



# OPERATING SYSTEMS STRUCTURES



# OPERATING SYSTEM SERVICES

# USER INTERFACE

- Command line interface(CLI):uses text commands and a method for entering them
- Batch interface(BI):commands and directives to control those commands are entered into files and those files are executed
- Graphical user interface(GUI):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 run that program
- The program must be able to end its execution, either normally or abnormally

# I/O operations

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

# FILE SYSTEM MANIPULATION

- Programs need to be able to read and write files and directories
- Create and delete files by name, search for a given file and list file information
- Some programs include permissions management to allow or deny access to files or directories based on file ownership

# COMMUNICATIONS

- Process needs to exchange information with other process
- Processes executing on same computer system or on different computer systems
- Communication may be implemented via shared memory or through message passing
- Message passing: packets of information are moved between processes

# ERROR DETECTION

- Errors may occur in CPU, memory hardware, I/O devices and in the user program
- For each type of error, the 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 use the system efficiently

# RESOURCE ALLOCATION

- When there are multiple jobs running at the same time, resources must be allocated to each of them
- Resources: CPU cycles, main memory, file storage and I/O devices
- CPU scheduling routines: to determine how best to use the CPU
- Routines to allocate printers, modems, USB storage drives and other peripheral devices

# ACCOUNTING

- Keeping a track of which users are using how much and what kinds of computer resources can be used for accounting or simply for accumulating usage statistics
- Usage statistics → reconfigure the system to improve computing services

# PROTECTION AND SECURITY

- Protection involves ensuring that all access to system resources is controlled
- To make a system secure, the user needs to authenticate himself or herself to the system

# User operating system interface

- Command line interface or command interpreter – allows users to directly enter commands that are to be performed by the OS
- Graphical user interface (GUI)

# Command interpreter

- Some operating systems include command interpreter in the kernel
- Windows XP and UNIX treat the command interpreter as a special program that is running when the job is initiated
- Interpreters → shells : on systems with multiple command interpreters
- Ex: On UNIX and Linux systems : Bourne shell, C shell, Bourne-Again shell, Korn shell, etc

- Main function is to get and execute the user- specified commands like create, delete, list, print, copy, execute , and so on
- Two ways to implement the commands:
  - 1) command interpreter itself contains the code to execute the command
  - 2) implementing most commands through system programs (ex: rm file.txt)

# Graphical user interface

- Provides a mouse-based-window-and-menu-system as an interface
- Provide a desktop metaphor where the mouse is moved to position its pointer on images, or icons, on the screen that represent programs, files, directories and system functions

# SYSTEM CALLS

- provide an interface to the services made
- available by an OS available as routines written in C and C++, assembly language

# API


- Application developers design programs according to an application programming interface (API)
- Specifies a set of functions that are available to an application programmer, including the parameters that are passed to each function and the return values the programmer can expect.

# Three of the most common APIs

- Win32 API for windows system
- POSIX API for POSIX-based systems( all versions of UNIX, Linux, and Mac OS X )
- Java API for designing programs that run on Java virtual machine

# Why would an application programmer prefer programming according to an API rather than invoking actual system calls?

- program portability: an application programmer designing a program using an API can expect her program to compile and run on any system that supports the same API
- actual system calls often can be more detailed and difficult to work with than the API available to an application programmer

- 
- The run time support system( a set of functions built into libraries included with a compiler) for most programming languages provides a system-call interface that serves as the link to system calls made available by the OS
  - intercepts function calls in the API and invokes the necessary system call within the OS a number is associated with each system call
  - the system call interface maintains a table indexed according to these numbers then invokes the intended system call in the OS kernel and returns the status of the system call and any return values

# Three methods to pass parameters

- Pass parameters in registers
- Parameters are stored in a block, or table, in memory, and the address of the block is passed as parameter in a register
- Parameters can be placed or pushed onto the stack by the program and popped off the stack by the OS

# Types of system calls

- Process control
- File manipulation
- Device manipulation
- Information maintenance
- communications

# 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 date
- 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

# (get process attributes and set process attributes)

- to determine and reset the attributes of a job or process, including the job's priority, its maximum allowable execution time and so on
- Usage : when a new process is created
- In order to control its execution

# (create process or submit job)

- To create a new program that can be multiprogrammed and save the memory image of existing program

# (terminate process)

- To terminate a process when it is incorrect or no longer needed

# (wait time)

- To wait for a certain amount of time
- Usage – when we need to wait for already created processes to finish their execution

# (wait event)

- To wait for a specific event to occur

# (signal event)

- Used by jobs or processes to signal that the event has occurred