USN ::

CMR Institute of Technology, Bangalore
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
I – INTERNAL ASSESSMENT

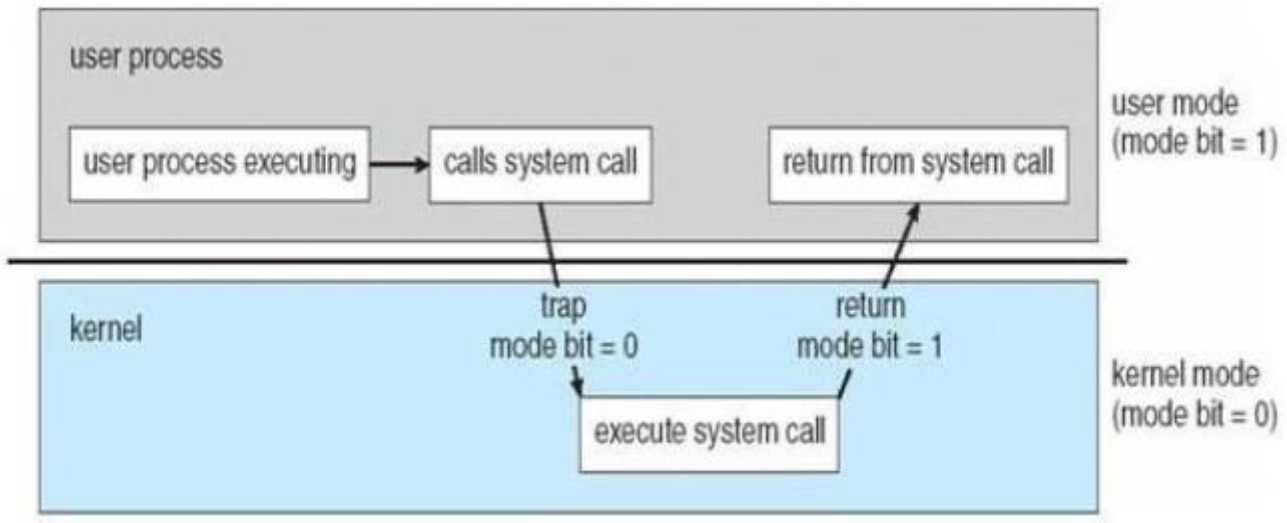
Semester: 4-CBCS 2018
Subject: OPERATING SYSTEMS (18CS43)
Faculty: Ms Savitha S

Date: 19 May 2021
Time: 01:00 PM - 02:00 PM
Max Marks: 50

Answer any 5 question(s)
SCHEME AND SOLUTION

Q.No			Marks	CO	BT/CL
1	a	What is an operating system? a) collection of programs that manages hardware resources b) system service provider to the applicationprograms c) interface between the hardware and application programs d) all of the mentioned Answer:d	1	CO1	L1
	b	In Unix, Which system call creates the new process? a) fork b) create c) new d) none of the mentioned Answer: a	1	CO1	L1
	c	Which of the following requires a device driver? a) Register b) Cache c) Main memory d) Disk Answer:d	1	CO1	L1
	d	To access the services of operating system, the interface is provided by the_____a) System calls b) API c) Library d) Assembly instructions Answer: a	1	CO1	L1
	e	Which of the following statements is false? a) Virtual memory implements the translation of a program's address space into physical memory address space. b) Virtual memory allows each program to exceed the size of the primary memory. c) Virtual memory increases the degree of multiprogramming. d) Virtual memory reduces the context switching overhead.d Answer: d	1	CO1	L2
	f	What is bootstrapping also called? a)Cold boot b)Cold hot boot c)Cold hot strap d) Hot boot Answer:a	1	CO1	L1
	g	Which of the following does not interrupt a running process? a) A device b) Timer c) Scheduler process d) Power failure Answer:c	1	CO1	L2
	h	A process executes the code. fork (); fork (); fork (); The total number of child processes created is a) 3 b) 4 c) 7 d) 8 Answer:c	2	CO1	L3
	i	What is interprocess communication? a) communication within the process b) communication between two process c) communication between two threads of same process d) none of the mentioned Answer:b	1	CO1	L1
2	a	The address of the next instruction to be executed by the current process is provided by the_____a) CPU registers b) Program counter c) Process stack d) Pipe Answer:b	1	CO1	L1
	b	What is a Process Control Block? a) Process type variable b) Data Structure c) A secondary storage section d) A Block in memory Answer:b	1	CO1	L1
	c	What is the degree of multiprogramming? a) the number of processes executed per unit time b) the number of processes in the ready queue c) the number of processes in the I/O queue d) the number of processes in memory Answer:d	1	CO1	L1
	d	What is the objective of multiprogramming? a) Have a process running at all time b) Have multiple programs waiting in a queue ready to run c) To increase CPU utilization d) None of the mentioned Answer:c	1	CO1	L2
	e	When the process issues an I/O request_____a) It is placed in an I/O queue b) It is placed in a waiting queue c) It is placed in the ready queue d) It is placed in the Job queue Answer:a	1	CO1	L2
	f	If all processes I/O bound, the ready queue will almost always be_____and the Short term Scheduler will have a_____to do. a) full, little b) full, lot c) empty, little d) empty, lot Answer:c	1	CO1	L2
	g	What is a medium-term scheduler? a) It selects which process has to be brought into the ready queue b) It selects which process has to be executed next and allocates CPU c) It selects which process to remove from memory by swapping d) None of the mentioned Answer:c	1	CO1	L1
	h	In a time-sharing operating system, when the time slot given to a process is completed, the process goes from the running state to the a) Blocked state b) Ready state c) Suspended state d) Terminated state Answer:b	1	CO1	L2

	i	Message passing system allows processes to_____a) communicate with each other without sharing the same address space b) communicate with one another by resorting to shared data c) share data d) name the recipient or sender of the message Answer:a	1	CO1	L2
	j	In the non blocking send_____a) the sending process keeps sending until the message is received b) the sending process sends the message and resumes operation c) the sending process keeps sending until it receives a message d) none of the mentioned Answer:b	1	CO1	L1
3		a) Explain the role of operating System from different view points. Operating System can be viewed from two viewpoints– User views & System views ***User Views:-T***he user's view of the operating system depends on the type of user. • If the user is using **standalone** system, then OS is designed for ease of use and high performances. Here resource utilization is not given importance. • If the users are at different **terminals** connected to a mainframe or minicomputers, by sharing information and resources, then the OS is designed to maximize resource utilization. OS is designed such that the CPU time, memory and i/o are used efficiently and no single user takes more than the resource allotted to them. • If the users are in **workstations**, connected to networks and servers, then the user have a system unit of their own and shares resources and files with other systems. Here the OS is designed for both ease of use and resource availability (files). • Other systems like embedded systems used in home devie (like washing m/c) & automobiles do not have any user interaction. There are some LEDs to show the status of its work • Users of **hand held** systems, expects the OS to be designed for ease of use and performance per amount of battery life ***System Views:-*** Operating system can be viewed as a **resource allocator** and **control program**. • **Resource allocator** - The OS acts as a manager of hardware and software resources. CPU time, memory space, file-storage space, I/O devices, shared files etc. are the different resources required during execution of a program. There can be conflicting request for these resources by different programs running in same system. The OS assigns the resources to the requesting program depending on the priority. • Control Program – The OS is a control program and manage the execution of user program to prevent errors and improper use of the computer. b) With a neat diagram explain dual mode operating in operatingsystem. **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 will sit quietly, waiting for something to happen. Events are signaled by the occurrence of an interrupt or a trap. A **trap (or** an **exception)** is a software-generated interrupt. For each type of interrupt, separate segments of code in the operating system determine what action should be taken. An interrupt service routine is provided that is responsible for dealing with the interrupt. ***Dual-Mode Operation*** Since the operating system and the user programs share the hardware and software resources of the computer system, it has to be made sure that an error in a user program cannot cause problems to other programs and the Operating System running in the system. The approach taken is to use a hardware support that allows us to differentiate among various modes of execution. The system can be assumed to work in two separate **modes** of operation: 1. User mode 2. Kernel mode (supervisor mode, system mode, or privileged mode). • A hardware bit of the computer, called the **mode bit**, is used to indicate the current mode: kernel (0) or user (1). With the mode bit, we are able to distinguish between a task that is executed by the operating system and one that is executed by the user. • When the computer system is executing a user application, the system is in user mode. When a user application requests a service from the operating system (via a system call), the transition from user to kernel mode takes place.	05 2.5 2.5 05	CO1	L2



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 (that is, changes the mode bit from 1 to 0). Thus, 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.

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. The instruction to switch to user mode is an example of a privileged instruction.

 Initial control is within the operating system, where instructions are executed in kernel mode. When control is given to a user application, the mode is set to user mode. Eventually, control is switched back to the operating system via an interrupt, a trap, or a system call.

4	Illustrate with a neat sketch and explain the process state transition diagram and the process control block.	05	CO1	L2

Process State

A Process has 5 states. Each process may be in one of the following states –

1. **New** - The process is in the stage of being created.
2. **Ready** - The process has all the resources it needs to run. It is waiting to be assigned to the processor.
3. **Running** – Instructions are being executed..
4. **Waiting -** The process is waiting for some event to occur. For example the process may be waiting for keyboard input, disk access request, inter-process messages, a timer to go off, or a child process to finish.
5. **Terminated -** The process has completed its execution.

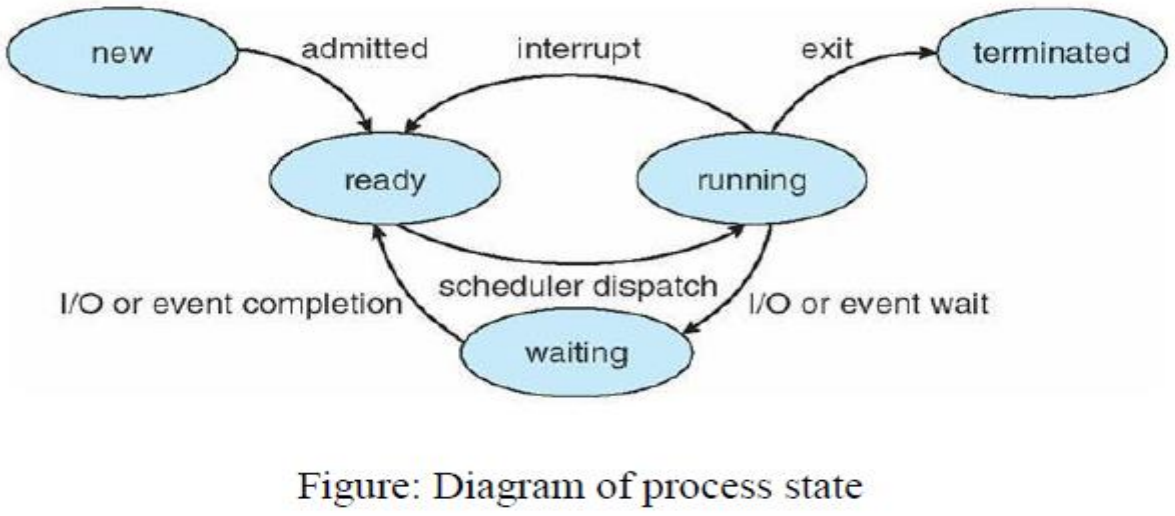


Figure: Diagram of process state

Process Control Block

For each process there is a Process Control Block (PCB), which stores the process-specific information as shown below –

- **Process State** – The state of the process may be new, ready, running, waiting, and so on.
- **Program counter –** The counter indicates the address of the next instruction to be executed for this process.
- **CPU registers -** The registers vary in number and type, depending on the computer architecture. They include accumulators, index registers, stack pointers, and general-purpose registers. Along with the program counter, this state information must be saved when an interrupt occurs, to allow the process to be continued correctly afterward.
- **CPU scheduling information-** This information includes a process priority, pointers to scheduling queues, and any other scheduling parameters.
- **Memory-management information –** This include information such as the value of the base and limit registers, the page tables, or the segment tables.
- **Accounting information** – This information includes the amount of CPU and real time used, time limits, account numbers, job or process numbers, and so on.
- **I/O status information –** This information includes the list of I/O devices allocated to the process, a list of open files, and so on. The PCB simply serves as the repository for any information that may vary from process to process.

Figure: Process control block (PCB)

5		Discuss the methods to implement message passing IPC in detail. A mechanism to allow process communication without sharing address space. It is used in distributed systems. • Message passing systems uses system calls for "send message" and "receive message". • A communication link must be established between the cooperating processes before messages can be sent. • There are three methods of creating the link between the sender and the receiver- • Direct or indirect communication (naming) • Synchronous or asynchronous communication (Synchronization) • Automatic or explicit buffering. **1. Naming** Processes that want to communicate must have a way to refer to each other. They can use either direct or indirect communication. **a) Direct communication** the sender and receiver must explicitly know each other's name. The syntax for send() and receive() functions are as follows- • **send** (*P, message*) – send a message to process P • **receive**(*Q, message*) – receive a message from process Q Properties of communication link : • A link is established automatically between every pair of processes that wants to communicate. The processes need to know only each other's identity to communicate. • A link is associated with exactly one pair of communicating processes • Between each pair, there exists exactly one link. Types of addressing in direct communication – • Symmetric addressing – the above described communication is symmetric communication. Here both the sender and the receiver processes have to name each other to communicate. • Asymmetric addressing – Here only the sender name is mentioned, but the receiving data can be from any system. **send(P,** message) --- Send a message to process ***P*** **receive(id,** message). Receive a message from any process Disadvantages of direct communication – any changes in the identifier of a process, may have to change the identifier in the whole system(sender and receiver), where the messages are sent and received. **b) Indirect communication** uses shared mailboxes, or ports. A mailbox or port is used to send and receive messages. Mailbox is an object into which messages can be sent and received. It has a unique ID. Using this identifier messages are sent and received. Two processes can communicate only if they have a shared mailbox. The send and receive functions are – • **send**(*A, message*) – send a message to mailbox A • **receive**(*A, message*) – receive a message from mailbox A Properties of communication link: • A link is established between a pair of processes only if they have a shared mailbox • A link may be associated with more than two processes • Between each pair of communicating processes, there may be any number of links, each link is associated with one mailbox. • A mail box can be owned by the operating system. It must take steps to – • create a new mailbox • send and receive messages from mailbox • delete mailboxes. **2. Synchronization** The send and receive messages can be implemented as either **blocking** or **non-blocking**. • **Blocking (synchronous) send -** sending process is blocked (waits) until the message is received by receiving process or the mailbox. • **Non-blocking (asynchronous) send** - sends the message and continues (doesnot wait) • **Blocking (synchronous) receive -** The receiving process is blocked until a message is available • **Non-blocking (asynchronous) receive** - receives the message without block. The received message may be a valid message or null.	10 2.5 2.5 5	CO1	L2

3. Buffering

When messages are passed, a temporary queue is created. Such queue can be of three capacities:
Zero capacity – The buffer size is zero (buffer does not exist). Messages are not stored in the queue. The senders must block until receivers accept the messages.
Bounded capacity- The queue is of fixed size(n). Senders must block if the queue is full. After sending 'n' bytes the sender is blocked.
Unbounded capacity **- The queue is of infinite capacity. The sender never blocks.**

6	a) With neat diagram explain the concept of Virtual Machines. The fundamental idea behind a virtual machine is to abstract the hardware of a single computer (the CPU, memory, disk drives, network interface cards, and so forth) into several different execution environments, thereby creating the illusion that each separate execution environment is running its own private computer. • Creates an illusion that a process has its own processor with its own memory. • Host OS is the main OS installed in system and the other OS installed in the system are called guest OS.	10 05	CO1	L2

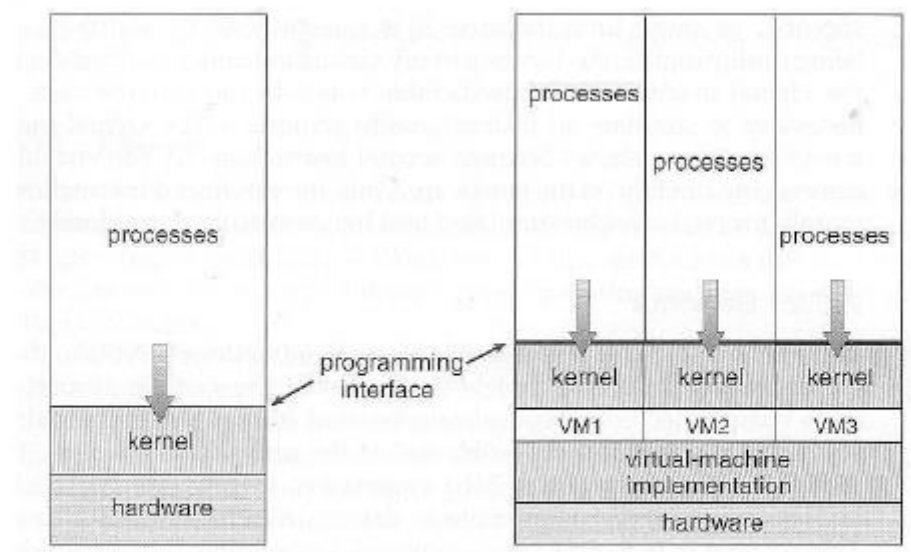


Figure: System modes. (A) Non-virtual machine (b) Virtual machine

Implementation

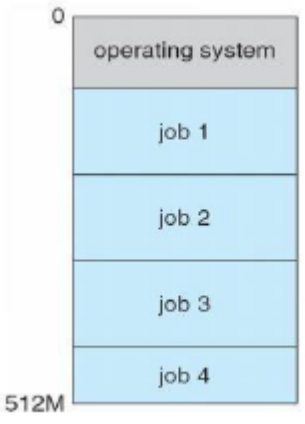
- The virtual-machine concept is useful, it is difficult to implement.
- Work is required to provide an exact duplicate of the underlying machine. Remember that the underlying machine has two modes: user mode and kernel mode.
- The virtual-machine software can run in kernel mode, since it is the operating system. The virtual machine itself can execute in only user mode.

Benefits

- Able to share the same hardware and run several different execution environments(OS).
- Host system is protected from the virtual machines and the virtual machines are protected from one another. A virus in guest OS, will corrupt that OS but will not affect the other guest systems and host systems.
- Even though the virtual machines are separated from one another, software resources can be shared among them. Two ways of sharing s/w resource for communication are:
- To share a file system volume (part of memory).
- To develop a virtual communication network to communicate between the virtual machines.
- The operating system runs on and controls the entire machine. Therefore, the current system must be stopped and taken out of use while changes are made and tested. This period is commonly called *system development time.* In virtual machines such problem is eliminated. User programs are executed in one virtual machine and system development is done in another environment

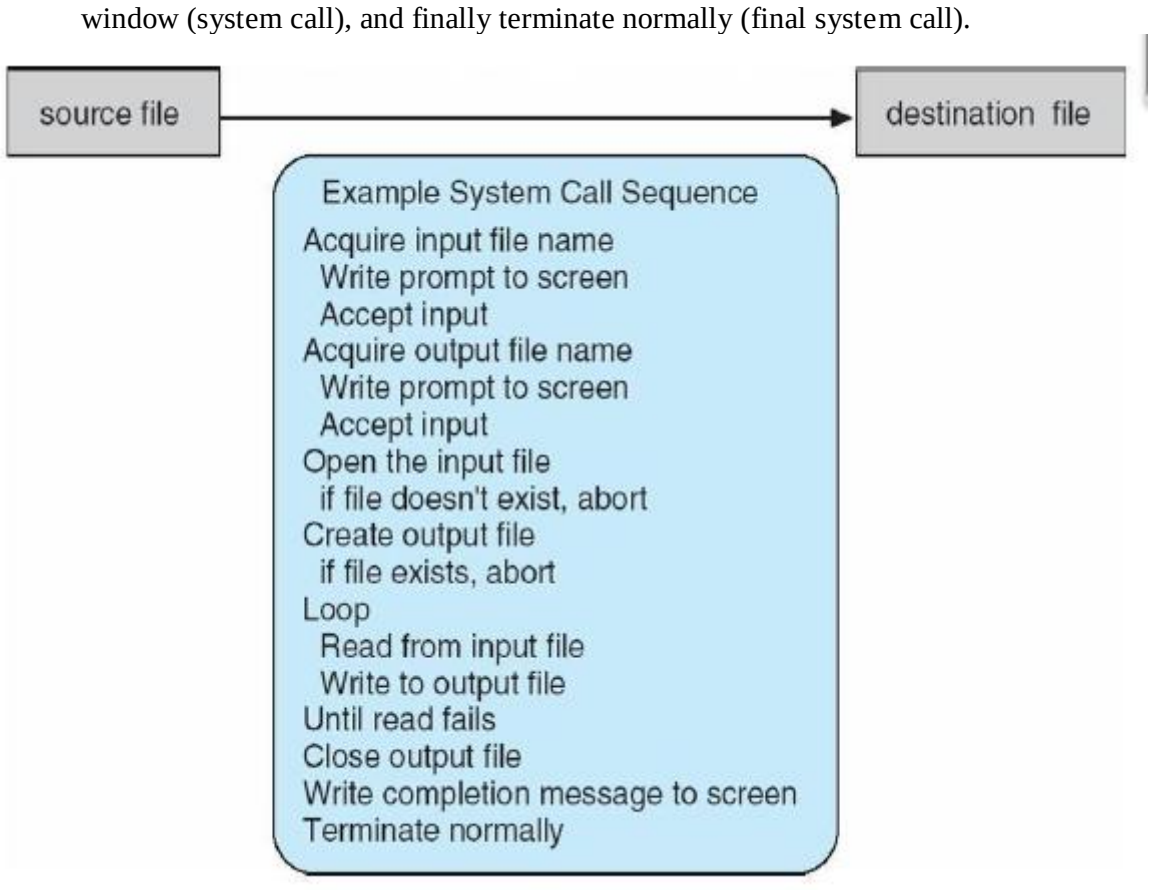
c) Compare multiprogramming and time sharing systems.

One of the most important aspects of operating systems is the ability to multiprogram. A single user cannot keep either the CPU or the I/O devices busy at all times. **Multiprogramming** increases CPU utilization by organizing jobs, so that the CPU always has one to execute.



The operating system keeps several jobs in memory simultaneously as shown in figure. This set of jobs is a subset of the jobs kept in the job pool. Since the number of jobs that can be kept simultaneously in memory is usually smaller than the number of jobs that can be kept in the job pool(in secondary memory). The operating system picks and begins to execute one of the jobs in memory. Eventually, the job may have to wait for some task, such as an I/O operation, to complete. In a non-multiprogrammed system, the CPU would sit idle.

- In a multiprogrammed system, the operating system simply switches to, and executes, another job. When *that* job needs to wait,

		the CPU is switched to *another* job, and so on. • Eventually, the first job finishes waiting and gets the CPU back. Thus the CPU is never idle. • Multiprogrammed systems provide an environment in which the various system resources (for example, CPU, memory, and peripheral devices) are utilized effectively, but they do not provide for user interaction with the computer system. In **Time sharing** (or **multitasking) systems,** a single CPU executes multiple jobs by switching among them, but the switches occur so frequently that the users can interact with each program while it is running. The user feels that all the programs are being executed at the same time. Time sharing requires an **interactive** (or **hands-on) computer system,** which provides direct communication between the user and the system. The user gives instructions to the operating system or to a program directly, using a input device such as a keyboard or a mouse, and waits for immediate results on an output device. Accordingly, the **response time** should be short— typically less than one second. A time-shared operating system allows many users to share the computer simultaneously. As the system switches rapidly from one user to the next, each user is given the impression that the entire computer system is dedicated to his use only, even though it is being shared among many users.			
7		What is a system call? With example explain the different categories of system calls. System calls is a means to access the services of the operating system generally written in C or C++, although some are written in assembly for optimal performance. • The below figure illustrates the sequence of system calls required to copy a file content from one file (input file) to another file (output file). • An example to illustrate how system calls are used: writing a simple program to read data from one file and copy them to another file • There are number of system calls used to finish this task. The first system call is to write a message on the screen (monitor). Then to accept the input filename. Then another system call to write message on the screen, then to accept the output filename. • When the program tries to open the input file, it may find that there is no file of that name or that the file is protected against access. In these cases, the program should print a message on the console (another system call) and then terminate abnormally (another system call) and create a new one (another system call). • Now that both the files are opened, we enter a loop that reads from the input file (another system call) and writes to output file (another system call). • Finally, after the entire file is copied, the program may close both files (another system call), write a message to the console or window (system call), and finally terminate normally (final system call).  • Most programmers do not use the low-level system calls directly, but instead use an "Application Programming Interface", API. • Instead of direct system calls provides for greater program portability between different systems. The API then makes the appropriate system calls through the system call interface, using a system call table to access specific numbered system calls. • Each system call has a specific numbered system call. The system call table (consisting of system call number and address of the particular service) invokes a particular service routine for a specific system call. • The caller need know nothing about how the system call is implemented or what it does during execution. • **Types of System Calls** The system calls can be categorized into six major categories: 1. Process Control 2. File management 3. Device management 4. Information management 5. Communications 6. Protection	10	CO1	L2

- Process control
  - end, abort
  - load, execute
  - create process, terminate process
  - get process attributes, set process attributes
  - wait for time
  - wait event, signal event
  - allocate and free memory
- File management
  - create file, delete file
  - open, close
  - read, write, reposition
  - get file attributes, set file attributes
- Device management
  - request device, release device
  - read, write, reposition
  - get device attributes, set device attributes
  - logically attach or detach devices
- Information maintenance
  - get time or date, set time or date
  - get system data, set system data
  - get process, file, or device attributes
  - set process, file, or device attributes
- Communications
  - create, delete communication connection
  - send, receive messages
  - transfer status information
  - attach or detach remote devices