Lecture 1

First Meeting & Introduction/Overview I

Lecture Information

Ceng328 *Operating Systems* at February 17, 2011

Dr. Cem Özdoğan
Computer Engineering Department
Çankaya University

Contents

First Meeting I

- CENG 328 Operating Systems Spring 2011
- THURSDAY 10:40-12:30 (T1) B301/302
- FRIDAY 08:40-10:30 (T2) B308/309
- TUESDAY 12:40-14:30 (L1) MPLab
- FRIDAY 12:40-14:30 (L2) MPLab
- FRIDAY 14:40-16:30 (L3) MPLab
- Instructor: Cem Özdoğan, Department of Materials Science and Engineering, A318
- TA: Efe Çiftçi
- WEB page:
  http://siber.cankaya.edu.tr/ozdogan/OperatingSystems/spring2011/index.html
- Announcements: Watch this space for the latest updates.

  `Pazar 13.Subat.2011 23:42 In the first lecture, there will be first metting and Introduction/ Overview. The laboratory notes for the first week is published, see Course Schedule section.`

First Meeting II

- Important announcements will be posted to the **Announcements section of the web page**, so please check this page frequently.
- You are responsible for all such announcements, as well as announcements made in lecture.
- All/Some the example c-files (for lecturing and lab. sessions) will/may be accessible via the link.
- The tutorial link is active.
- Anyone wants to get a live CD without installing linux, download from local server.
  - Ubuntu live CD
  - Pardus live CD

Lecture Information

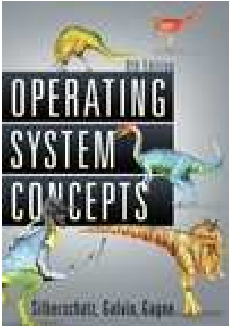
- There are two groups for lecturing, you may attend any one of the lecture hours.
- But, “Please” attend your predefined sessions regularly.
- You will be expected to do significant programming assignments, as well as run programs we supply and analyse the output.
- These programs will be written in C programming language. For programming assignments, other languages will be accepted (such as Java, C++, but no programming assistance will be given).
- The UNIX operating system will be introduced to you first in the lab sessions.
- You MAY have quizzes (10-15 minutes, may be less; but not scheduled as before) for **the previous lecture/chapter’s subjects**.
- There won’t be any make-up for these quizzes.

Overview

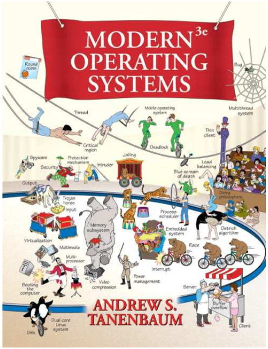
- Ceng 328 is intended as a general introduction to the techniques used to implement operating systems and related kinds of systems software.
- Among the topics covered will be;
  - basic operating system structure,
  - process and thread synchronization,
  - process scheduling and resource management,
  - process management (creation, synchronization, and communication),
  - memory management techniques, main-memory management, virtual memory management,
  - file-system structure,
  - control of disks and other input/output devices,
  - deadlock prevention, avoidance, and recovery.
- This course assumes familiarity with basic computer organization (e.g., processors, memory, and I/O devices).

Text Book

- **Required:** Readings will be assigned in Operating System Concepts, 8th Edition by Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, John Wiley and Sons, January 2008.
- **Recommended:** Modern Operating Systems, 3rd Edition by Andrew S. Tanenbaum, Prentice Hall, 2008. Another frequently used text book that covers the same material with a different approach



Required



Recommended

Grading Criteria & Policies

- There will be a midterm and a final exam, will count 20% and 40% of your grade, respectively.
- Quiz: 15% (worst of the quizzes will be discarded).
- Assignments (or Term Project): 15%.
- Attendance is required and constitutes part of your course grade; 10%. Attendance is not compulsory, but you are responsible for everything said in class.
- I encourage you to ask questions in class. You are supposed to ask questions. Don't guess, ask a question!
- The code/homework you submit must be written completely by you. You can use anything from the textbook/notes with a clear understanding.

What Is An Operating System? I

- An operating system (OS) acts as an intermediary between the user of a computer and the computer hardware.
- The purpose of an OS is to provide an environment in which a user can execute programs in a **convenient** and **efficient** manner.
- An OS is software that manages the computer hardware.
  - *Mainframe OSs* are designed primarily to optimize utilization of hardware.
  - *Personal computer (PC) OSs* support complex games, business applications, and everything in between.
  - *OSs for handheld computers* are designed to provide an environment in which a user can easily interface with the computer to execute programs.
- Thus, some OSs are designed to be convenient, others to be efficient, and others some combination of the two.

What Is An Operating System? II

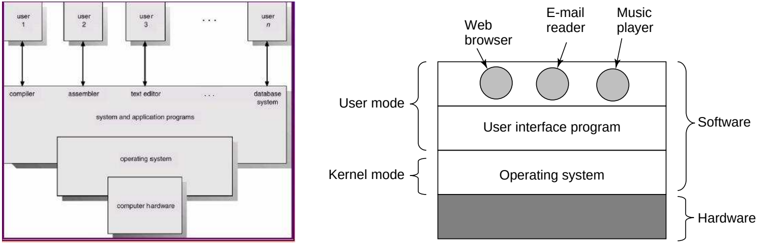


Figure: Abstract view. Where the OS fits in.

What Is An Operating System? III

- A computer system can be divided roughly into four components: *the hardware*, *the OS*, the *application programs*, and *the users* (see Fig. 1).
  1. *Hardware*
     - Electronic, mechanical, optical devices. The central processing unit (CPU), the memory, and the input/output (I/O) devices-provides the basic computing resources for the system.
     - The hardware must provide appropriate mechanisms.
  2. *Software*
     - Programs. Software can be grouped into the following categories:
     - *systems software* (OS & utilities).
     - *applications software* (user programs).
- An OS 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.

User View - The OS as an Extended Machine

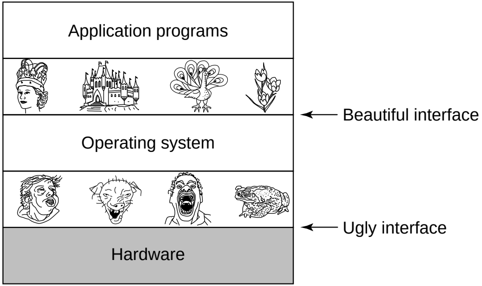


Figure: Operating systems turn ugly hardware into beautiful abstractions.

- provides an abstraction layer over the concrete hardware,
- use the computer hardware in an efficient manner (converting hardware into useful form),
- “hide” the complexity of the underlying hardware. See Fig. 2

System View - The OS as a Resource Manager

- From the computer's point of view, the OS is the program most intimately involved with the hardware. In this context, we can view an OS as a <u>resource allocator</u>.
- Resource - "Something valuable" e.g. CPU time, memory space (RAM), file-storage space, I/O devices (disk), and so on.
- The OS acts as the manager of these resources. Includes multiplexing (sharing) resources in two different ways. Each program gets
  - **time** with the resource
  - **space** on the resource
- Facing numerous and possibly conflicting requests for resources, the OS must decide
  - how to allocate them to specific programs (processes, jobs)
  - how to protect applications from one another,
  - how to provide fair and efficient access to resources,
  - how to operate and control the various I/O devices.

Defining OS and Functionalities

- The fundamental goal of computer systems is to execute user programs and to make solving user problems easier.
- The common functions of controlling and allocating *resources* are then brought together into one piece of software: the OS.
- In addition, we have no universally accepted definition of what is part of the OS.
- A more common definition is that the OS is the one program running at all times on the computer (usually called the kernel), with all else being systems programs and application programs.
- What mechanisms? What policies?
- *Challenges:* Desired functionalities of OS depend on outside factors like users' & application's "Expectations" and "Technology changes" in Computer Architecture (hardware). *OS must adapt:*
  - change abstractions provided to users,
  - change algorithms to change these abstractions,
  - change low-level implementation to deal with hardware.

Computer-System Operation I

- A modern general-purpose computer system consists of one or more CPUs and a number of device controllers connected through a common bus that provides access to shared memory (see Fig. 3).
- The CPU and the device controllers can execute concurrently, *competing for memory cycles*.
- To ensure orderly access to the shared memory, a memory controller is provided whose function is to synchronize access to the memory.

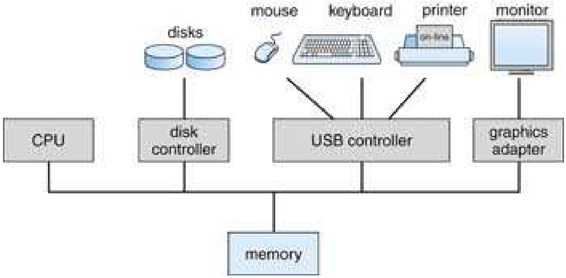


Figure: A modern computer system.

Computer-System Operation I

- For a computer to start running, when it is powered up or rebooted-it needs to have an initial program (bootstrap program) to run.
- Typically, it is stored in read-only memory (ROM) or electrically erasable programmable read-only memory (EEPROM), known by the general term firmware, within the computer hardware.
- It initializes all aspects of the system, from CPU registers to device controllers to memory contents.
- The bootstrap program must know how to load the OS and to start executing that system. To accomplish this goal, the bootstrap program must <u>locate</u> and <u>load</u> into memory the OS kernel.

Computer-System Operation II

UNIX System initialization and Bootstrapping;

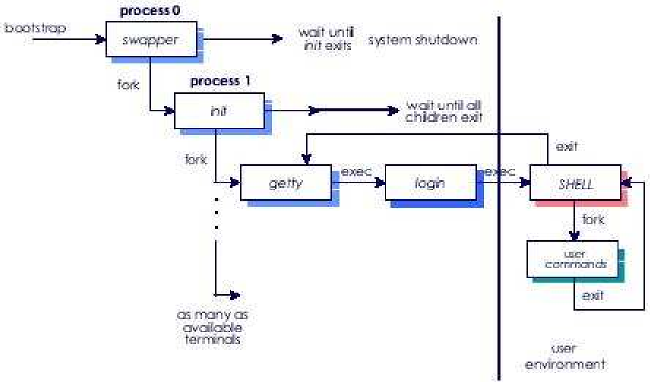


Figure: UNIX System initialization

Computer-System Operation III

- Once the kernel boots, we have a running Linux system. It isn't very usable, since the kernel doesn't allow direct interactions with "user space".
- So, the system runs one program: **init** and waits for some event to occur. This program is responsible for everything else and is regarded as the father of all processes.
- The kernel then retires to its rightful position as system manager handling "kernel space" (see Fig. 4).
- Some portions of the OS remain in main memory to provide services for critical operations, such as dispatching, interrupt handling, or managing (critical) resources.
- These portions of the OS are collectively called the *kernel*.

Kernel = OS - transient components
remains *comes and goes*

Computer-System Operation IV

- The occurrence of an event is usually signaled by an interrupt from either the hardware or the software.
  - Hardware may trigger an interrupt at any time by sending a signal to the CPU, usually by way of the system bus.
  - Software may trigger an interrupt by executing a special operation called a **system call** (also called a monitor call).
- The interrupt must transfer control to the appropriate interrupt service routine (ISR) (see Fig. 5).

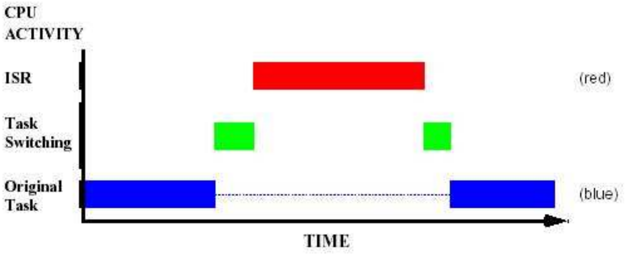


Figure: Interrupt

Computer-System Operation V

The interrupt service routine executes; on completion, the CPU resumes the interrupted computation. A time line of this operation is shown in Fig. 6.

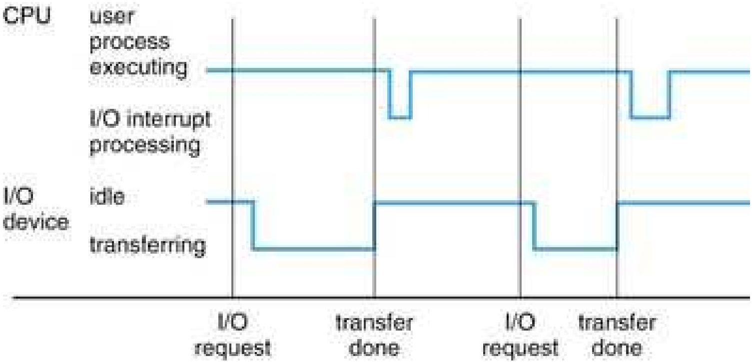


Figure: Interrupt time line for a single process doing output.

Storage Structure I

- Computer programs must be in main memory.
- Interaction is achieved through a sequence of load or store instructions to specific memory addresses.
- The load instruction moves a word (collection of bytes, each word has its own address) from main memory to an internal register within the CPU, whereas the store instruction moves the content of a register to main memory.
- A typical *instruction-execution cycle*, as executed on a system with a *von Neumann architecture*,
    - First fetches an instruction from memory and stores that instruction in the instruction register.
    - The instruction is then decoded and may cause operands to be fetched from memory and stored in some internal register.
    - After the instruction on the operands has been executed, the result may be stored back in memory.
- Notice that the memory unit sees only a stream of memory addresses; it does not know how they are generated (by the instruction counter, indexing, indirection, literal addresses, or some other.

Storage Structure II

- Program counter (PC) holds address of the instruction to be fetched next,
- The processor fetches the instruction from memory,
- Program counter is incremented after each fetch,
- Overlapped on modern architectures (pipelining).
- *Fetch-execute cycle* (see Fig. 7)

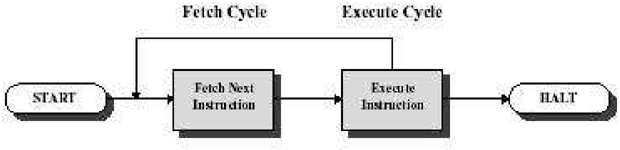


Figure: Fetch and Execute Cycle

Storage Structure III

- Ideally, we want the programs and data to reside in main memory permanently. This arrangement usually is not possible for the following two reasons:
  1. Main memory is usually too small to store all needed programs and. data permanently.
  2. Main memory is a volatile storage device that loses its contents when power is turned off or otherwise lost.
- Thus, most computer systems provide secondary storage as an *extension* of main memory. The most common secondary-storage device is a magnetic disk.
- Many programs then use the disk as both a source and a destination of the information for their processing. Hence, the proper management of disk storage is of *central importance* to a computer system.

Storage Structure IV

The main differences among the various storage systems lie in speed, cost, size, and volatility. The wide variety of storage systems in a computer system can be organized in a hierarchy (See Fig. 8) according to speed and cost.

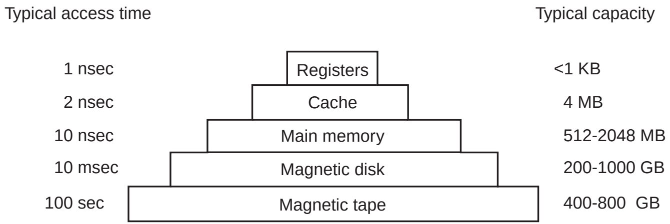


Figure: A typical memory hierarchy. The numbers are very rough approximations.

Storage Structure V

- Memory system must use only as much expensive memory as necessary while providing as much inexpensive, nonvolatile memory as possible.
- Caches can be installed to improve performance where a large access-time or transfer-rate disparity exists between two components. Cache memory (see Fig. 10left);

- Contain a small amount of very fast storage which holds a subset of the data held in the main memory.
- Processor first checks cache. If not found in cache, the block of memory containing the needed information is moved to the cache replacing some other data.

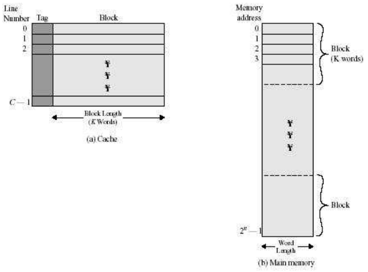
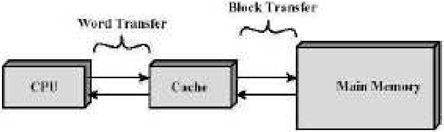


Figure: Cache and Main Memory

Storage Structure VI



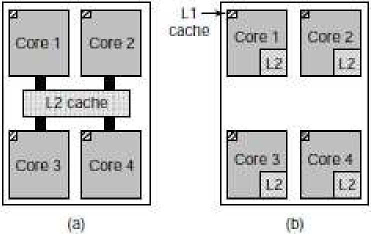


Figure: Upper: Cache Memory. Lower: (a) A quad-core chip with a shared L2 cache. (b) A quad-core chip with separate L2 caches.

Storage Structure VII

Future storage technology includes 3-dimensional crystal structures which allow optical access to a dense 3-dimensional storage facility (see Fig. 11).

http://www.voyle.net/Guest Writers/Michael E. Thomas/Atomic_press.htm

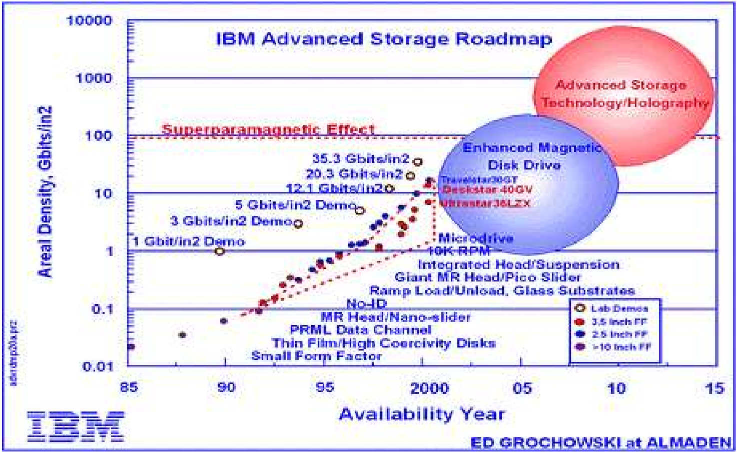
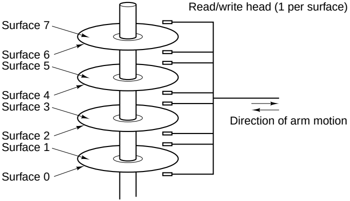


Figure: IBM Advanced Storage Roadmap.

I/O Structure I

- Storage is only one of many types of I/O devices within a computer.
- A large portion of OS code is dedicated to managing I/O, both because of its importance to the reliability and performance of a system and because of the varying nature of the devices.
- The magnetic discs in the drive are rotating and magnetic heads move in and out in order to access any part of the surface area on the disc that holds data.
- This means access usually involves a disc rotation delay and also a head positioning delay (see Fig. 12left).



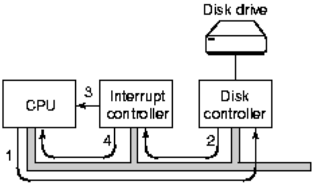


Figure: Left: Structure of a disk drive. Right: The steps in starting an I/O device.

I/O Structure II

- A general-purpose computer system consists of CPUs and multiple device controllers that are connected through a common bus (see Fig. 13).
- Each device controller is responsible for moving the data between the peripheral devices that it controls and its local buffer storage (see Fig. 12right).
- Typically, OSs have a device driver for each device controller. Software that communicates with controller is called device driver. To start an I/O operation;
  - The device driver loads the appropriate registers within the device controller.
  - The device controller, in turn, examines the contents of these registers to determine what action to take (such as “read a character from the keyboard”).
  - The controller starts the transfer of data from the device to its local buffer.
  - Once the transfer of data is complete, the device controller informs the device driver via an interrupt that it has finished its operation.
  - The device driver then returns control to the OS, possibly returning the data or a pointer to the data if the operation was a read.

I/O Structure III

A system bus would involve a pathway along which data could travel (usually 32-bits side-by-side i.e. in bit-wise parallel).

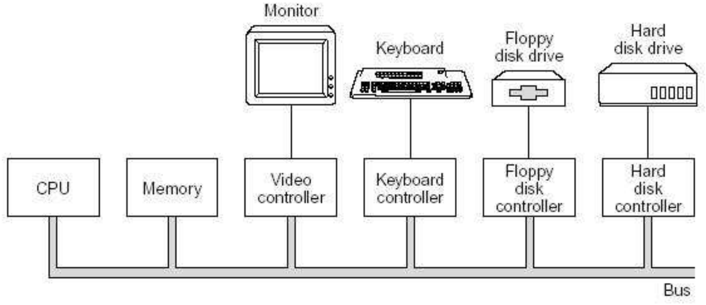


Figure: Top-level Components.

I/O Structure IV

- Interrupts normal sequence of execution. I/O requests can be handled *synchronously* or *asynchronously*.
  - *In a synchronous system,* a program makes the appropriate OS call, as the CPU is now executing OS code, the original program's execution is halted i.e. it waits.
  - *In an asynchronous system,* a program makes its request via the OS call, then its execution resumes, it will most likely not have had its request serviced yet!
- This form of <u>interrupt-driven I/O</u> is fine for moving small amounts of data but can produce high overhead when used for bulk data movement such as disk I/O.
- To solve this problem, direct memory access (DMA) is used. After setting up buffers, pointers, and counters for the I/O device, the device controller transfers an entire block of data directly to or from its own buffer storage to memory, with <u>no intervention</u> by the CPU.