Computer Science 162: Operating Systems and Systems Programming

Scribe: Tyler Nguyen

Lecture 1: August 23, 2017

1 What is an operating system?

- Special layer of software that provides application software access to hardware resources
  - Convenient abstraction of complex hardware devices
  - Protected access to shared resources
  - Security and authentication
  - Communication amongst logical entities

1.1 What Does an OS do?

- Provide abstractions to apps
  - File systems
  - Processes, threads
  - VM, containers
  - Naming system
  - ...
- Manage resouces:
  - Memory, CPU, storage, ...

1.2 OS Basics: “Virtual Machine” Boundary

- Software
- OS Hardware Virtualiztion
  - Threads
  - Processes
  - Address Spaces

  - Files
  - Windows
  - Sockets

- Hardware

  - Processor
  - Memory
  - Storage
  - Networks
  - Inputs
  - Displays

2 What makes Operating Systems exciting and challenging

2.1 Technology Trends: Moore's Law

Moore's law: Gordon Moore (co-founder of Intel) predicted in 1965 that the transistor density of semiconductor chips would double roughly every 18 months. Microprocessors have become smaller, denser, and more powerful.

2.2 New Challenge: Slowdown in Joy's law of Performance

2.3 Another Challenge: Power Density

Super-exponential growth in power density.

- Moore's law extrapolation

  - Potential power density reaching amazing levels!

- Flip side: battery life more important

3 People-to-Computer Ratio Over Time

- Today: multiple CPUs per person!

  - Approaching hundreds?

3.1 The End of Moore's Law...

- Moore's Law has (officially) ended – Feb 2016

  - No longer getting 2× transistors per chip every 18 months...
  - or even every 24 months

- May have only 2-3 smallest geometry fabrication plants left:
  - Intel and Samsung and/or TSMC
- Vendors moving to 3D stacked chips
  - More layers in older geometries

3.2 Storage Capacity

Increasing exponentially

3.3 Network Capacity

Increasing exponentially

3.4 Challenge: Complexity

- Applications consisting of...
  - ...a variety of software modules that...
  - ...run on a variety of devices (machines) that
    - ...implement different hardware architectures
    - ...run competing applications
    - ...fail in unexpected ways
    - ...can be under a variety of attacks
- Not feasible to test software for all possible environments and combinations of components and devices
  - The question is not whether there are bugs, but how serious they are!

3.5 How do we tame complexity?

- Every piece of computer hardware different
  - Different CPU
    - Pentium, PowerPC, ColdFire, ARM, MIPS
  - Different amounds of memory, disk, ...
  - Different types of peripherals
    - Mice, keyboards, sensors, cameras, fingerprint readers
  - Different networking environment
    - Cable, DSL, Wireless, Firewalls

3.6 OS Tool: Virtual Machine Abstraction

3.6.1 Virtual Machines

- Software emulation of an abstract machine
  - Give programs illusion they own the machine
  - Make it look like hardware has features you want
- Two types of "Virtual Machines"
  - Process VM: supports execution of a single program; this functionality is typically provided by OS
  - System VM

3.6.2 Process VMs

- Programming simplicity
  - Each process thinks it has all memory/CPU time
  - Each process thinks it owns all devices
  - Different devices appear to have same high level interface
  - Device interfaces more powerful than raw hardware
    - Bitmapped display →windowing system
    - Ethernet card →reliable, ordered, networking (TCP/IP)
- Fault isolation

3.6.3 System Virtual Machines: Layers of OSs

- Useful for OS development
  - WHen an OS crashes, restricted to one VM
  - Can aid testing programs on other OSs