Operating System

Lecture Course in Spring Term 2018

Al-Nahrain University

M.Sc. Ahmed Altaher

Operating System

Will have one lecture (Tue 10:30 AM) per week.

Continuous Assessment 40 %

- Two Mid Exams 30 % (2 * 15 %)
  - 15 % (to be confirmed)
  - 15 % (to be confirmed)
- Quiz, Assignments, Class Activity 10 %

Final Exam 60 %

Final Exam + Continuous Assessment towards Final Mark

Supplementary (If allowed): By examination only (100%)

Operating System

Lecture Notes

Assignment Delivery

Etc.

http://mdl.coie-nahrain.edu.iq/

Course Aim

This course aims to introduce the student to the concepts of operating systems.

The topics covered in this course include: Introduction to computer and operating systems; Processes; Threads; IPC, Process Scheduling; Process Synchronization; Deadlocks; Memory Management and Virtual Memory; File Systems: Interface and Implementation; Mass-Storage Structure and Management; and I/O Systems.

Operating System

Week No.	Topics
1	Introduction; Operating system structures
2	Linux Debian / Ubuntu – Virtual Machine
3	Processes
4	Processes ; Threads
5	CPU Scheduling
6	Process Synchronization
7	Process Synchronization; Deadlocks
8	Main Memory
9	Main Memory; Midterm Exam
10	Virtual Memory
11	Virtual Memory; Mass-Storage Structure
12	Mass-Storage Structure
13	File-System Interface
14	File-System Implementation
15	I/O Systems

Operating System

Text Book:

Operating System Concepts, 9th Edition, 2012 by Silberschatz, A., Galvin, P. B., Gagne, G.

Reference:

Operating Systems Internals and Design Principles, 8th Edition, 2015 by William Stallings.

Programming Languages: C / Java

Lecture one /

OS Introduction

AIMS

- what operating systems are ?
- what they do ?
- How they are designed and constructed ?
- what the common features of an operating system are ?
- what an operating system does for the user ?

OS Introduction

what operating systems are ?

- A program that acts as an intermediary between a user of a computer and the computer hardware

- Operating system goals:

  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner

An operating system is large and complex, it must be created piece by piece. Each of these pieces should be a well-delineated portion of the system, with carefully defined inputs, outputs, and functions.

OS Introduction

A computer system can be divided roughly into four components:

- **hardware (CPU, Memory, I/O)**
- **operating system (Control)**
- **application programs (word processors, spreadsheets, compilers, and Web browsers, etc.)**
- **Users**

In other words we can also view a computer system as consisting of hardware, software, and data.

What OS do?

The operating system provides the means for proper use of these resources in the operation of the computer system. An operating system is similar to a government. Like a government, it performs no useful function by itself. It simply provides an ***environment*** within which other programs can do useful work.

OS Introduction

OS

User View

Single, Multi, No Interface

System View

Source Allocator

Kernel (Core of OS)

The one program running at all times on the computer—usually called the **kernel**.

1. System programs, which are associated with the operating system but are not necessarily part of the kernel.

2. Application programs, which include all programs not associated with the operation of the system.

Start

Bootstrap program, (**ROM**), (**EEPROM**) electrically erasable programmable read-only memory, F**irmware. Dynamic random-access memory (DRAM)**.

System processes, interrupt, system call

Operating system structure

One of the most important aspects of operating systems is the ability to multiprogram.

A single program cannot, in general, keep either the CPU or the I/O devices busy at all times. Single users frequently have multiple programs running.

Multiprogramming increases CPU utilization by organizing jobs (code and data) so that the CPU always has one to execute.

OS Introduction

Simple Structure Ms-Dos

Non Simple Structure -- UNIX

Layered Approach

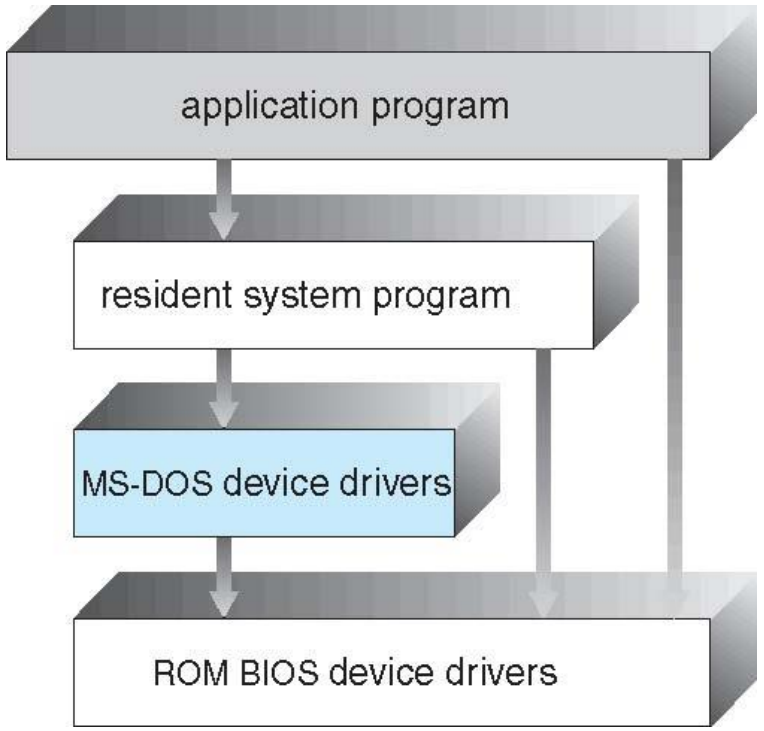
Microkernel System Structure

Modules

Hybrid Systems (Mac OS X, IOS, Android)

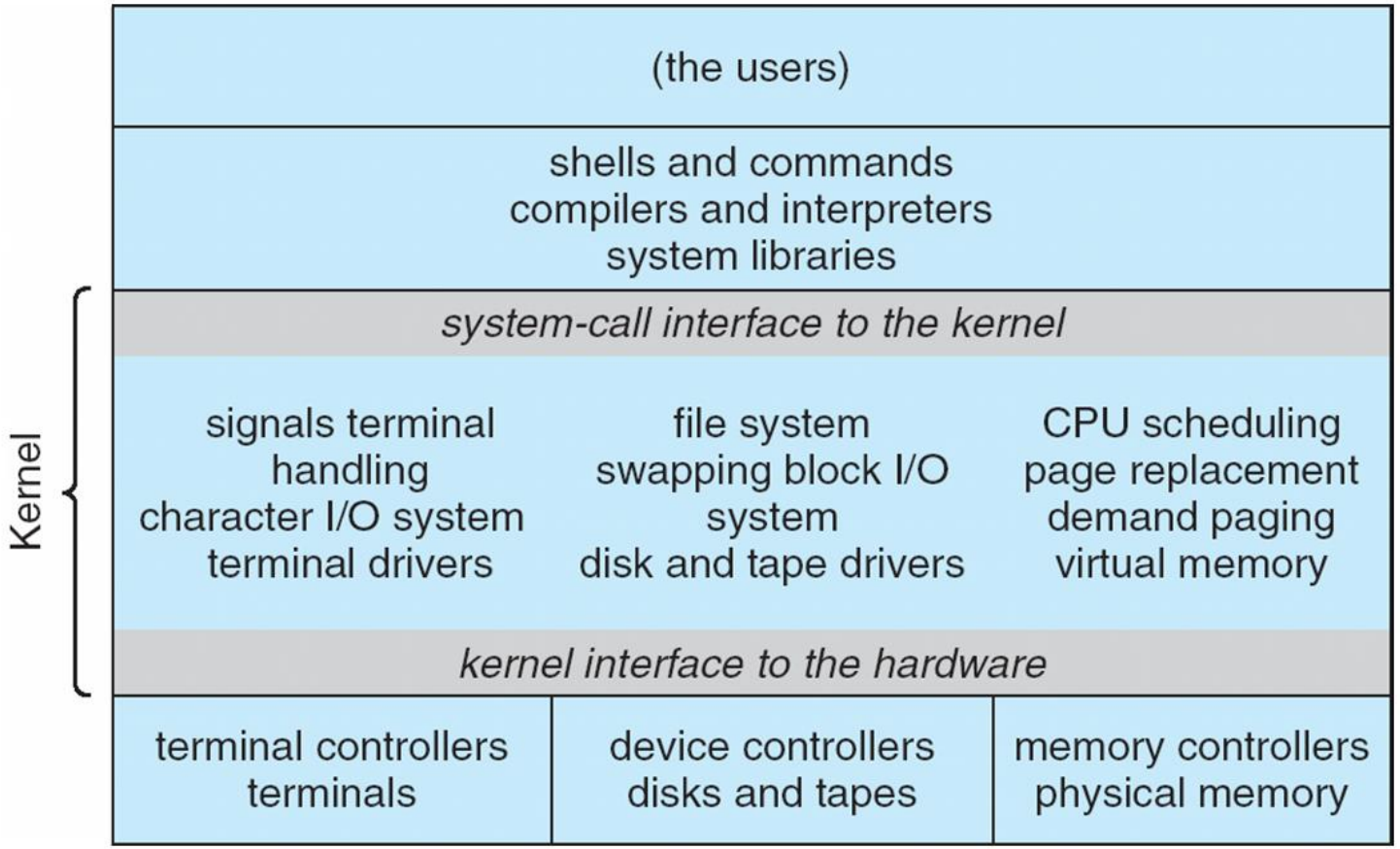
Simple Structure -- MS-DOS

- MS-DOS – written to provide the most functionality in the least space
  - Not divided into modules
  - Although MS-DOS has some structure, its interfaces and levels of functionality are not well separated



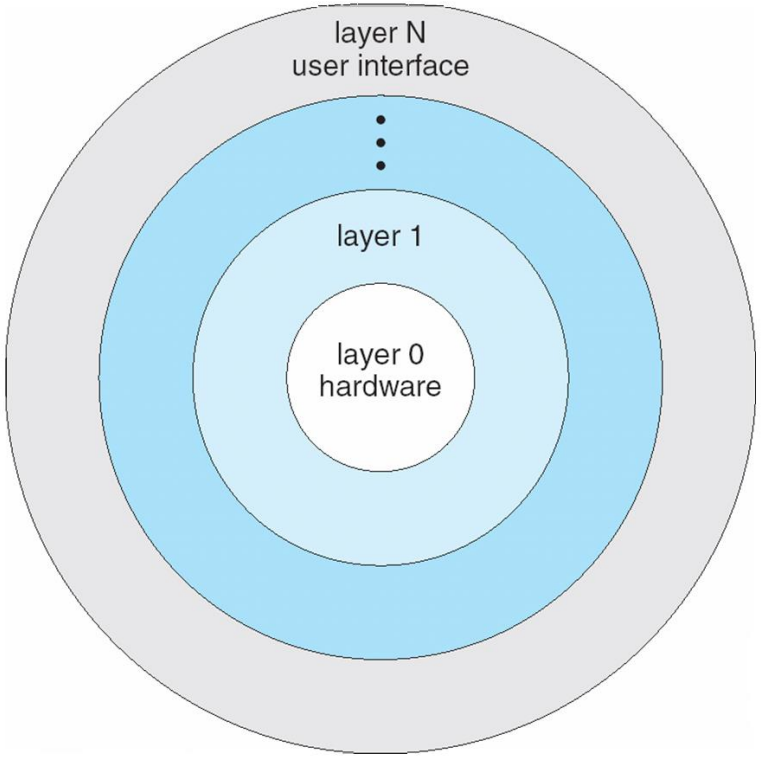
Traditional UNIX System Structure

Beyond simple but not fully layered



Layered Approach

- 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.
- With modularity, layers are selected such that each uses functions (operations) and services of only lower-level layers



Microkernel System Structure

- Moves as much from the kernel into user space
- **Mach** example of **microkernel**
  - Mac OS X kernel (**Darwin**) partly based on Mach
- 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
- Detriments:
  - Performance overhead of user space to kernel space communication

Modules

- Many modern operating systems implement **loadable kernel modules**
  - Uses object-oriented approach
  - Each core component is separate
  - Each talks to the others over known interfaces
  - Each is loadable as needed within the kernel
- Overall, similar to layers but with more flexible
  - Linux, Solaris, windows, etc

OS Introduction

Time sharing (or **multitasking**) is a logical extension of multiprogramming. In time-sharing systems, the 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.

A time-shared operating system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time-shared computer.

Concepts to be considered:

Process, job scheduling, CPU scheduling, swapping, virtual memory,

Operating System Operations

1- Dual-Mode and Multimode Operation

2- Timer

Modern operating systems are **interrupt driven,** events are almost always signaled by the occurrence of an interrupt or a trap.

The interrupt-driven nature of an operating system defines that system's general structure.

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 to deal with the interrupt.

1- Dual-Mode and Multimode Operation

In order to ensure the proper execution of the operating system, we must be able to distinguish between the execution of operating-system code (**kernel mode)** and user defined code (**user mode)**. The approach taken by most computer systems is to provide hardware support that allows us to differentiate among various modes of execution.

2- Timer

We must ensure that the operating system maintains control over the CPU.

We cannot allow a user program to get stuck in an infinite loop or to fail to call system services and never return control to the operating system. To accomplish this goal, we can use a **timer**.

Operating-System Design and Implementation

At the highest level, the design of the system will be affected by the choice of hardware and the type of system: batch, time sharing, single user, multiuser, distributed, real time, or general purpose.

1- Design User Goals

The system should be convenient to use, easy to learn and to use, reliable, safe, and fast.

2- Design System Goals

Achieving the mentioned aims are the system goal (how to achieve it)

Of course, these specifications are not particularly useful in the system design, since there is no general agreement on how to achieve them.

Conclusion

Since the operating system and the users share the hardware and software resources of the computer system, we need to make sure that an error in a user program could cause problems only for the one program running. With sharing, many processes could be adversely affected by a bug in one program.

For example, if a process gets stuck in an infinite loop, this loop could prevent the correct operation of many other processes. More subtle errors can occur in a multiprogramming system, where one erroneous program might modify another program, the data of another program, or even the operating system itself.

Without protection against these sorts of errors, either the computer must execute only one process at a time or all output must be suspect. A properly designed operating system must ensure that an incorrect (or malicious) program cannot cause other programs to execute incorrectly.

Chapter One & Two

Exercises & Practice Exercises

Next Lecture

Virtual Machine, Linux Debian – Ubuntu

Installation, Command Line