Operating System (OS Services, Components, System Calls)

Deepika Rani Sahu

Asst. Professor

CSE Department

Contents

- Operating System Services
- Operating System Components
- System Calls

Operating System Services

- OS provides services to programs and to the users of those programs.
- Operating systems provide an environment for execution of programs. It makes programming tasks easier.
- Common services provided by operating system :
  - **User interface** - Almost all operating systems have a user interface (**UI**).
    - Varies between **Command-Line (CLI), Graphics User Interface (GUI), Batch Interface**

Operating System Services

- **Command-Line (CLI) :** Uses text commands and method for entering the programs.
- **Graphics User Interface (GUI)**
- **Batch Interface :** Commands and directives to control those commands are entered into files and those files are executed.

Operating System Services

- **Program execution** - The system must be able to load a program into memory and to run that program. The programs must be able to **end its 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** - Programs need to read and write files and directories, create and delete them, search them, list file information, permission management.

Operating System Services

- **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**.
- **Error detection** – OS needs to be constantly aware of possible errors
  - May occur in the CPU and memory hardware, in I/O devices and in user program

Operating System Services

- 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

– **Resource allocation** - When multiple users or multiple jobs running concurrently, resources must be allocated to each of them
  - Many types of resources - CPU cycles, main memory, file storage, I/O devices.

Operating System Services

- **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

Operating System Components

- Process Management
- Main Memory Management
- File Management
- I/O System Management
- Secondary Management
- Networking
- Protection System
- Command-Interpreter System

Process Management

- A **process** is a program in execution. A process needs certain resources, including CPU time, memory, files, and I/O devices, to accomplish its task.
- These resources are either given to the process when it is created or allocated to it while it is running. In addition to it, various initialization input may be passed along.

Program	Process
1. It is passive entity.	1. It is active entity.
2. A program is not itself a process.	2. A process is the unit of work in the system
3. It is the set of instructions and commands required to accomplish certain tasks	3. A program in execution is called process.

Process Management

- The operating system is responsible for the following activities in connection with process management.

  - Process creation and deletion.
  - process suspension and resumption.
  - Scheduling processes and threads on the CPU
  - Provision of mechanisms for:
    - process synchronization
    - process communication

Main-Memory Management

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

File Management

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

I/O System Management

- Purpose of OS is to hide the peculiarities of specific hardware devices such as I/O devices from the user.
- The I/O subsystem consists of several components:
  - A memory-management component that includes buffering, caching and spooling
  - A general device-driver software
  - Drivers for specific hardware devices

Secondary-Storage Management

- Since main memory (*primary storage*) is volatile and too small to accommodate all data and programs permanently, the computer system must provide *secondary storage* to back up main memory.

- Most modern computer systems use disks as the principle on-line storage medium, for both programs and data.

Secondary-Storage Management

- The operating system is responsible for the following activities in connection with disk management:
  - Free space management
  - Storage allocation
  - Disk scheduling

Networking (Distributed Systems)

- A *distributed* system is a collection of processors that do not share memory or a clock. Each processor has its own local memory.

- The processors in the system are connected through a communication network.

- Communication takes place using a *protocol*.

Networking (Distributed Systems)

- A distributed system provides user access to various system resources.

- Access to a shared resource allows:
  - Computation speed-up
  - Increased data availability
  - Enhanced reliability

Protection System

- *Protection* refers to a mechanism for controlling access by programs, processes, or users to both system and user resources.

- The protection mechanism must:
  - distinguish between authorized and unauthorized usage.
  - specify the controls to be imposed.
  - provide a means of enforcement.

Command-Interpreter System

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

Command-Interpreter System (Cont.)

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

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

SYSTEM CALL

Deepika Rani Sahu

Bootstrap Loader

- The bootstrap program must locate the operating-system kernel and load it into memory.
- Once the kernel is loaded and executing, it can start providing services to the system and its users.
- Some services are provided outside of the kernel, by system programs that are loaded into memory at boot time to become **system processes that run the entire time the kernel is running.**
- On UNIX, the first system process is “init,” and it starts many other system programs.
- Once this phase is complete**, the system is fully booted,** and the **system waits for some event to occur.**

Interrupt

- The occurrence of an event is signaled by an interrupt(hardware and software)
- **Hardware interrupt:** Hardware may trigger an interrupt at any time by sending a signal to the CPU, usually by way of the system bus.
- **Software interrupt (exception or trap):**
  - Software may trigger an interrupt by executing a special operation called a **system call**
  - Software error (e.g., division by zero)
  - Request for operating system service
  - Other process problems include infinite loop, processes modifying each other or the operating system

Interrupt

- When the CPU is interrupted, it stops what it is doing and immediately transfers execution to a fixed location.
- The fixed location usually contains the starting address where the service routine for the interrupt is located.
- The interrupt service routine executes; on completion, the CPU resumes the interrupted computation.

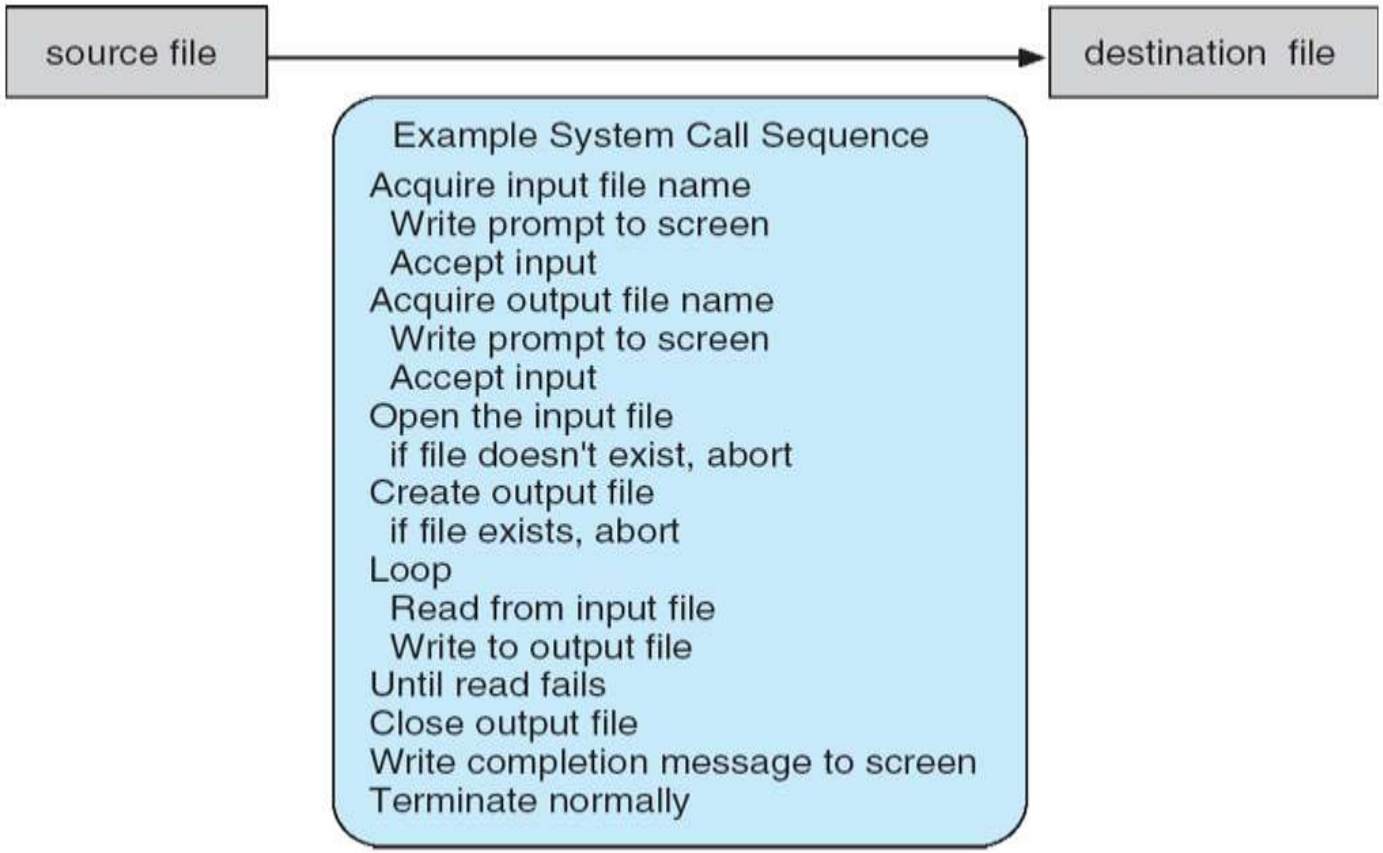
System Calls

- System calls provide the interface between a running program and the operating system.

- Three general methods are used to pass parameters between a running program and the operating system.
  - Pass parameters in *registers*.
  - Store the parameters in a table in memory, and the table address is passed as a parameter in a register.
  - *Push* (store) the parameters onto the *stack* by the program, and *pop* off the stack by operating system.

System Calls

- System call sequence to copy the contents of one file to another file



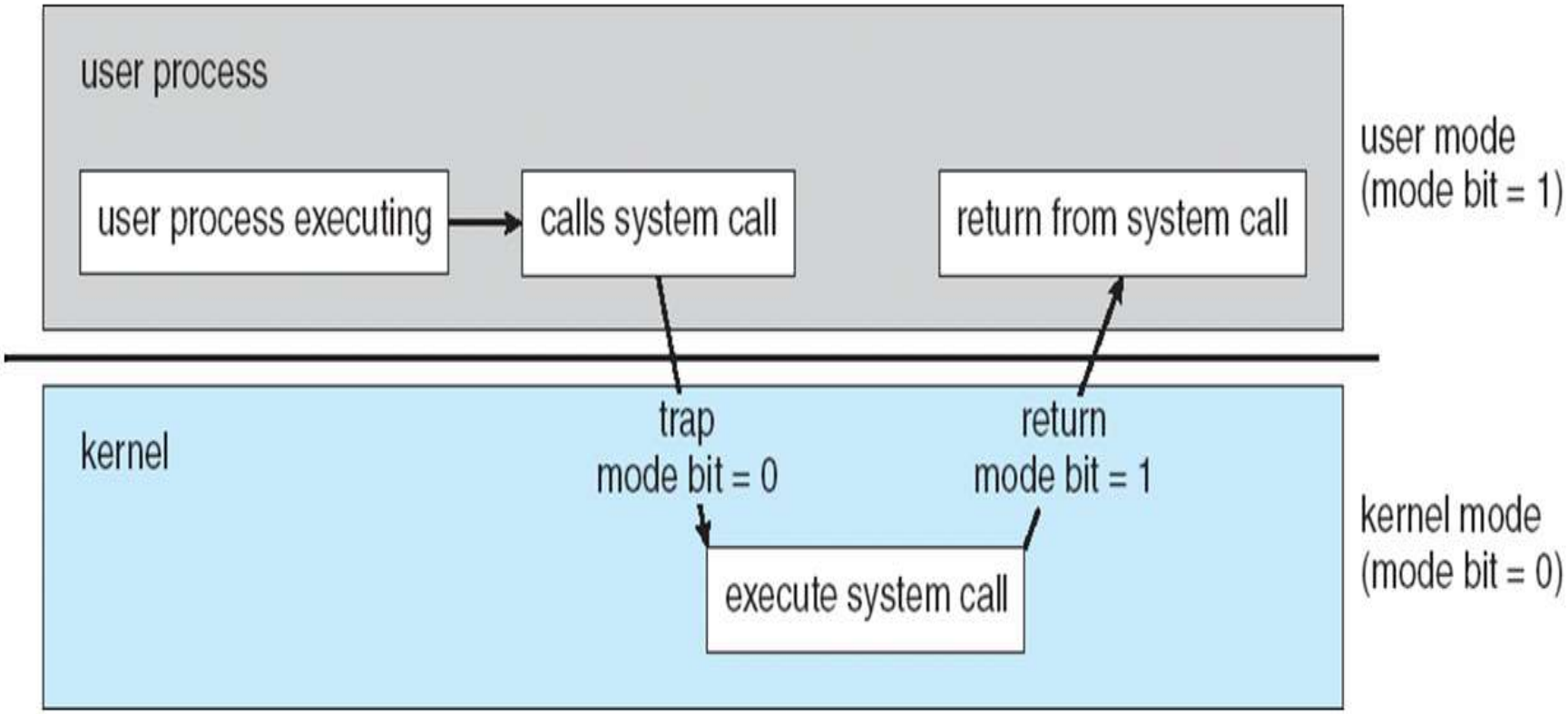
Dual Mode Operation

- Dual-mode operation allows OS to protect itself and other system components
- 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).**
- With the mode bit, we can distinguish between a task that is executed on behalf of the operating system and one that is executed on behalf of the user.
- When the computer system is executing on behalf of a user application, the system is in **user mode.**

Dual Mode Operation

- When a user application requests a service from the operating system (via a system call), the system must do transition from user to kernel mode to fulfill the request.
- 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

Cont..


user process
user process executing
calls system call
return from system call
user mode
(mode bit = 1)
kernel
trap
mode bit = 0
return
mode bit = 1
execute system call
kernel mode
(mode bit = 0)

System Call Mechanism

- 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 state of the mode bit to 0).
- Thus, whenever the operating system gains control of the computer, it is in kernel mode.
- The system always switches to user mode (by setting the mode bit to 1) before passing control to a user program.

Cont..

- Timer to prevent infinite loop / process hogging resources
  - Timer is set to interrupt the computer after some time period
  - Operating system set the counter (privileged instruction)
  - Keep a counter that is decremented by the physical clock
  - When counter zero generate an interrupt
  - Set up before scheduling process to regain control or terminate program that exceeds allotted time

Types of System Calls

- Process control
  - create process, terminate process
  - end, abort
  - load, execute
  - get process attributes, set process attributes
  - wait for time
  - wait event, signal event
  - allocate and free memory
  - Dump memory if error
  - **Debugger** for determining **bugs, single step** execution
  - **Locks** for managing access to shared data between processes

Types of System Calls

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

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

Types of System Calls (Cont.)

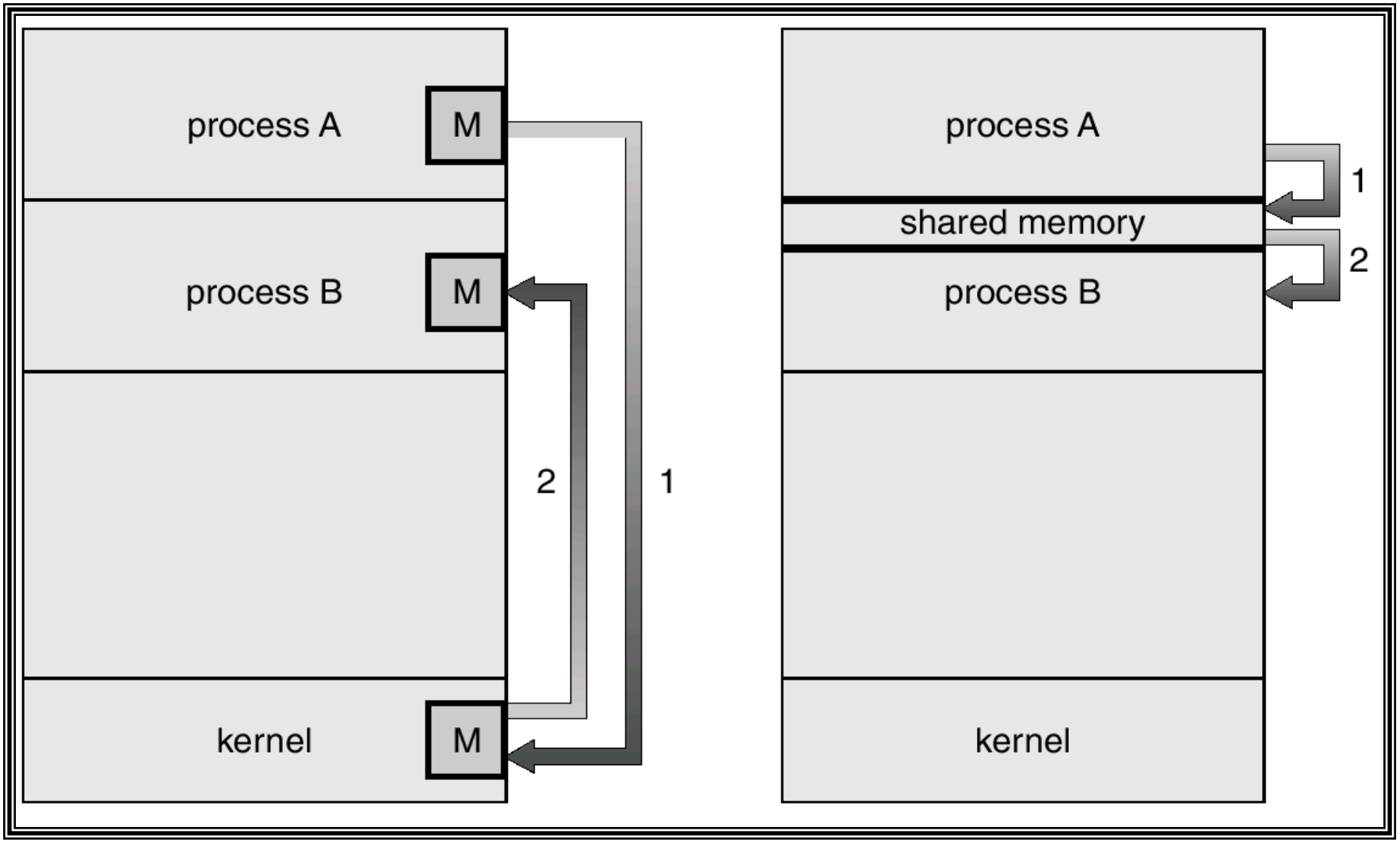
- Information maintenance
  - get time or date, set time or date
  - get system data, set system data
  - get and set process, file, or device attributes
- Communications
  - create, delete communication connection
  - send, receive messages if **message passing model** to **host name** or **process name**
    - From **client** to **server**

Types of System Calls (Cont.)

- **Shared-memory model** create and gain access to memory regions
- transfer status information
- attach and detach remote devices

Message Passing

Shared Memory



Types of System Calls (Cont.)

- Protection
  - Control access to resources
  - Get and set permissions
  - Allow and deny user access

Examples of Windows and Unix System Calls

	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()


ANY QUERIES???