ]
   a) a delay in processing due to operating system overload
   b) signals from hardware or software requesting attention from the operating system
   c) messages received from other computers
   d) None of the above.

14.System calls are invoked by using (**NPTEL/GATE1999**)
   a) software interrupt b) polling [ ]
   c) indirect jump d) a privileged instruction

SECTION-B

Descriptive Questions

1. Define operating system. Explain the **operations** of an operating system?
2. With a neat sketch explain the **Dual-mode** operation?
3. Explain the need of attaching **timer** in operating system?
4. With a neat sketch explain the **structure** of operating system.
5. With a neat sketch explain the structure of traditional **UNIX** operating system?
6. Describe the **services** that an operating-system provides to users?
7. List and explain different types of **system calls**?
8. What are the advantages and disadvantages of **layered approach**?
9. Write pros and cons of **micro kernels**?
10. Explain the need of **module structure** in operating system?