Operating Systems and Hardware Overview

Portions of this material from U. Manitoba

Components of a Computer System
- A computer system consists of both hardware and software
- Many different types:
  - Tablets, laptops
  - Desktops, workstations
  - Embedded systems
  - Servers
  - Mainframes
  - Super Computers

Inside a Computer Case

Hardware Components
- Processor
- Main memory
- Secondary memory
  - hard disk, optical disks, tapes, ...
- Input devices
  - keyboard, mouse, microphone, ...
- Output devices
  - monitor, printer, speaker, ...

Processors

Main Memory
Motherboard of a Laptop

Motherboard of iPhone 3G

Buses
- Key abstraction for hardware
- A hardware-level messaging system for components
- A lot of EE-type specifications about clock rate, voltage, etc.
- You mostly need to know that the specs for all components have to agree

Bus Intuition
- For a given device type, the bus client speaks the protocol over the wire
- Encapsulates low-level device implementation

A logical view of hardware
- CPU(s)
- RAM
- PCI-X Bus
- North Bridge (Fast devices: e.g., GPU)
- BIOS
- SATA
- PCI Bus
- South Bridge ("Slow" Devices: e.g., Disk, USB, Most network)
Buses galore

- Memory Bus: connects Northbridge components (CPU, RAM, GPU, Southbridge controller)
- Disk buses:
  - Controller on the motherboard
  - Often on south bridge
  - Speaks to disks
  - SATA, IDE, SCSI, etc
- USB, Firewire, etc. are all bus protocols

Fewer Bridges

- Newer system organizations are moving more devices to the North bridge, and consolidating more things on the CPU itself.

So what is an OS?

One view of an OS

Another simple view of an OS

A less happy view of an OS
So which one is right?
• They all are

An OS serves three masters
1. Give users a desktop environment
2. Give applications a more usable abstraction of the hardware
3. Give hardware manufacturers an abstraction of the applications

Background (1)
• CPUs have 2 modes: user and supervisor
  – Sometimes more, but whatevs
• Supervisor mode:
  – Issue commands to hardware devices
  – Power off, Reboot, Suspend
  – Launch missiles, Do awesome stuff
• User mode:
  – Run other code, hardware tattles if you try anything reserved for the supervisor

Master #2: Applications
• Application Programming Interface (API)
  – Win32 (Windows)
  – POSIX (Unix/Linux)
  – Cocoa/Cocoa Touch (Mac OS/iOS)
• Application-facing functions provided by libraries
  – Injected by the OS into each application
OS architecture

Famous libraries, anyone?
- Windows: ntdll.dll, kernel32.dll, user32.dll, gdi32.dll
- Linux/Unix: libc.so, ld.so, libpthread.so, libm.so

Caveat 1
- Libraries include a lot of code for common functions
  - Why bother reimplementing sqrt?
- They also give high-level abstractions of hardware
  - Files, printer, dancing Homer Simpson USB doll
- How does this work?

System Call
- Special instruction to switch from user to supervisor mode
- Transfers CPU control to the kernel
  - One of a small-ish number of well-defined functions
- How many system calls does Windows or Linux have?
  - Windows ~1200
  - Linux ~350
Caveat 2

- Some libraries also call special apps provided by the OS, called a daemon (or service)
  - Communicate through kernel-provided API
- Example: Print spooler
  - App sends pdf to spooler
  - Spooler checks quotas, etc.
  - Turns pdf into printer-specific format
  - Sends reformatted document to device via OS kernel

OS architecture

Master 3: Hardware

- OS kernels are programmed at a higher low level of abstraction
  - Disk blocks vs. specific types of disks
- For most types of hardware, the kernel has a "lowest common denominator" interface
  - E.g., Disks, video cards, network cards, keyboard
  - Think Java abstract class
  - Sometimes called a hardware abstraction layer (HAL)
- Each specific device (Nvidia GeForce 600) needs to implement the abstract class
  - Each implementation is called a device driver

What about Master 1

- What is the desktop?
- Really just a special daemon that interacts closely with keyboard, mouse, and display drivers
  - Launches programs when you double click, etc.
  - Some program libraries call desktop daemon to render content, etc.

An OS serves three masters

1. Give users a desktop environment
   - Desktop, or window manager, or GUI
2. Give applications a more usable abstraction of the hardware
   - Libraries (+ system calls and daemons)
3. Give hardware manufacturers an abstraction of the applications
   - Device Driver API (or HAL)
So what is Linux?
• Really just an OS kernel
  – Including lots of device drivers
• Conflated with environment consisting of:
  – Linux kernel
  – Gnu libc
  – X window manager daemon
  – CUPS printer manager
  – Etc.

So what is Ubuntu? Centos?
• A distribution: bundles all of that stuff together
  – Pick versions that are tested to work together
  – Usually also includes a software update system

OSX vs iOS?
• Same basic kernel (a few different compile options)
• Different window manager and libraries

What is Unix?
• A very old OS (1970s), innovative, still in use
• Innovations:
  – Kernel written in C (first one not in assembly)
  – Co-designed C language with Unix
  – Several nice API abstractions
  – Fork, pipes, everything a file
• Several implementations: *BSDs, Solaris, etc.
  – Linux is a Unix-like kernel

What is POSIX?
• A standard for Unix compatibility
• Even Windows is POSIX compliant!

RTFM
• man pages are your friend!
  – (not a dating service)
  – Ex: ‘man malloc’, or ‘man 3 printf’
    • Section 3 is usually where libraries live - there is a command-line utility printf as well
• Use ‘apropos term’ to search for man entries about term
• Windows has MSDN online, which is good
Summary

• Understand basic hardware terminology
• Understand what an OS is
  – Three masters
  – Nomenclature
• Questions?