Introduction to Operating System

Sagarmatha College of Science and Technology

Objectives of the course

- Introduce the underlining principles of an uni-processor operating system.
- Explores the concept of multiprogramming, virtual memory and resource management.
- Case study of some well-known operating systems.

Text Books and Materials

Text Books:

1. Tanenbaum, A. S., *Modern Operating Systems, Second Edition, P*
2. Silberschatz, A. and P. B. Galvin, *Operating System Concepts, Sixth* Edition, John-Wiley. (not Java edition)

References :

1. Research and Technical Papers
2. Stallings, W., *Operating Systems, Fourth Edition, Pearson.*
3. Deitel H. M., *Operating Systems, Second Edition, Addition Wesley.*
4. Tanenbaum, A. S. and Woodhull, A. S., *Operating Systems Design and Implementation, Second Edition, PHI.*

Lab References:

Linux Manual, Linux Programming Manual and C book.

Unit 1
Introduction to Operating System(OS)

Objective of the Chapter

- After reading this unit, you will be able to
  - Define and describe the function of operating system
  - Explain the evolution of OS
  - Define the structure of OS

Computer System

Computer system can be divided into four components

Hardware: provides basic computing resources

-CPU, memory, I/O devices

Operating system: Controls and coordinates use of hardware among various applications and users

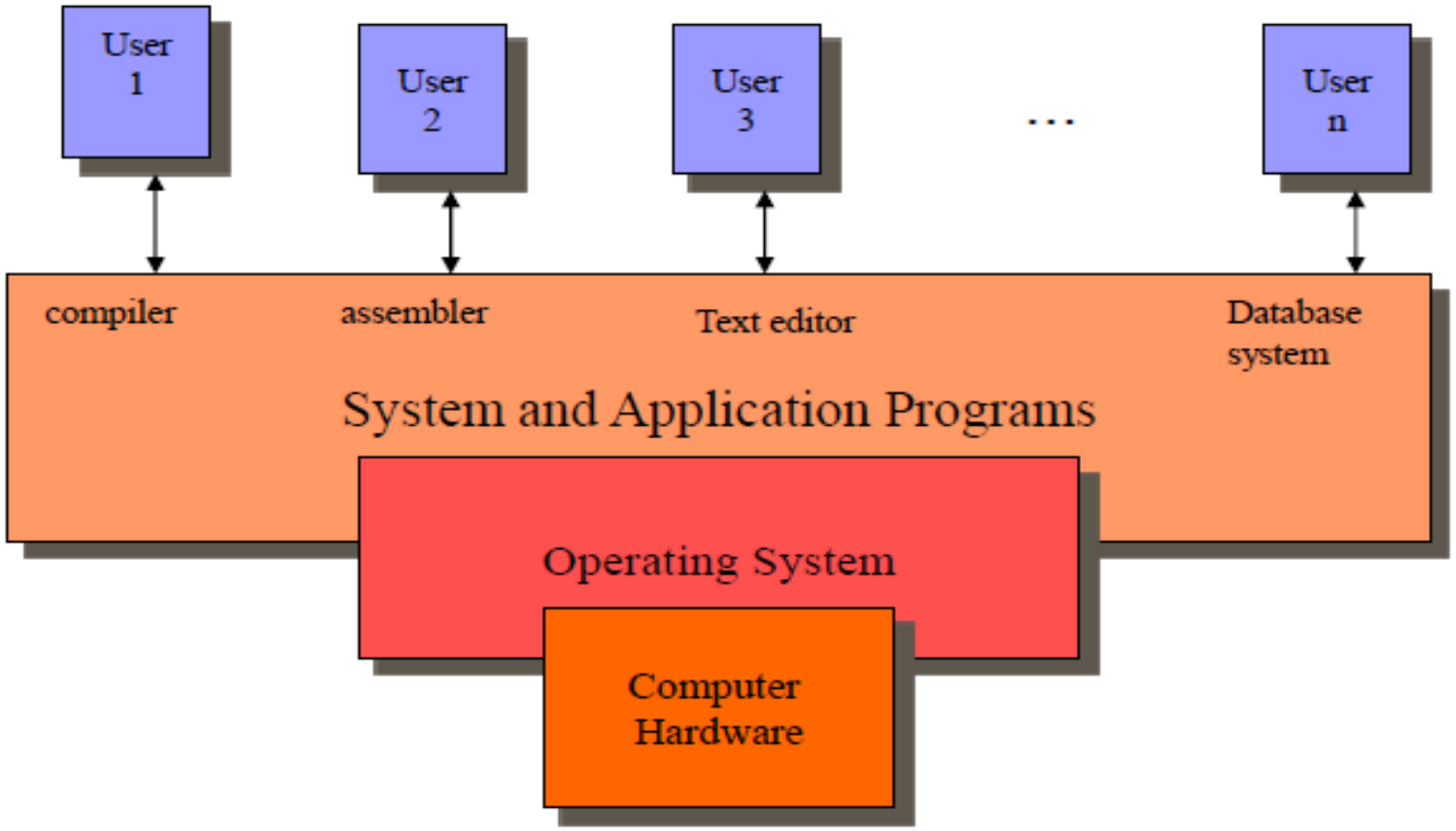
Application programs: define the ways in which the system resources are used to solve the computing problems of the users

-Word processors, compilers, web browsers, database systems, video games

Users:

-People, machines, other computers

Where does OS sits?



What is Operating System(OS)?

- An OS is a program that acts an intermediary between the user of a computer and computer hardware.
- OS is a system software which provides an environment to help the user to execute the programs.
  - Extend machine
- OS simplifies and manages the complexity of running application programs efficiently.
- OS is a resource manager which allocates and manages various resource like CPU, Main memory, I/O device, Hard disk etc.- resource manager

Function of OS

- OS perform the following important functions
  - **CPU Management**: It means assigning **processor** to different **task** which has to be performed by computer.
  - **Memory management:** It means allocation of main memory to different task to execute.
  - **Input/output management**: it means co-ordination between different input and output devices while one or more program are running.
  - **File system management:** OS is responsible for maintenance of file system.

Operating Systems Views

- **As a Resource manager**
  - to allocate resources (software and hardware) of the computer system and manage them efficiently.

- **As an extended machine**
  - Controls execution of user programs and operation of I/O devices.

OS as an Extended Machine

- OS function is to present the user with the equivalent of an extended machine or virtual machine that is easier to program than the underlying hardware.
- OS creates higher-level abstraction for programmer.
- What are the right abstractions? How to achieve this?

OS as an Extended Machine(cont…)

- **Example: (Floppy disk I/O operation)**
  - disks contains a collection of named files
  - each file must be open for READ/WRITE
  - after READ/WRITE complete close that file
  - no any detail to deal

- OS shields the programmer from the disk hardware and presents a simple file oriented interface.

OS as a Resource Manager

- OS primary function is to manage all component of a complex system.
- *What happens if three programs try to print their output on the same printer at the same time?*
- *What happens if two network users try to update a shared document at same time?*
- OS should take care of these.

OS as resource Manager(cont…)

- OS as a resource manager does the following:
  - Virtualizes resource so multiple users/ applications can share the resources
  - Protect applications from one another
  - Provide efficient and fair access to resources.
- *To do these task, OS designer should include the appropriate policy and mechanism in OS design*

OS Evolution(History)

- Present OS haven’t been developed overnight.
- Like any other system, OS also have evolved over a period of time.
- Evolution starts from the very primitive OS to the present most complex and versatile OS

OS Evolution(History)

Computer Generation, Component and OS Types

1st (1945-55)	Vacuum Tubes	User Driven
2nd (1955-65)	Transistor	Batch
3rd (1965-80)	IC	Multiprogramming
4th (1980-present)	PC	Client Server/Distributed

Batch System

- In batch system, operator hired to run computer, the user prepared a job –which consisted of the program, the data, and some control information about the nature of the jobs – and submitted it to the computer operator.
- Output appears after some minutes and user collects the outputs from the operator.
- Why the name batch system?
  - Batch: Group of jobs submitted to machine together
  - Operator collects jobs; orders efficiently; runs one at a time
- Here the OS is very simple and its only task is to transfer control from one job to another.

Batch System

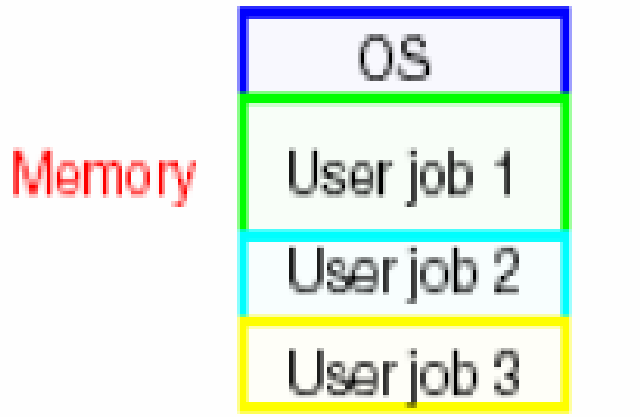
- Advantages:
  - Average setup costs over many jobs
  - Operator more skilled at loading tapes
  - Keep machine busy while programmer thinks
- Problem:
  - Lack of interaction between the user and the job.
  - User must wait for results until batch collected and submitted
  - (*If bug receive, memory and register dump; submit job again!)*

Multiprogramming operating system (MOS)

- A job pool on the disk consist of a number of jobs that are ready to be executed.
- Subset of these jobs reside in memory during execution.
- OS picks one job ‘x’ and assign it to CPU to executes in memory.
- When this job x needs an I/O operation to complete, the CPU is idle.
- Now OS switches the CPU to another job (y) to utilize it until the previous job (x) resume (ready) to execute.
- As long as there are jobs in memory waiting for CPU, the CPU is never idle.
- Choosing one out of several jobs in memory for execution is called CPU scheduling (CPU execute jobs)

Multiprogramming (MOS)

What is the main function of Multiprogramming OS?



New OS functionality

- Job scheduling policies
- Memory management and protection (virtual memory)

Timesharing OS

- Problem with MOS:
  - increasing number of users who want to interact with programs while they are running.
  - humans' time is expensive don't want to wait.
- Solution

  -Timesharing (a variant of Multiprogramming)
- *The CPU executes multiple jobs by switching among them, but the switches occurs so frequently that the user can interact with each program while it is running.*

Time sharing OS (MOS)

- It is the logical extension of Multiprogramming
- OS provides each user one time slice(slot or share) and each process execute for one time slot in one round.
- The CPU switches from one job to another at the end of time slice.
- The switching is so fast that the user gets the illusion that the CPU is executing only one user's job

Timesharing OS(cont…..)

- Goal:
  - Improve user's response time (interaction)
- Advantage:
  - Users easily submit jobs and get immediate feedback
  - Reduces CPU idle time
- Disadvantage:
  - Problem of reliability
  - Question of security and integrity of user programs and data
  - Problem of data communication
- New OS functionality
  - More complex job scheduling, memory management
  - Concurrency control and synchronization

Personal Computers (PC-OS)

- The main mode of use of PC is by a single user(as name personal computer).
- The System of PC consist of two parts:
  - BIOS(basic input output system) which is stored in a ROM.
  - The other part called DOS is stored in a floppy disk or hard disk.
- BIOS does what is known is power on self test. Having done this it reads small portion of OS from the disk known as boot portion. Then DOS do all other task such as file management, disk management, CPU management.

Personal Computers (PC-OS)

- Dedicated machine per user
- CPU utilization is not a prime concern
  - *Hence, some of the design decision made for advanced system may not be appropriate for these smaller system.*
  - *But other design decisions, such as those for security, are appropriate because PCs can now be connected to other computer and users through the network and the web.*
- There are the different types of OS for PC such as Windows, MacOS, UNIX, Linux.

Multiprocessor OS

- Most system to date are single-processor system, which will be having only one CPU.
- However there is a need for multiprocessor system.
- A System having more than one processor and all these processor will share the computer bus, the clock and sometime memory, is Multiprocessor OS.

Distributed OS

- It is the logical extension of multiprocessor system
- In such system, all computation of jobs are distributed among several process.
- Here processor don’t share memory and clock. They communicate with one another via communication lines.
- *Benefits:*
  - *Resource sharing*
  - *Computation speedup*
  - *Reliability*
  - *Communication*

Mainframe OS

- The operating systems for mainframes are heavily oriented toward processing many jobs at once, most of which need prodigious amounts of I/O.
- They typically offer three kinds of services: batch, transaction processing, and timesharing.
- A batch system is one that processes routine jobs without any interactive user present. Claims processing in an insurance company or sales reporting for a chain of stores is typically done in batch mode.
- Transaction-processing systems handle large numbers of small requests, for example, check processing at a bank or airline reservations.
- Timesharing systems allow multiple remote users to run jobs on the computer at once, such as querying a big database.

Server OS

- One level down are the server operating systems.
- They run on servers, which are either very large personal computers, workstations, or even mainframes.
- They serve multiple users at once over a network and allow the users to share hardware and software resources.
- Servers can provide print service, file service, or Web service.
- Internet providers run many server machines to support their customers and Websites use servers to store the Web pages and handle the incoming requests.
- Typical server operating systems are Solaris, FreeBSD, Linux and Windows Server 201x

Network OS

- A Network Operating System runs on a server and provides the server the capability to manage data, users, groups, security, applications, and other networking functions.
- The primary purpose of the network operating system is to allow shared file and printer access among multiple computers in a network, typically a local area network (LAN), a private network or to other networks.
- Advantages
  - *Centralized servers are highly stable.*
  - *Upgrades to new technologies and hardware can be easily integrated into the system.*
  - *Remote access to servers is possible from different locations and types of systems.*
- Disadvantages
  - *High cost of buying and running a server.*
  - *Dependency on a central location for most operations.*
  - *Regular maintenance and updates are required.*

Real-Time OS

- These systems are characterized by having time as a key parameter.
- Real-time systems are used when there are rigid time requirements on the operation of a processor or the flow of data and real-time systems can be used as a control device in a dedicated application.
- A real-time operating system must have well-defined, fixed time constraints, otherwise the system will fail.
- For example, Scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, air traffic control systems, etc.
- There are two types of real-time operating systems.
  - **Hard real-time systems**
    - Hard real-time systems guarantee that critical tasks complete on time.
  - **Soft real-time systems**
    - occasional deadline is acceptable and does not cause any permanent damage.

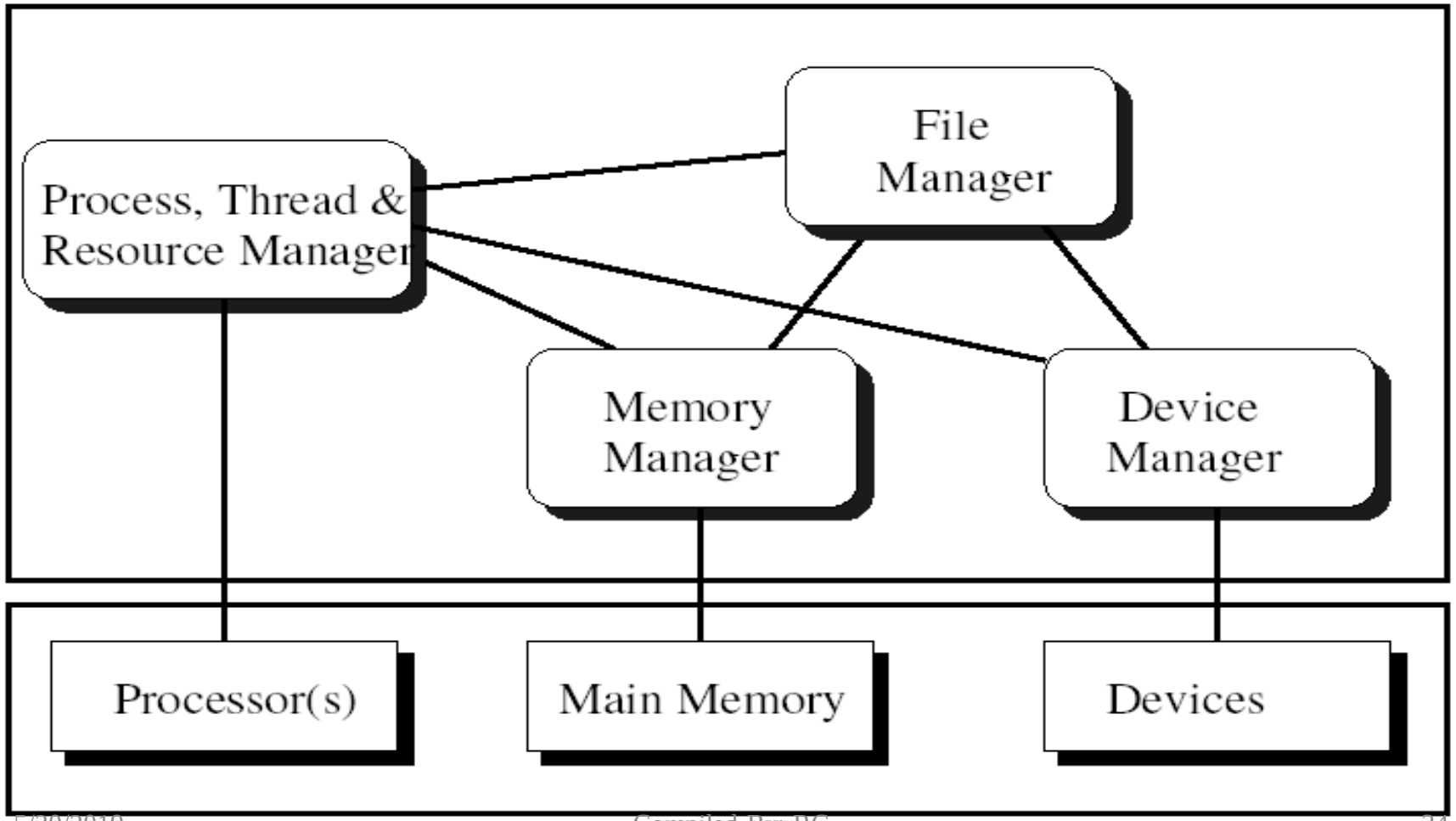
Embedded OS

- Embedded systems run on the computers that control devices that are not generally thought of as computers and which do not accept user-installed software.
- Typical examples are microwave ovens, TV sets, cars, DVD recorders, traditional phones, and MP3 players
- Systems such as Embedded Linux, QNX and VxWorks are popular in this domain.

Smart Card OS

- The smallest operating systems run on smart cards, which are credit-card-sized devices containing a CPU chip. They have very severe processing power and memory constraints.
- Some are powered by contacts in the reader into which they are inserted, but contactless smart cards are inductively powered, which greatly limits what they can do.
- Some of them can handle only a single function, such as electronic payments, but others can handle multiple functions.
- Often these are proprietary systems.

OS Organization or Components



Process Management

- A process is a program in execution.
- For example
  - A batch job is a process
  - A time-shared user program is a process
- A system task (e.g. spooling output to printer) is a process.
- Remember a program itself is not a process rather it is a passive entity.

Process Management

- 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 when it is running.
- When the process completes, the OS reclaims all the resources.
- Process manager (OS) handle all these responsibility

Process Management

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

- Process creation and deletion.
- Process suspension and resumption.
- Provision of mechanisms for:
  - Process synchronization
  - Process communication

Memory Management

Memory is a large array of words or bytes, each with its own address.

- It is a repository of quickly accessible data shared by the CPU and I/O devices.
- Main memory is a volatile storage device. It loses its contents in the case of system failure.

Memory Management

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.
- Allocate and deallocate memory space as needed.

File Management

A file is a collection of related information defined by its creator. Commonly, files represent programs (both source and object forms) and data.

- Most visible component of OS. Computers can store information on several different types of physical media (e.g. magnetic tap, magnetic disk, CD etc).
- A file a logical storage unit, which abstract away the physical properties of its storage device.

File Management

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 Management

All computers have physical devices for acquiring input and producing output.

- The I/O system consists of:
  - A memory management system for buffering and caching system
  - A general device-driver interface
  - Drivers for specific hardware devices
  - Disk management

Secondary Storage Management

- The computer system must provide *secondary storage* to back up main memory since main memory is volatile.

- 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

System calls

- System calls provide the interface between a process and the operating system.
  - These calls are generally available as assembly language instructions.
  - Some systems also allow to make system calls from a high level language, such as C.
- Example:
  - count = read(fd, buffer, nbytes)

Types of System Calls

- Process control
  - load, execute, abort, end, create process, allocate and free memory, wait event etc.
- Example
  - Pid=fork(), Creates a child process identical to parents.
  - exits(status), terminate the process execution and return the status.
  - Etc.

Types of System Calls

- File management:
  - Create file, delete file, open, close, read, write, get file attribute etc.
- Example
  - Read(fd, buffer, nbytes)
  - Write(fd, buffer, nbytes)
  - fd=open(file, how,….)
  - Close(fd)
  - etc.

Directory and file system management

Call	Description
s = mkdir(name, mode)	Create a new directory
s = rmdir(name)	Remove an empty directory
s = link(name1, name2)	Create a new entry, name2, pointing to name1
s = unlink(name)	Remove a directory entry
s = mount(special, name, flag)	Mount a file system
s = umount(special)	Unmount a file system

Miscellaneous

Call	Description
s = chdir(dirname)	Change the working directory
s = chmod(name, mode)	Change a file's protection bits
s = kill(pid, signal)	Send a signal to a process
seconds = time(&seconds)	Get the elapsed time since Jan. 1, 1970

UNIX	Win32	Description
fork	CreateProcess	Create a new process
waitpid	WaitForSingleObject	Can wait for a process to exit
execve	(none)	CreateProcess = fork + execve
exit	ExitProcess	Terminate execution
open	CreateFile	Create a file or open an existing file
close	CloseHandle	Close a file
read	ReadFile	Read data from a file
write	WriteFile	Write data to a file
lseek	SetFilePointer	Move the file pointer
stat	GetFileAttributesEx	Get various file attributes
mkdir	CreateDirectory	Create a new directory
rmdir	RemoveDirectory	Remove an empty directory
link	(none)	Win32 does not support links
unlink	DeleteFile	Destroy an existing file
mount	(none)	Win32 does not support mount
umount	(none)	Win32 does not support mount
chdir	SetCurrentDirectory	Change the current working directory
chmod	(none)	Win32 does not support security (although NT does)
kill	(none)	Win32 does not support signals
time	GetLocalTime	Get the current time

Operating System Structure

- Modern OS should be developed carefully due to their size and complexity.
- A common approach is to divide the systems into small components (layered approaches).
- Monolithic and layered are two approaches used.

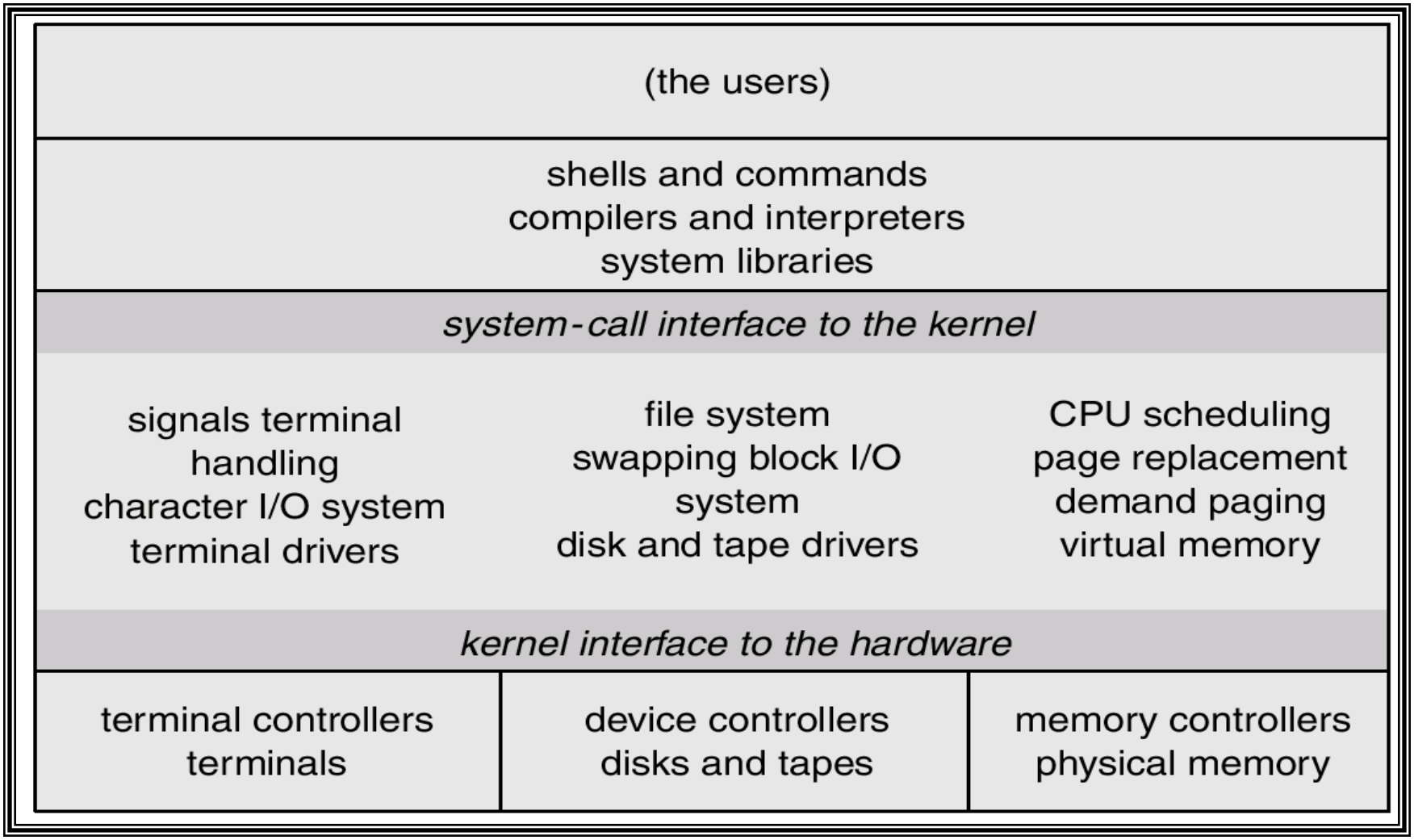
Monolithic structure

- No structure
- OS written as collection of procedures.
- Each procedure can call others
- No information hiding and protection
- Even a little structure is possible
  - Main program that invokes others
  - A set of services that carry out system calls
  - A set of utilities services
- MS DOS is an example.

Layered approaches

- Unix System is an example.
- The UNIX OS consists of two separable parts.
  - Systems programs – use kernel supported system calls to provide useful functions such as compilation and file manipulation.
  - The kernel
    - Consists of everything below the system-call interface and above the physical hardware
    - Provides the file system, CPU scheduling, memory management, and other operating-system functions; a large number of functions for one level.

Unix System Structure



Layered approaches

- The operating system is divided into a number of layers (levels), each built on top of lower layers. The bottom layer (layer 0), is the hardware; the highest (layer N) is the user interface.
- Layered approach provides modularity. With modularity, layers are selected such that each layer uses functions (operations) and services of only lower-level layers.
- Each layer is implemented by using only those operations that are provided lower level layers.
- The major difficulty is appropriate definition of various layers.

Microkernel System Structure

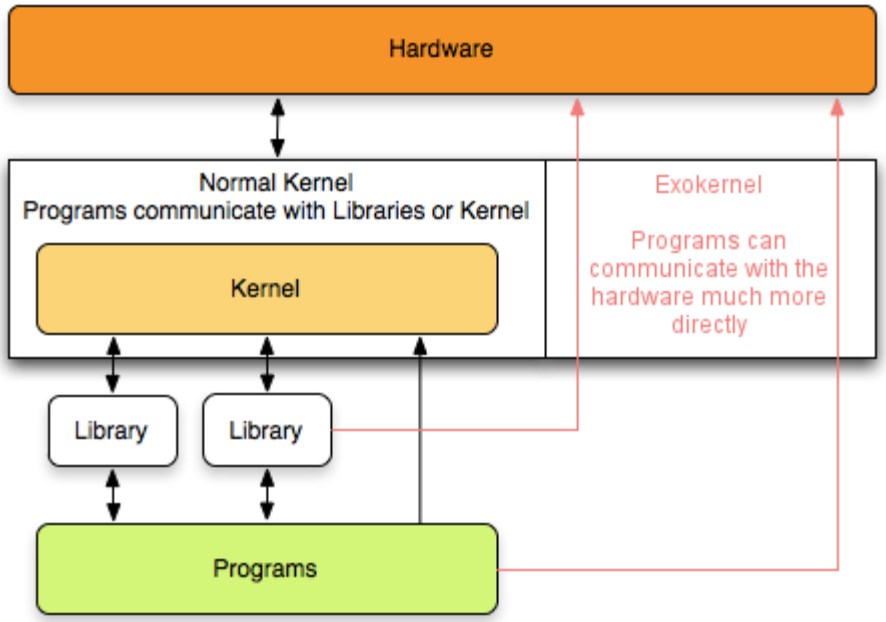
- Moves as much from the kernel into "*user*" space.
- Communication takes place between user modules using message passing.
- Benefits:
  - Easier to extend a microkernel
  - Easier to port the operating system to new architectures
  - More reliable (less code is running in kernel mode)
  - More secure

Nano-Kernel or Pico kernel

- The term *nanokernel* was coined by Jonathan S. Shapiro
- kernel where the total amount of kernel code, i.e. code executing in the privileged mode of the hardware, is very small.
- The term *picokernel* was sometimes used to further emphasize small size.
- A nanokernel delegates virtually all services — including even the most basic ones like interrupt controllers or the timer — to device drivers to make the kernel memory requirement even smaller than a traditional microkernel

Exokernel

- Exokernels are tiny, since functionality is limited to ensuring protection and multiplexing of resources, which are vastly simpler than conventional microkernels' implementation of message passing



Client-Server Model

- A slight variation of the microkernel idea is to distinguish two classes of processes, the **servers**, each of which provides some service, and the **clients**, which use these services.
- This model is known as the **client-server** model.
- Often the lowest layer is a microkernel, but that is not required.
- The essence is the presence of client processes and server processes.
- Communication between clients and servers is often by message passing.

Virtual Machines

- The heart of the system, known as the **virtual machine monitor**, runs on the bare hardware and does the multiprogramming, providing not one, but several virtual machines to the next layer up.
- However, unlike all other operating systems, these virtual machines are not extended machines, with files and other nice features.
- Instead, they are *exact* copies of the bare hardware, including kernel/user mode, I/O, interrupts, and everything else the real machine has.

Open-Source OS

- Open-source operating systems are those made available in source-code format rather than as compiled binary code.
- Linux is the most famous open-source operating system, while Microsoft Windows is a well-known example of the opposite closed-source approach.
- Benefits
  - More secure
  - Faster distribution
  - Error free code

System Programs

- System programs provide a convenient environment for program development and execution.
- They are also called system utilities.
- Some of them are simply user interfaces to system calls others are considerably more complex.
- They can be divided into these categories.
  - File management
  - Status management
  - File modification
  - Programming-language support
  - Program loading and execution
  - Communications

Shell

- It is UNIX command line interpreter.
- It is not a part of OS but it makes heavy use of many operating system features.
- It is also the main interface between a user sitting at his terminal and the operating system, unless the user is using a graphical user interface.
- Many shells exist, including *sh*, *csh*, *ksh*, and *bash*.

Home Works

- What does the CPU do when there are no programs to run?
- Classify the following applications as batch-oriented or interactive.
  - Word processing
  - Generating monthly bank statements
  - Computing pi to a million decimal places
- What is a purpose of system calls?
- What are the different OS structure possible?
- Write about different types of kernels.