COS 140: Foundations of Computer Science

Operating Systems

Fall 2018

Announcements

- Reading: None for today!
- Homework: none

Operating systems

- When you deal with a computer, you're really dealing with an *operating system*.
- Examples:
  - Windows (95/98, NT, 2000, XP, 7, 8, 10)
  - Macintosh OS X
  - UNIX – including Linux
  - iOS, Android, Palm OS, WebOS

Why have an operating system?

- ☐ They provide services that all programs need
- ☐ They allow multiple programs to run at once
- ☐ They are the computers' housekeepers
- ☐ They facilitate portability of programs

Ways of viewing an OS

- ☐ Service provider
- ☐ Resource manager
- ☐ Virtual machine

Service provider

- Most programs need (e.g.):
  - input and output
  - access to memory
  - ways to handle errors
  - timers and alarms
- Could have each program do this itself
- Operating system centralizes these (and other) services
  - Programmers don't have to program them again
  - Programs don't have to include all the code to do these things

Resource manager

- What *resources* does a computer have?
  - CPU, memory
  - disk (files, etc.)
  - I/O devices
- CPU must be managed to allow *multiprogramming* (*multitasking*)
- Why must other resources by managed:
  - ...in a single-user, no multitasking environment?
  - ...with multitasking?
  - ...in a multi-user, multitasking environment?

Virtual machine

- To user, "computer" $\equiv$ "operating system": a *virtual* machine
- OS (and utilities) $\Rightarrow$ environment for user
- Also for programs:
  - High-level, abstract view of hardware in terms of services OS provides: an *application programmer interface* (API)
  - Increases ease of programming
  - Increases portability – e.g., POSIX

Why Study OS in CS?

- Why not just a technical school topic?
- Active area of research and development
- Much underlying theory, many fundamental concepts need to be mastered by OS designers

Kernel

Kernel

- The heart of the OS is the *kernel*
- Kernel does:
  - scheduling
  - resource management
- Other things often considered parts of the OS:
  - shells
  - libraries
  - utilities
- Kernel vs user mode

Processes

Processes

- *Process* $\equiv$ program $+$ its state of execution
- Almost all modern OS: multiprogramming, pseudo-parallelism
- System calls:
  - interface between processes and kernel
  - system calls $+$ instructions $=$ program's (process') API for computer

Process state

- Why save a process' state?
  - So it can be interrupted, later resumed
- What is needed in the state?
  - program counter, stack pointer, CPU flags
  - where it is in memory, other memory-related information
  - bookkeeping information
- State of processes stored in *process table*
- *Process control block (PCB)*

Process synchronization

- Processes often share memory, or access to devices
- Race conditions
- Critical regions
- Need synchronization primitives, mechanisms

Process scheduling

- If only 1 CPU, only one process can run at once
- When should a process give up the CPU?
- *Scheduler* decides which one runs
- Simple scheduler: round-robin

Virtual Memory 15 / 18

Virtual memory

- Size of memory limits number of processes that can run at once
- Also limits size of any given process
- Computers almost always can address more memory than they have
  - 32-bit computer – how much memory could it address?
  - 64-bit computer?
- Wouldn't it be nice to allow full *address space* to be used?
- Also nice if each process could think it had its own address space, starting at address 0

Virtual memory

- ☐ Virtual memory gives each process the illusion of having its own, complete address space
- ☐ VM allows many more processes to run than will fit in memory
- ☐ Trick: use disk space instead of RAM (*swap space*, paging area)
- ☐ When a part of a process is not in use, *page it out* to disk
- ☐ When needed, page it in

File Sytems 17 / 18

File systems

- ☐ Another resource: disk space
- ☐ Files, directories (folders)
- ☐ Storing files
- ☐ Managing files
- ☐ Recovering from errors

Other stuff

- ☐ Much more to OS than has been said
- ☐ Memory management, swap space management
- ☐ Input/output
- ☐ Network $\leftrightarrow$ OS issues
- ☐ Handling deadlocks
- ☐ Computer security
- ☐ Distributed operating systems