CSE 3320

Chapter 1: Getting Started

Trevor Bakker

The University of Texas at Arlington

Agenda

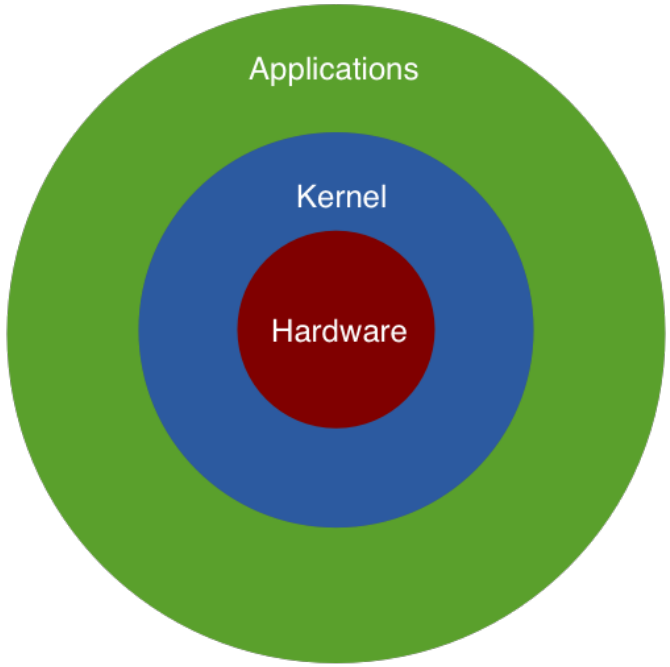
- Define several operating systems related terms and develop a common terminology
- Discuss hardware abstraction
- Discuss BIOS

What is an Operating System?

A collection of one or more software modules that manages and controls the resources of a computer or other computing device or electronic device and gives users and programs an interface to utilize these resources

Terminology

Kernel - The part of the OS that implements basic functionality and is always resident in memory.



Terminology

Services - functions the OS kernel provides to users, mostly through APIs via OS calls. These can be grouped into categories by functionality, such as file manipulation services (create, read, copy), memory allocation (get, free) or miscellaneous services (get system time)

Terminology

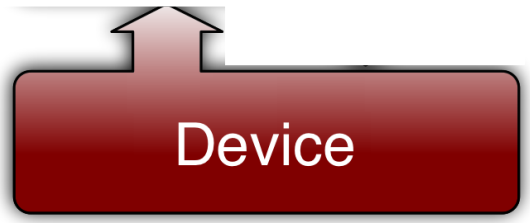
Utility - Programs no part of the OS kernel, but work closely with the kernel to provide was of use or access to system information.

Example: A shell such as bash, csh, or ksh provides a user interface to system services and can call other utilities such as ls.

Terminology

Device - A piece of hardware connected to the main computer system hardware.

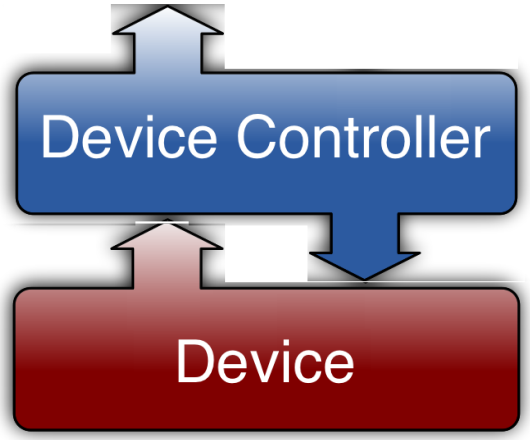
Example: hard drives, video card, mouse



Terminology

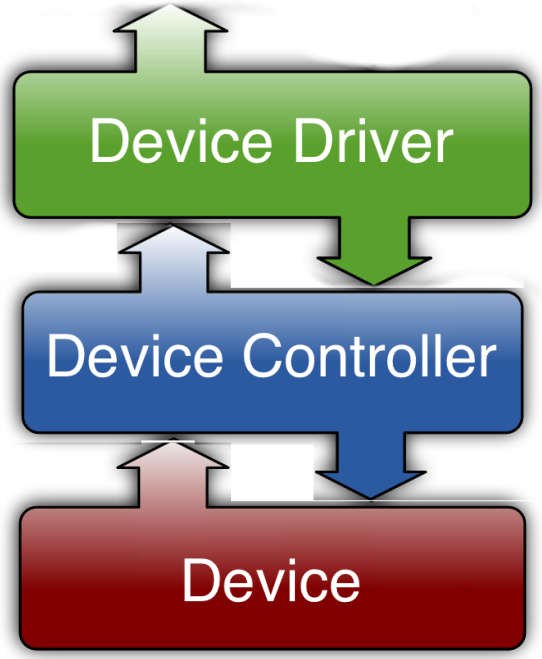
Device Controller - Hardware interface which helps connect a device or a group of similar devices to a computer system

Example: disk controller, USB controller



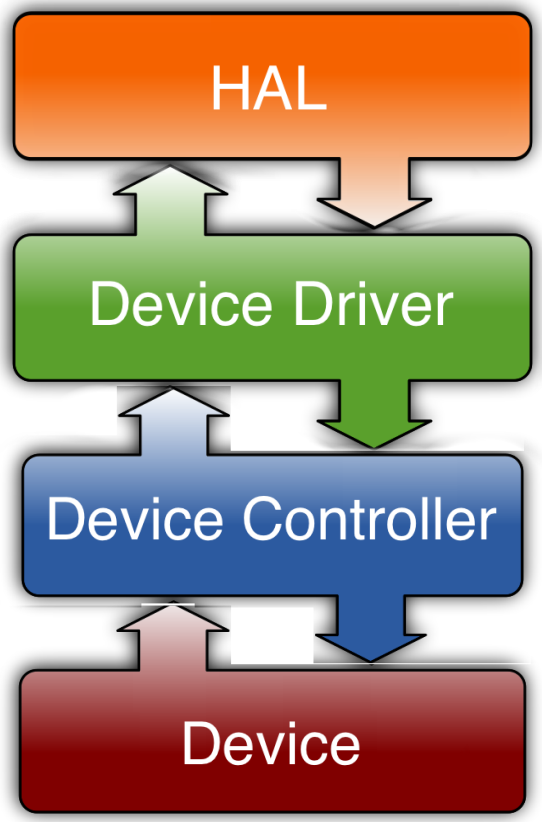
Terminology

Device Driver - A software routine that is part of the OS, and is used to communicate with and control a device through its device controller.



Terminology

Hardware Abstraction Layer - are software routines or modules that provide a device independent layer through which to communicate with hardware.



Terminology

BIOS (Basic Input/Output System) -Software that abstracts the common device hardware, such as keyboards, basic video, and system clock.

Image © Timothy Vollmer licensed under CC BY-NC-SA 2.0c

Why BIOS and abstraction layers?

- Abstracts the hardware
- The OS does can deal with all common devices the same.
  - Don’t need to rev your OS to handle a 112 key keyboard instead of an 88 key keyboard

BIOS History

- Early days of personal computers were DIY
  - Assembling hardware and programming simple programs were the norm (Assembly Language)

Altair image © Michael Holley and placed in Public Domain

Assembly Language

- Tedious
- Large complex programs are difficult and time-consuming to develop
- Users began to demand more functionality in programs

```
SUBTTL DSKREAD -- PHYSICAL DISK READ
PAGE
        procedure   DskRead,NEAR
ASSUME  DS:NOTHING,ES:NOTHING

; Inputs:
;       DS:BX = Transfer addr
;       CX = Number of sectors
;       DX = Absolute record number
;       ES:BP = Base of drive parameters
; Function:
;       Call BIOS to perform disk read
; Outputs:
;       DI = CX on entry
;       CX = Number of sectors unsuccessfully transfered
;       AX = Status word as returned by BIOS (error code in AL if error)
;       Zero set if OK (from BIOS)
;       Zero clear if error
; SI Destroyed, others preserved

        PUSH    CX
        MOV     AH,ES:[BP.dpb_media]
        MOV     AL,ES:[BP.dpb_UNIT]
        PUSH    BX
        PUSH    ES
        invoke  SETREAD
        JMP     DODSKOP

SUBTTL DWRITE -- SEE ABOUT WRITING
PAGE
        entry   DWRITE
ASSUME  DS:NOTHING,ES:NOTHING
```

Wild west of PC hardware

- 1970's and 1980's saw the commoditization of PCs
- By 1981 there were 200,000 microcomputers running CP/M, in more than 3000 different hardware configurations
- Operating systems need a better way to interface to a wide variety of hardware

Needed a better way

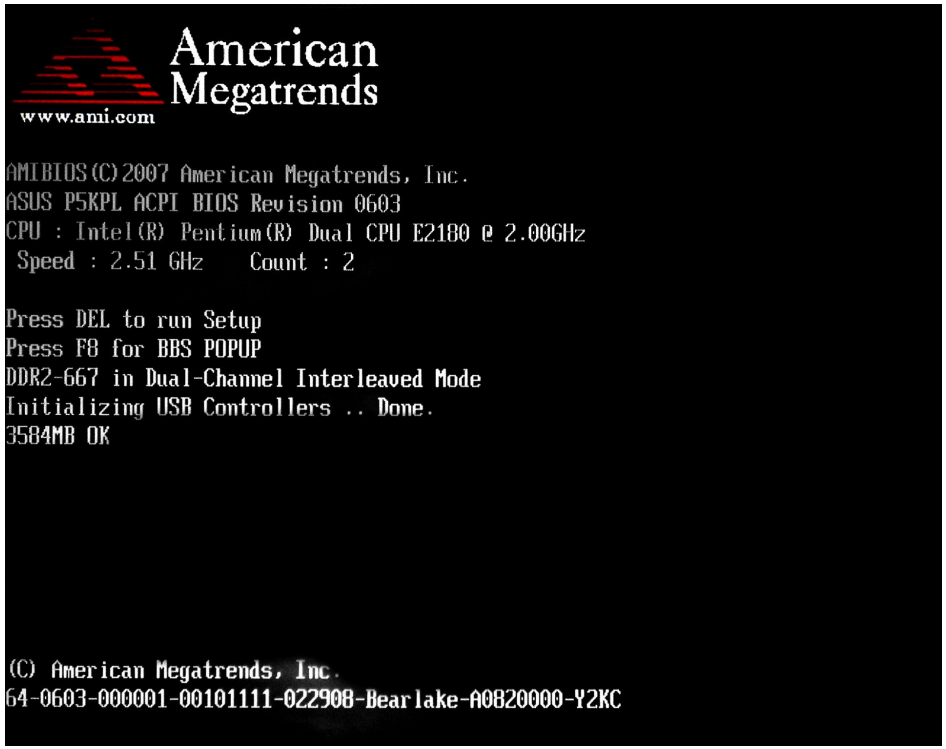
- Needed to be able to add new devices without writing directly to each new piece of hardware.
- Writing assembly to address hardware is fun but it's no way to build a complex and widely used operating system.
- Programmers needed to be able to write programs that could work on different systems with little changes.

BIOS developed

- BIOS (Basic Input/Output System) was invented by Gary Kildall
- First appeared in the CP/M operating system in 1975
- Stored on ROM on the motherboard
  - Replaced later with firmware BIOS that can be flashed and updated without removing the chip.

BIOS Purpose

- Initialize and test system components
  - POST (Power On Self Test)
- Load the boot loader or operating system
- Provide an abstraction layer for the hardware



How does BIOS boot the OS

1. Power is applied to the PC
    1. The power supply generates a power good signal which is received by the motherboard timer.
    2. The motherboard timer stops sending a reset signal to the CPU
2. The CPU begins processing instructions
    1. The first instruction read is from FFFF:0000h
    2. The instruction is a JMP opcode telling the CPU where to find the BIOS

How BIOS boots the OS

3. POST (Power On Self Test)

    1. BIOS starts with tests of the hardware on the motherboard (CPU, memory, IRQ controllers)

    2. BIOS searches memory between C000:000h and C780:000h for video BIOS, runs a checksum, and initializes the video adapter

    3. BIOS searches C800:000h to DF800:000h in 2KB increments for other adapter ROMs such as network adapters and verifies them

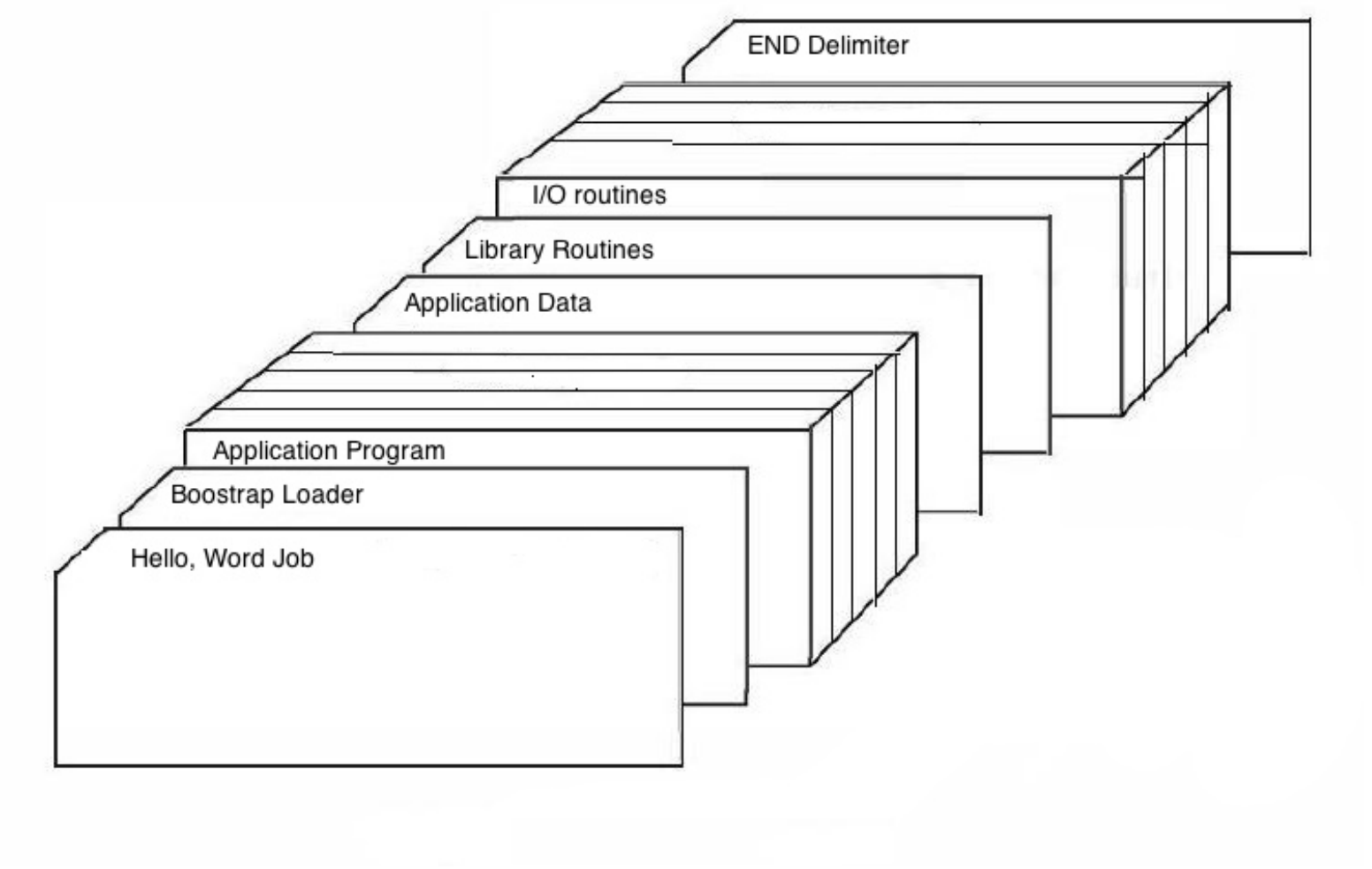
Booting the OS

4. BIOS searches the computers hard drive for the master partition boot sector and loads it into memory at 0000:7C00h
    1. Usually sector 1, head 0, cylinder 0
5. BIOS looks for the Volume Boot Sector, loads it into memory and begins to load the OS.

Origins of the OS

- Early computers were single-user
  - Only one user and one program could run at a time.
- Programs we stored on paper tape or punch card.
- Each program had not only application code but also a boot loader and all library routines

Stacking the Deck



Loading the Application

- Loader - Routine contained on the first card which would load the rest of the application into memory
  - Loaders began to grow in sophistication and could load in compiled programs
- Beginnings of early operating systems called monitors.