Introduction to System Programming

Computer System Overview, Operating System Overview.

Minal Shah

Outline

- Operating System
  - Introduction
  - Objectives
  - Functions
  - Evolution
  - Major Achievements
  - Characteristics of Modern Operating System

Operating System

- A program that controls the execution of application programs
- An interface between applications and hardware
- Main objectives of an OS:
  - **Convenience** : An OS makes a computer more convenient to use
  - **Efficiency** : An OS allows the computer system resources to be used in an efficient manner.
  - **Ability to evolve** : An OS should be constructed in such a way as to permit the effective development, testing, and introduction of new system functions without interfering with service.

Operating System Services

- **Program Development: Editors and debugger**
  - The OS provides a variety of facilities and services, such as editors and debuggers, to assist the programmer in creating programs.
  - Typically, these services are in the form of utility programs that, while not strictly part of the core of the OS, are supplied with the OS and are referred to as application program development tools.

Operating System Services

- **Program execution:**
  - A number of steps need to be performed to execute a program.
  - Instructions and data must be loaded into main memory, I/O devices and files must be initialized, and other resources must be prepared.
  - The OS handles these scheduling duties for the user.
- **Access to I/O devices**:
  - Each I/O device requires its own peculiar set of instructions or control signals for operation.
  - The OS provides a uniform interface that hides these details so that programmers can access such devices using simple reads and writes.

Operating System Services

- **Controlled access to files**
  - Accessing different media but presenting a common interface to users
  - Provides protection in multi-access systems
- **System access** :
  - Controls access to the system and its resources for shared or public systems.
  - The access function must provide protection of resources and data from unauthorized users

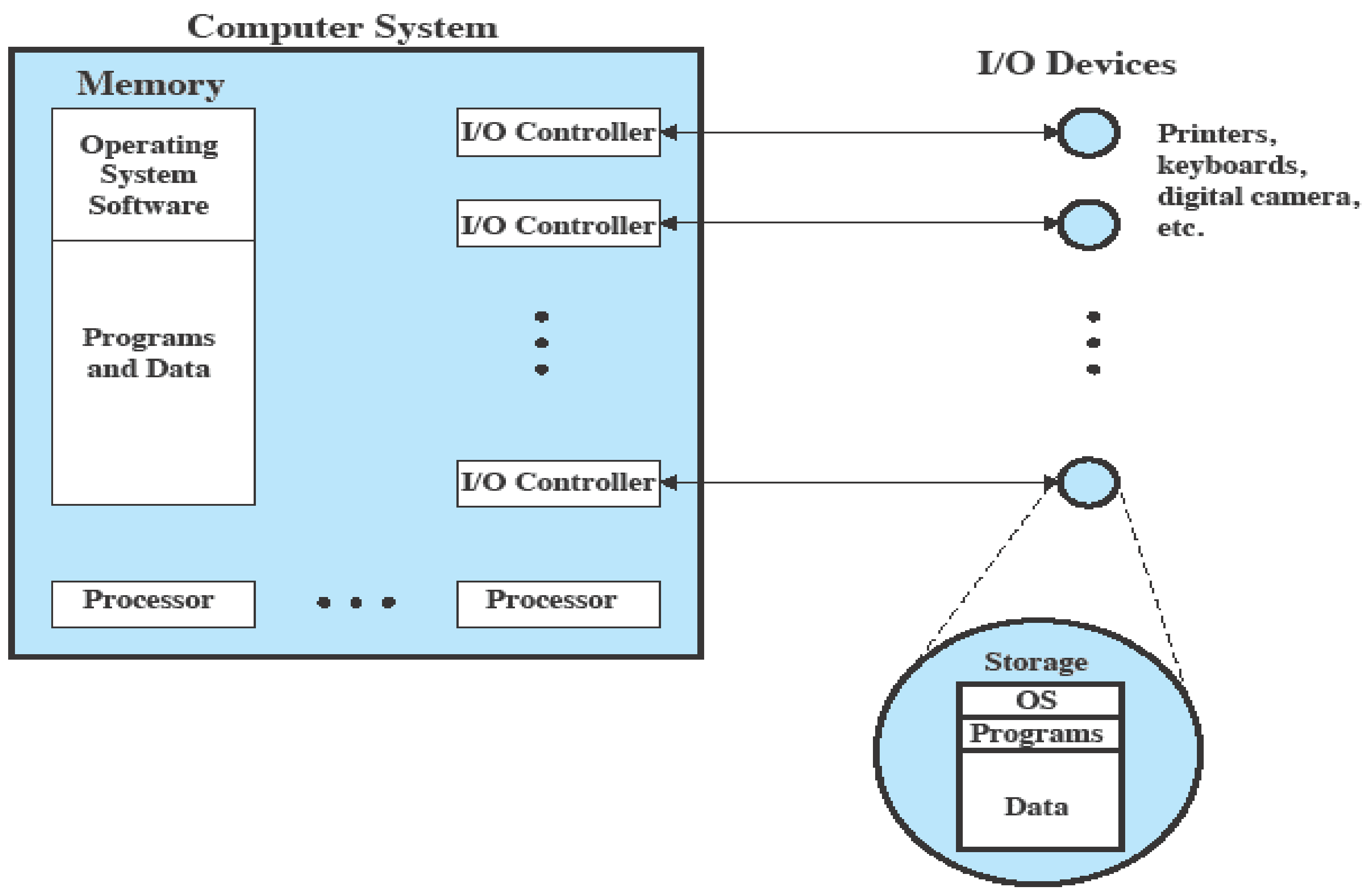
Operating System Services

- **Error detection and response**
  - Internal and external hardware errors
  - Software errors
  - Operating system cannot grant request of application
- **Accounting**
  - Collect usage statistics
  - Monitor performance parameters such as response time.

Operating System’s Role

- A computer is a set of resources for the movement, storage, and processing of data.
- The OS is responsible for managing these resources.
  - **As Software** : The OS functions in the same way as an ordinary computer software
  - It is a program or suite of programs executed by the processor.

OS As Resource Manager



Operating System’s Functions

- Process Management
- Memory Management
- Secondary Storage Management
- I/O Management
- File Management
- Protection
- Networking Management
- Command Interpretation.

Operating System’s Functions

- **Process Management** [This term was first used by the designers of Multics in the 1960s.]
  - A *process* is: A program in execution. An instance of a running program
  - The creation and deletion of both user and system processes
  - The suspension and resumption of processes.
  - The provision of mechanisms for process synchronization
  - The provision of mechanisms for deadlock handling.

Unix Process Inherited Properties

- user and group ids
- process group id
- controlling terminal
- setuid flag
- current working directory
- root directory (chroot)
- file creation mask
- signal masks
- close-on-exec flag
- Environment
- shared memory
- resource limits

Causes of Errors when Designing System Software

- Error in designing an OS are often subtle and difficult to diagnose
- Errors typically include:
  - Improper synchronization
  - Failed mutual exclusion [Often more than one user or program will attempt to make use of a shared resource at the same time.]
  - Non-determinate program operation [But when programs share memory, and their execution is interleaved by the processor, they may interfere with each other by overwriting common memory areas in unpredictable ways.]
  - Deadlocks [It is possible for two or more programs to be hung up waiting for each other]

Components of a Process

- A process consists of
  - An executable program
  - Associated data needed by the program (variables, work space, buffers, etc.)
  - Execution context of the program (or “process state”)
- The execution context contains all information the operating system needs to manage the process

Operating System’s Functions

- **Memory Management**
  - Keep track of which parts of memory are currently being used and by whom.
  - Decide which processes are to be loaded into memory when memory space becomes available.
  - Allocate and de-allocate memory space as needed.
- The OS has 5 principal storage management responsibilities
  - Process isolation
  - Automatic allocation and management
  - Support of modular programming
  - Protection and access control
  - Long-term storage

Operating System’s Functions

- **Process isolation** : The OS must prevent independent processes from interfering with each other’s memory, both data and instructions.
- **Automatic allocation and management** : Programs should be dynamically allocated across the memory hierarchy as required. Allocation should be transparent to the programmer.
- **Support of modular programming** : Programmers should be able to define program modules, and to create, destroy, and alter the size of modules dynamically.
- **Protection and access control** :Sharing of memory, at any level of the memory hierarchy, creates the potential for one program to address the memory space of another. The OS must allow portions of memory to be accessible in various ways by various users.

Operating System’s Functions

- **Long-term storage :** Virtual memory
- File system implements long-term store
- Virtual memory allows programs to address memory from a logical point of view
  - Without regard to the limits of physical memory
- **Paging** : Allows process to be comprised of a number of fixed-size blocks, called pages
  - Virtual address is a page number and an offset within the page
  - Each page may be located any where in main memory

Operating System’s Functions

- **Secondary Management**
  - Free space management
  - Storage allocation
  - Disk scheduling..

Operating System’s Functions

- **I/O Management**
  - A buffer caching system
  - To activate a general device driver code
  - To run the driver software for specific hardware devices as and when required..

Operating System’s Functions

- **File Management**
  - The creation and deletion of files.
  - The creation and deletion of directory.
  - The support of primitives for manipulating files and directories.
  - The mapping of files onto disk storage.
  - Backup of files on stable (non volatile) storage.
  - Protection and security of the files.

Operating System’s Functions

- **Protection**
  - The various processes in an operating system must be protected from each other’s activities.
  - Protection refers to a mechanism for controlling the access of programs, processes, or users to the resources defined by a computer controls to be imposed, together with some means of enforcement.
  - Main issues are: Availability, Confidentiality, Data integrity, Authenticity

Operating System’s Functions

- **Networking**
  - A distributed system is a collection of processors that do not share memory or a clock. Instead, each processor has its own local memory, and the processors communicate with each other through various communication lines, such as high speed buses or telephone lines.
  - Distributed systems vary in size and function.
  - They may involve microprocessors, workstations, minicomputers, and large general purpose computer systems.
  - Access to a shared resource allows computation speed-up, data availability, and reliability.

Operating System’s Functions

- **Command Interpretation**
  - The command interpreter is the primary interface between the user and the rest of the system.
  - The command statements themselves deal with process management, I/O handling, secondary storage management, main memory management, file system access, protection, and networking

Operating System’s Functions

- **Scheduling and Resource Management**
  - Resource allocation policies must consider: Fairness, Differential responsiveness, Efficiency
  - Fairness: All processes that are competing for the use of a particular resource to be given approximately equal and fair access to that resource. This is especially so for jobs of the same class, that is, jobs of similar demands.
  - Differential responsiveness: The OS should attempt to make allocation and scheduling decisions to meet the total set of requirements.
  - Efficiency: The OS should attempt to maximize throughput, minimize response time, and accommodate as many users as possible.

DESIRABLE QUALITIES OF OS

- The desirable qualities of an operating system are in terms of: **Usability, Facilities, Cost, and Adaptability**.
- **Usability**: Robustness, *Accept all valid inputs and can handle them,* Consistency, Proportionality, Convenience, Powerful with high level facilities.
- **Facilities**: Sufficient for intended use, Complete, Appropriate.
- **Costs**: Want low cost and efficient services, Good algorithms, *Make use of space/time tradeoffs, special hardware,* Low overhead, *Cost of doing nothing should be low. E.g., idle time at a terminal,* Low maintenance cost, *System should not require constant attention.*

DESIRABLE QUALITIES OF OS

- **Adaptability**:
  - Tailored to the environment. [*Support necessary activities. Do not impose unnecessary restrictions.]*
  - *What are the things people do most -- make them easy.[* Changeable over time.]
  - *Adapt as needs and resources change. e.g., expanding memory and new devices, or new user population.*
  - Extendible-Extensible [*Adding new facilities and features - which look like the old ones.]*

Evolution of Operating System

- Operating systems will evolve over time
  - Hardware upgrades plus new types of hardware : early versions of UNIX and the Macintosh operating system did not employ a paging mechanism because they were run on processors without paging hardware.
  - Subsequent versions of these operating systems were modified to exploit paging capabilities.
  - Also, the use of graphics terminals and page-mode terminals instead of line-at-a-time scroll mode terminals affects OS design.
  - For example, a graphics terminal typically allows the user to view several applications at the same time through “windows” on the screen.

Evolution of Operating System

- **New services** :
  - In response to user demand or in response to the needs of system managers, the OS expands to offer new services.
  - e.g. if it is found to be difficult to maintain good performance for users with existing tools, new measurement and control tools may be added to the OS.
- **Fixes** :
  - Any OS has faults. These are discovered over the course of time and fixes are made.
  - Of course, the fix may introduce new faults.

Types of Operating System

- Modern computer operating systems may be classified into three groups, which are distinguished by the nature of interaction that takes place between the computer user and his or her program during its processing.
- The three groups are called **batch, timesharing and real-time operating systems.**

Types of Operating System

- **Batch Processing Operating System**
  - In a batch processing operating system environment users submit jobs to a central place where these jobs are collected into a batch, and subsequently placed on an input queue at the computer where they will be run.
  - In this case, the user has no interaction with the job during its processing, and the computer's response time is the turnaround time "the time from submission of the job until execution is complete, and the results are ready for return to the person who submitted the job".

Types of Operating System

- **Time Sharing**
  - Another mode for delivering computing services is provided by time sharing operating systems.
  - In this environment a computer provides computing services to several or many users concurrently on-line.
  - Many users are sharing the central processor, the memory, and other resources of the computer system in a manner facilitated, controlled, and monitored by the operating system.
  - The user, in this environment, has nearly full interaction with the program during its execution, and the computer's response time may be expected to be no more than a few second.

Types of Operating System

- **Real Time Operating System (RTOS) :**
  - The third class is the real time operating systems, which are designed to service those applications where response time is of the essence in order to prevent error, misrepresentation or even disaster.
  - Examples of real time operating systems are those which handle airlines reservations, machine tool control, and monitoring of a nuclear power station.
  - The systems, in this case, are designed to be interrupted by external signals that require the immediate attention of the computer system.
  - An RTOS typically has very little user-interface capability, and no end-user utilities.

Types of Operating System

- **Multiprogramming Operating System** :
  - A multiprogramming operating system is a system that allows more than one active user program (or part of user program) to be stored in main memory simultaneously.
  - Thus, it is evident that a time-sharing system is a multiprogramming system, but note that a multiprogramming system is not necessarily a time-sharing system.
  - A batch or real time operating system could, and indeed usually does, have more than one active user program simultaneously in main storage.

Types of Operating System

- **Multiprocessing System :**
  - A multiprocessing system is a computer hardware configuration that includes more than one independent processing unit.
  - It is generally used to refer to large computer hardware complexes found in major scientific or commercial applications.

Types of Operating System

- **Networking Operating System**:
  - A networked computing system is a collection of physical interconnected computers.
  - The operating system of each of the interconnected computers must contain, in addition to its own stand-alone functionality, provisions for handing communication and transfer of program and data among the other computers with which it is connected.
  - Network operating systems need a network interface controller and some low-level software to drive it, as well as programs to achieve remote login and remote files access, but these additions do not change the essential structure of the operating systems.

Types of Operating System

- **Distributed Operating System**:
  - A distributed computing system consists of a number of computers that are connected and managed so that they automatically share the job processing load among the constituent computers, or separate the job load as appropriate particularly configured processors.
  - Such a system requires an operating system which, in addition to the typical stand-alone functionality, provides coordination of the operations and information flow among the component computers.

Types of Operating System

- **Distributed Operating System**:
  - The networked and distributed computing environments and their respective operating systems are designed with more complex functional capabilities.
  - In a network operating system, the users are aware of the existence of multiple computers, and can log in to remote machines and copy files from one machine to another. Each machine runs its own local operating system and has its own user (or users).

Types of Operating System

- **Distributed Operating System:**
  - A distributed operating system, in contrast, is one that appears to its users as a traditional uni-processor system, even though it is actually composed of multiple processors.
  - In a true distributed system, users should not be aware of where their programs are being run or where their files are located; that should all be handled automatically and efficiently by the operating system.

Types of Operating System

- **Operating Systems for Embedded Devices** :
  - As embedded systems (PDAs, cellphones, point-of-sale devices, VCR’s, industrial robot control, or even your toaster) become more complex hardware-wise with every generation
  - As more features are put into them day-by-day, applications they run require more and more to run on actual operating system code in order to keep the development time reasonable.

Evolution of Operating System

- **Stages include**
  - Serial Processing
  - Simple Batch Systems
  - Multiprogrammed batch systems
  - Time Sharing Systems

Evolution of Operating System [Serial Processing]

- late 1940s to the mid-1950s
- No operating system
- Machines run from a console with display lights, toggle switches, input device, and printer
- Programs in machine code were loaded via the input device (e.g., a card reader).
    - If an error halted the program, the error condition was indicated by the lights.
    - If the program proceeded to a normal completion, the output appeared on the printer.

Evolution of Operating System [Serial Processing]

- **Problems include**:
  - **Scheduling** :Most installations used a hardcopy sign-up sheet to reserve computer time. Typically, for a block of time in multiples of a half hour or so, which could lead to waste of time if finish faster.
  - On the other hand, the user might run into problems, not finish in the allotted time, and be forced to stop before resolving the problem.

Evolution of Operating System [Serial Processing]

- **Problems include**:
  - Setup time: A single program, called a job, could involve loading the compiler plus the high-level language program (source program) into memory, saving the compiled program (object program) and then loading and linking together the object program and common functions.
  - Each of these steps could involve mounting or dismounting tapes or setting up card decks.
  - If an error occurred, the hapless user typically had to go back to the beginning of the setup sequence.
  - Thus, a considerable amount of time was spent just in setting up the program to run

Evolution of Operating System [Simple batch system]

- Early computers were extremely expensive
  - Important to maximize processor utilization
  - The wasted time due to scheduling and setup time was unacceptable.
  - To improve utilization, the concept of a batch operating system was developed.
- Monitor : The central idea behind the simple batch-processing scheme is the use of a piece of software known as the **monitor.**
  - Software that controls the sequence of events
  - Batch jobs together
  - Program returns control to monitor when finished

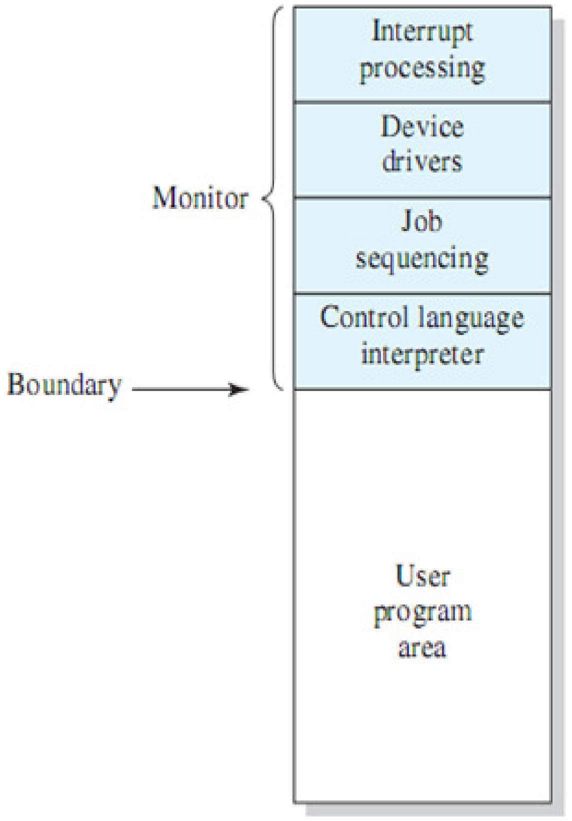
Evolution of Operating System [Simple batch system]

- With this type of OS, **the user no longer has direct access to the processor**.
- Instead, the user submits the job on cards or tape to a computer operator, who batches the jobs together sequentially and places the entire batch on an input device, for use by the monitor.
- Each program is constructed to branch back to the monitor when it completes processing, at which point the monitor automatically begins loading the next program.

Evolution of Operating System [Simple batch system]

Monitor’s perspective :

- Monitor controls the sequence of events
- *Resident Monitor* is software always in memory
- The rest of the monitor consists of utilities and common functions that are loaded as subroutines to the user program at the beginning of any job that requires them
- The monitor reads in jobs one at a time from the input device.



Memory Layout for a Resident Monitor

Evolution of Operating System [Simple batch system Desirable H/W]

- Memory protection for monitor
  - Jobs cannot overwrite or alter.
  - If such an attempt is made, the processor hardware should detect an error and transfer control to the monitor.
  - The monitor would then abort the job, print out an error message, and load in the next job.
- Timer
  - Prevent a job from monopolizing system
  - The timer is set at the beginning of each job.
  - If the timer expires, the user program is stopped, and control returns to the monitor.

Evolution of Operating System [Simple batch system Desirable H/W]

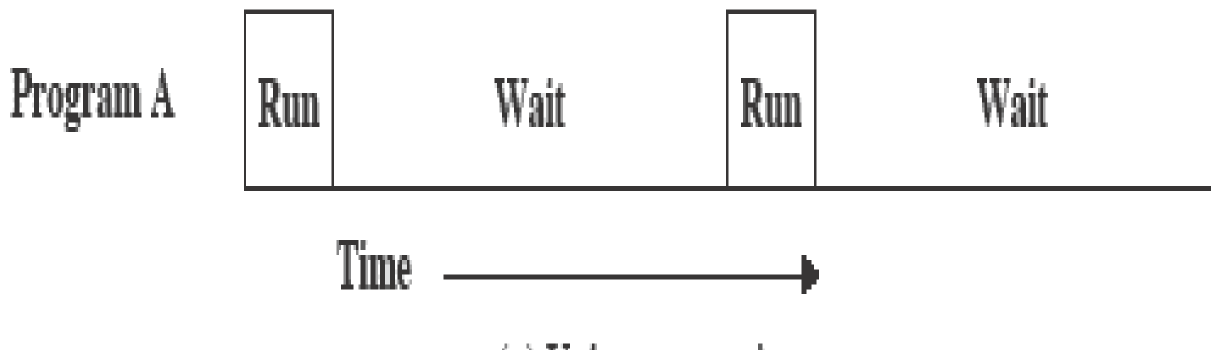
- Privileged instructions
  - Only executed by the monitor (Certain machine level instructions)
  - If the processor encounters such an instruction while executing a user program, an error occurs causing control to be transferred to the monitor.
  - Among the privileged instructions are I/O instructions, so that the monitor retains control of all I/O devices.
  - This prevents a user program from accidentally reading job control instructions from the next job.
  - If a user program wishes to perform I/O, it must request that the monitor perform the operation for it.

Evolution of Operating System [Simple batch system Desirable H/W]

- Interrupts
  - Early computer models did not have this capability.
  - This feature gives the OS more flexibility in controlling of user programs
- Modes of Operation (Necessary for memory protection)
  - User Mode :
    - User program executes in user mode
    - Certain areas of memory protected from user access
    - Certain instructions may not be executed
  - Kernel Mode
    - Monitor executes in kernel mode
    - Privileged instructions may be executed, all memory accessible.

Evolution of Operating System [Uniprogramming batch system]

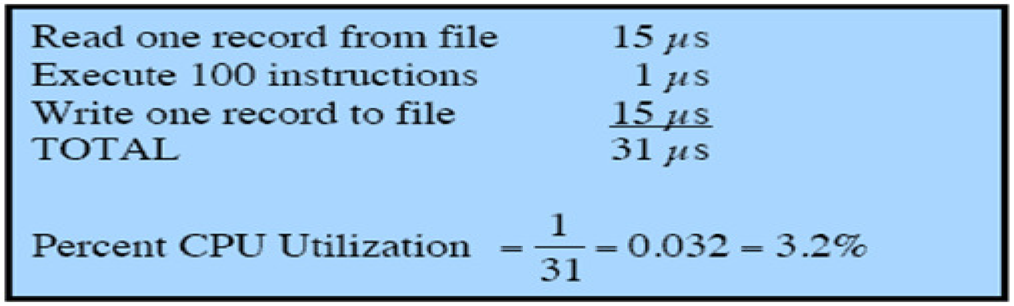
- Processor must wait for I/O instruction to complete before preceding



(a) Uniprogramming

Evolution of Operating System [Multiprogrammed batch system]

- CPU is often idle
  - Even with automatic job sequencing.
  - I/O devices are slow compared to processor



Read one record from file	15 μs
Execute 100 instructions	1 μs
Write one record to file	15 μs
TOTAL	31 μs

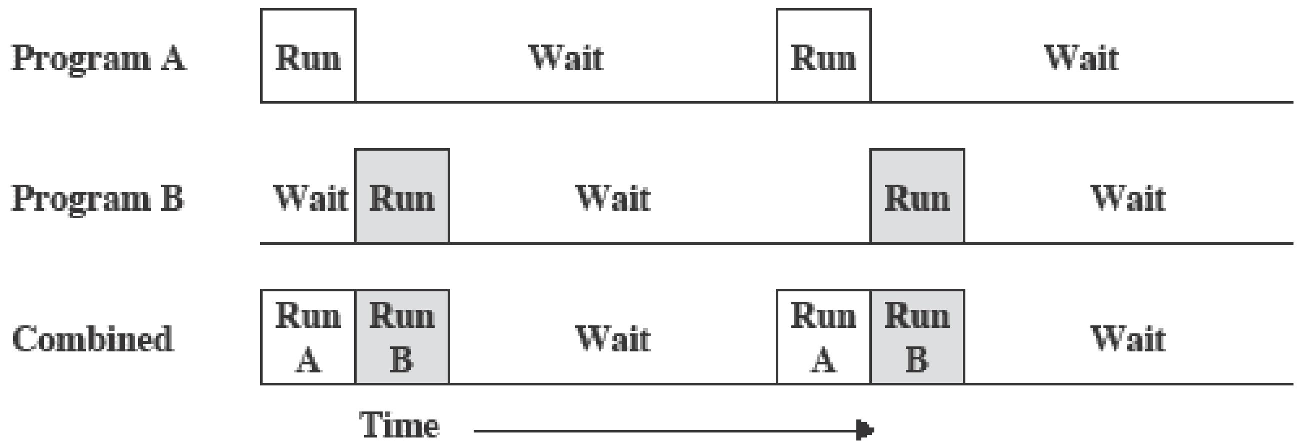
Percent CPU Utilization $= \frac{1}{31} = 0.032 = 3.2\%$

System Utilization Example

  - The calculation concerns a program that processes a file of records and performs, on average, 100 machine instructions per record. In this example the computer spends over 96% of its time waiting for I/O devices to finish transferring data to and from the file.

Evolution of Operating System [Multiprogramming batch system]

- When one job needs to wait for I/O, the processor can switch to the other job

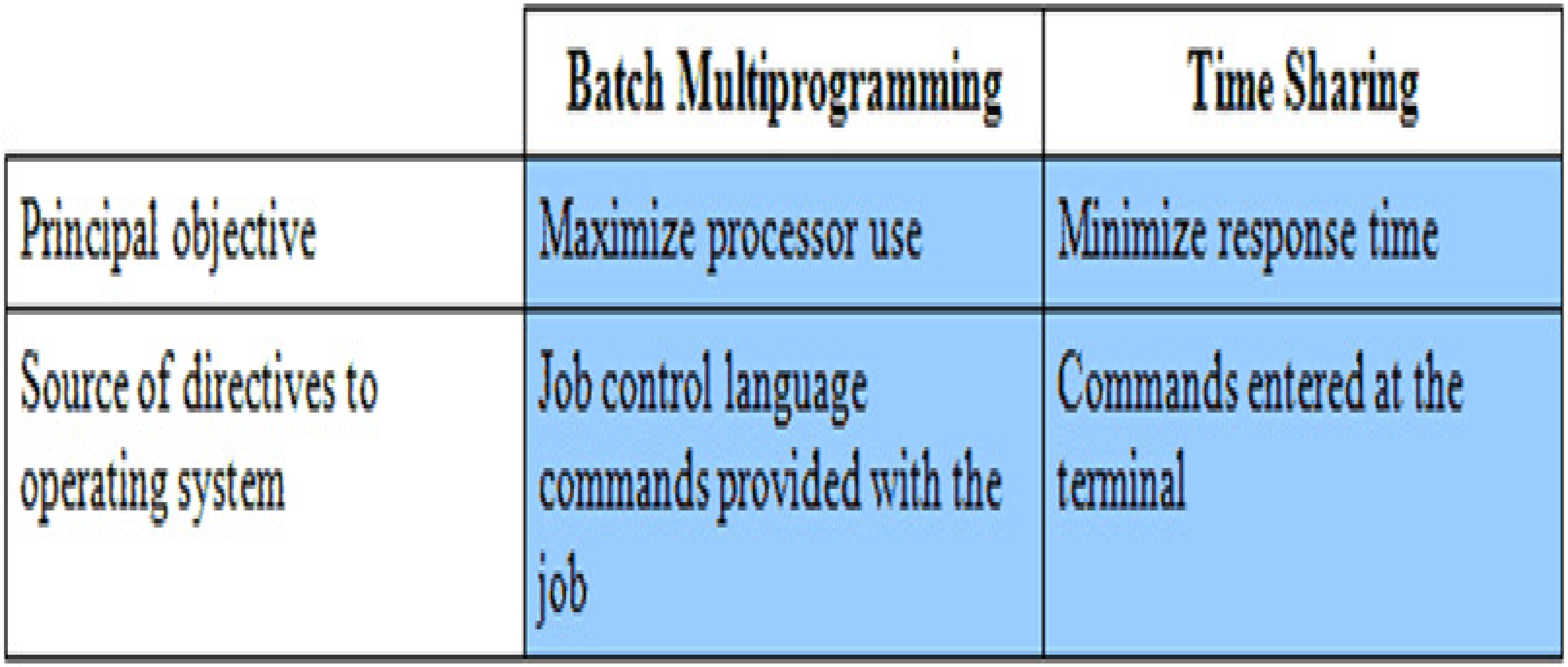


(b) Multiprogramming with two programs

Evolution of Operating System [Time Sharing system]

- ❑ Using multiprogramming to handle multiple interactive jobs
- ❑ Processor’s time is shared among multiple users
- ❑ Multiple users simultaneously access the system through terminals
- ❑ That option was not available in the 1960s, when most computers were big and costly. Instead, time sharing was developed.

Batch Processing VS Time Sharing



	Batch Multiprogramming	Time Sharing
Principal objective	Maximize processor use	Minimize response time
Source of directives to operating system	Job control language commands provided with the job	Commands entered at the terminal

Problems and Issues

- Multiple jobs in memory must be protected from each other’s data
- File system must be protected so that only authorised users can access
- Contention for resources must be handled
- Printers, storage etc

Modern Operating Systems

- Different Architectural Approaches
  - Microkernel architecture
  - Multithreading
  - Symmetric multiprocessing
  - Distributed operating systems
  - Object-oriented design

Modern Operating Systems

- Microkernel architecture
  - Most early OS are a monolithic kernel
  - Most OS functionality resides in the kernel.
  - Typically, a monolithic kernel is implemented as a single process, with all elements sharing the same address space.
  - A microkernel assigns only a few essential functions to the kernel : Address spaces, Interprocess communication (IPC), Basic scheduling.
  - Other OS services are provided by processes, sometimes called servers, that run in user mode and are treated like any other application by the microkernel.
  - This approach decouples kernel and server development.
  - The microkernel approach simplifies implementation, provides flexibility, and is well suited to a distributed environment.

Modern Operating Systems

- Multithreading
  - Process is divided into threads that can run concurrently
  - Thread: Dispatchable unit of work executes sequentially and is interruptible.
  - Process is a collection of one or more threads. This corresponds closely to the concept of a program in execution.
  - By breaking a single application into multiple threads, the programmer has great control over the modularity of the application and the timing of application-related events.
  - Multithreading is useful for applications that perform a number of essentially independent tasks that do not need to be serialized.

Modern Operating Systems

- Symmetric multiprocessing (SMP)
  - An SMP can be defined as a standalone computer system with the following characteristics:
    - multiple processors.
    - These processors share same main memory and I/O facilities.
    - All processors can perform the same functions (hence the term symmetric).
  - The OS of an SMP schedules processes or threads across all of the processors.
  - Advantages are:
    - Performance : Allowing parallel processing
    - Availability : Failure of a single process does not halt the system
    - Incremental Growth : Additional processors can be added.
    - Scaling

Modern Operating Systems

- Distributed Operating Systems
  - Provides the illusion of
    - a single main memory space and
    - single secondary memory space
    - unified access facilities, such as a distributed file system
  - Early stage of development multiple processors.
  - Although clusters are becoming increasingly popular, the state of the art for distributed operating systems lags that of uniprocessor and SMP operating systems.

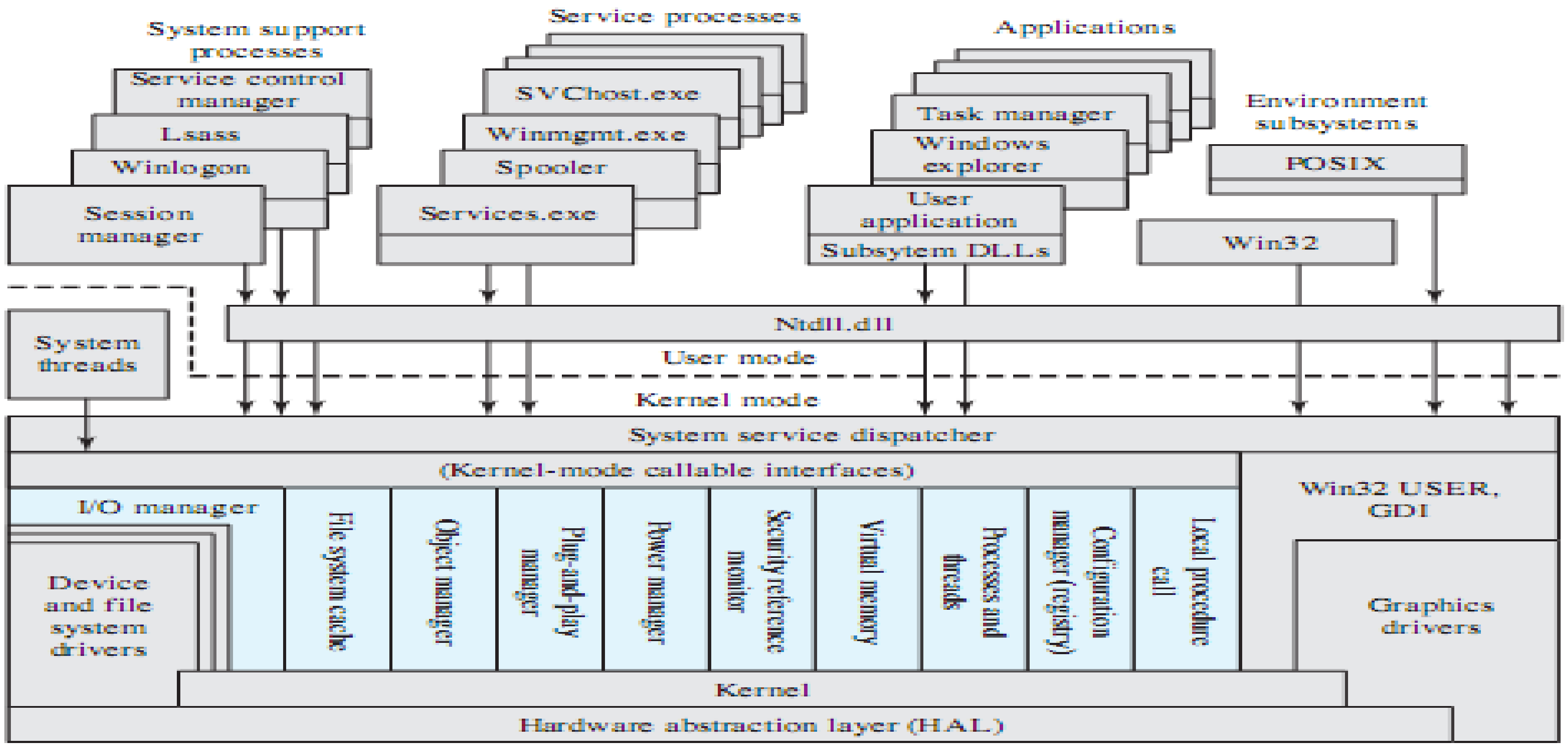
Modern Operating Systems

- Object-oriented design
  - Used for adding modular extensions to a small kernel
  - Enables programmers to customize an operating system without disrupting system integrity

Microsoft Windows Overview

- Single-User Multitasking
  - From Windows 2000 on Windows development developed to exploit modern 32-bit and 64-bit microprocessors. It is rival to mainframes in speed, hardware variety, and memory capacity.
  - Designed for single users who run multiple programs
  - Main drivers are:
    - Increased memory and speed of microprocessors
    - Support for virtual memory

Microsoft Windows Architecture



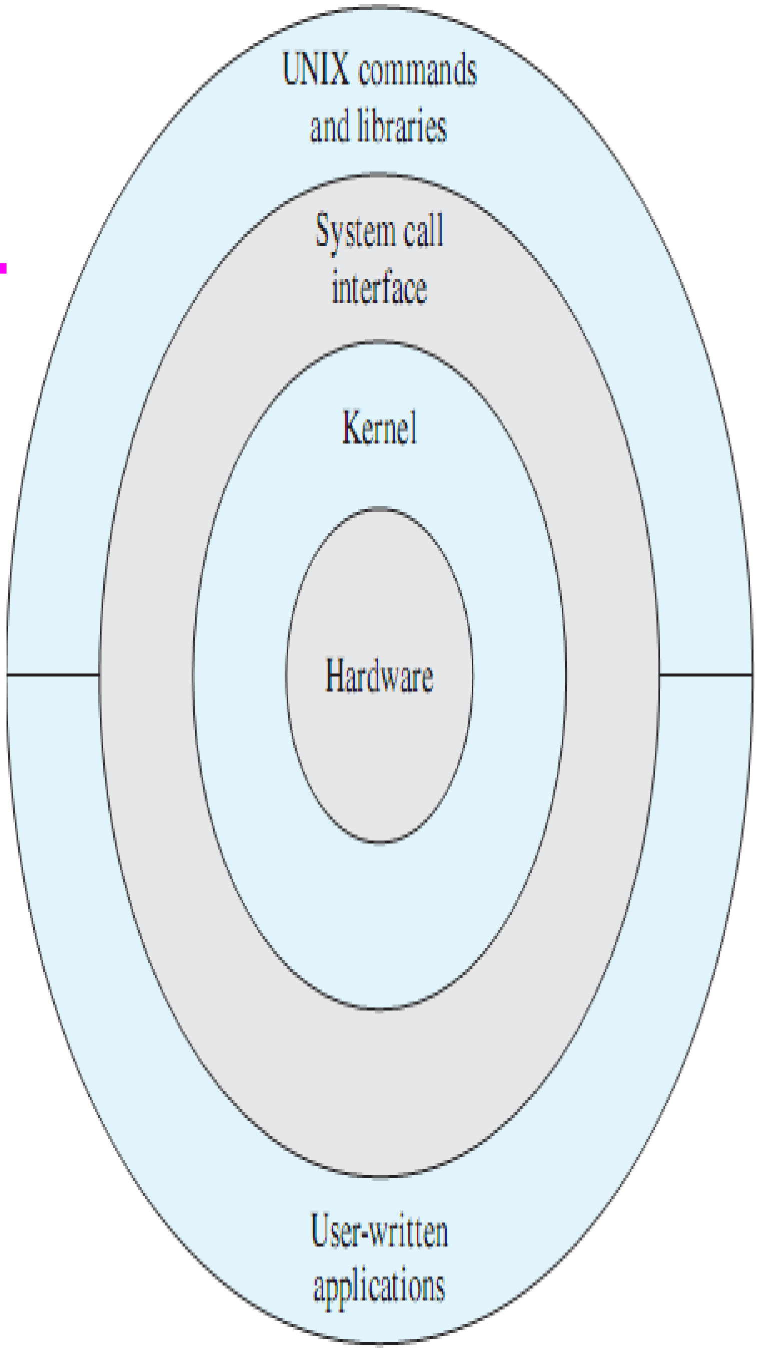
Lsass = local security authentication server
POSIX = portable operating system interface
GDI = graphics device interface
DLL = dynamic link libraries

Colored area indicates Executive

Figure 2.13 **Windows and Windows Vista Architecture [RUSS05]**

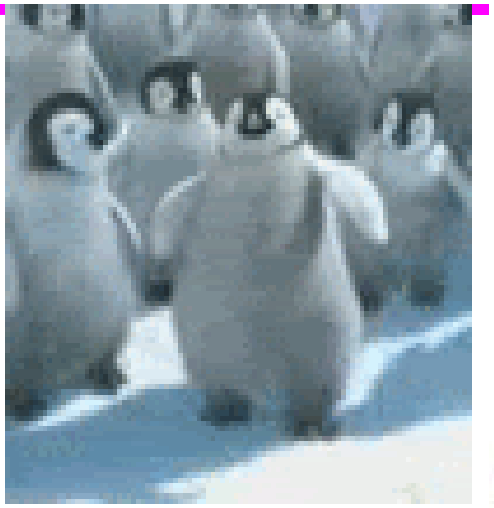
Unix System Architecture

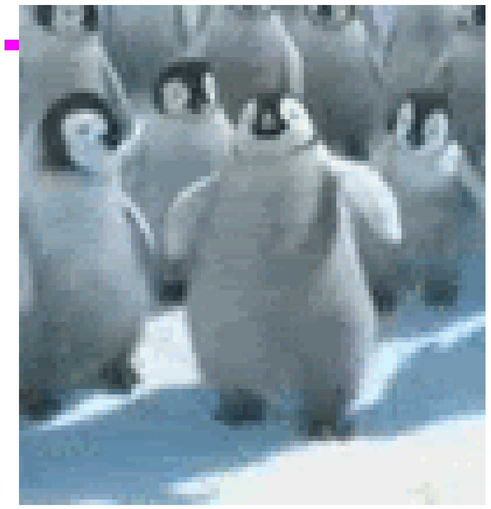
- UNIX also comes equipped with a number of user services and interfaces that are considered part of the system.
- These can be grouped into the shell, other interface software, and the components of the C compiler (compiler, assembler, loader).
- The layer outside of this consists of user applications and the user interface to the C compiler

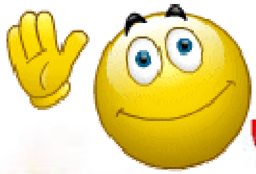


Linux System Architecture

- Modular Monolithic Kernel
  - Although monolithic, the kernel is structures as a collection of modules : Loadable modules, An object file which can be linked and unlinked at run time
  - Characteristics of modules: Dynamic Linking, Stackable modules
  - Dynamic linking : A kernel module can be loaded and linked into the kernel while the kernel is already in memory and executing.
    - A module can also be unlinked and removed from memory at any time.
    - Saves kernel memory.
  - Stackable modules: The modules are arranged in a hierarchy.
    - Individual modules serve as libraries when they are referenced by client modules higher up in the hierarchy, and as clients when they reference modules further down.



THANKS




THANKS

THANKS

THANKS