COS 318: Operating Systems

Introduction

Jaswinder Pal Singh and a Fabulous Course Staff
Computer Science Department
Princeton University

http://www.cs.princeton.edu/courses/archive/fall16/cos318/

Today

- Course information and logistics
- What is an operating system?
- Evolution of operating systems
- Why study operating systems?

Information and Staff

- Website
  - http://www.cs.princeton.edu/courses/archive/fall18/cos318/
- Textbooks
  - Modern Operating Systems, 4th Edition, Tanenbaum and Bos
- Instructors
  - Jaswinder Pal Singh, Office: 423 CS, Hours: Mon 1:30 – 3 pm
- Teaching assistants (offices and hours to be posted on web site)
  - Lance Goodridge
  - James Heppenstall
- Undergraduate Assistants (to be finalized)

Grading

- Projects 70%
- Exam 20%
- Class Participation 10%

- Exam will be in-class, likely sometime after midterm week (watch for announcements)

Projects

- Build a small but real OS kernel, bootable on real PCs
- A lot of hacking (in C & x86 assembly) but very rewarding
- Projects
  - Bootloader (150-300 lines)
  - Non-preemptive kernel (200-250 lines)
  - Preemptive kernel (100-150 lines)
  - Inter-process communication and device driver (300-350 lines)
  - Virtual memory (300-450 lines)
  - File system (500+ lines)

5

Projects

- How
  - Pair with a partner for project 1, 2 and 3
  - Pair with a different partner for project 4 and 5
  - Do the final project yourself (no partners)
  - Design review at the end of week one
  - All projects due Sundays at 11:55 pm
- Where to do the projects
  - Develop on courselab machines, via remote login
  - Instructions on how to develop and submit will be on the assignment web pages

6

Project Grading

- Design Review
  - Requirements will be specified for each project
  - Sign up online for making appointments for design review etc
  - 0-5 points for each design review
  - 10% deduction for missing an appointment
- Project completion
  - Assigned project points plus possible extra points
- Late policy for grading projects
  - 1 hour: 98.6%, 6 hours: 92%, 1 day: 71.7%
  - 3 days: 36.8%, 7 days: 9.7%

7

Logistics

- Precepts
  - Time: Mon 7:30pm – 8:20pm in CS 105 (unless changed)
  - No second session
- For project 1
  - A tutorial on assembly programming and kernel debugging
    - Mon 9/17: 7:30-8:20pm in CS 105 (unless changed)
  - Precept
    - 9/24: 7:30-8:20pm in CS 105 (unless changed)
  - Design review
    - 9/24 (Monday) 1:30pm – evening (Friend 010, unless changed)
    - Sign up online (1 slot per team)
  - Due: 9/30 (Sunday) 11:55pm

8

Use Piazza for Discussions

- Piazza is convenient
  - Most of you love it (?)
- Search, ask and answer questions
  - Students are encouraged to answer questions on Piazza
  - Staff will try to answer in a timely manner
- Only use email if your question is personal/private
  - For questions about your specific project grade: send email to the TA in charge

9

Ethics and Other Issues

- Follow Honor System
  - Ask teaching staff if you are not sure
  - Asking each other questions is okay: best place is on Piazza
  - **Work must be your own (or your team's)**
- If you discover any solutions online
  - Tell teaching staff
- Do not put your code or design on the web, in social media, or anywhere public or available to others …
- *Most important thing to do in this course:*

  Do not violate the Honor Code

10

COS318 in Systems Course Sequence

- Prerequisites
  - COS 217: Introduction to Programming Systems
  - COS 226: Algorithms and Data Structures
- 300-400 courses in systems
  - **COS318: Operating Systems**
  - COS320: Compiler Techniques
  - COS333: Advanced Programming Techniques
  - COS432: Information Security
  - COS475: Computer Architecture
- Courses needing COS318
  - COS 461: Computer Networks
  - COS 518: Advanced Operating Systems
  - COS 561: Advanced Computer Networks

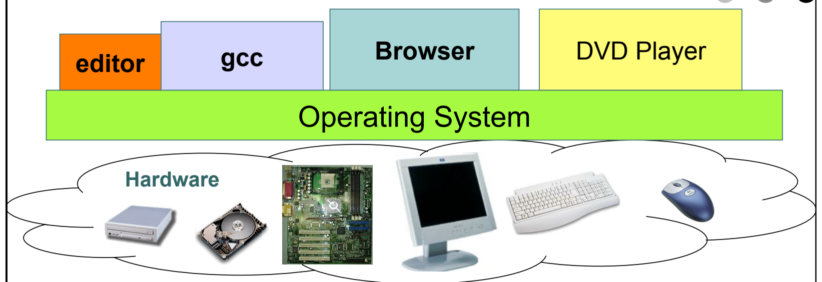
11

Today

- Course information and logistics
- What is an operating system?
- Evolution of operating systems
- Why study operating systems?

12

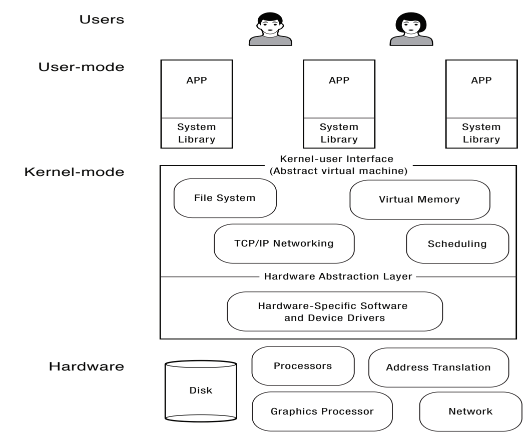
What Is Operating System?



- Software between applications and hardware
- Provide abstractions to layers above
- Implement abstractions for and manage resources below

13

In a Little More Depth: The Software



What Do Operating Systems Do?

- Provides abstractions to user-level software above
  - User programs can deal with simpler, high-level concepts
    - E.g. files instead of disk blocks, virtual memory instead of physical, etc.
  - Hide complex and unreliable hardware, and the variety of hardware
  - Provide illusions like “sole application running” or “infinite memory”
  - For each area, can ask: what is the HW interface, what is the nicer interface the OS provides, what is the even nicer one the library provides?
- Implement the abstractions: manage resources
  - Manage application interaction with hardware resources
  - Provide standard services: program execution, I/O operations, file system manipulation, communication, accounting
  - Allow multiple applications and multiple users to share resources effectively without hurting one another
  - Protect applications from one another and from crashing the system

15

Some Examples

- What if a user tries to access disk blocks directly?
- What if a user program can access all RAM memory?
- What if a user runs the following code:

```
int main() {
        while(1)
                fork();
}
```

- What if many programs are running infinite loops?

```
while (1);
```

16

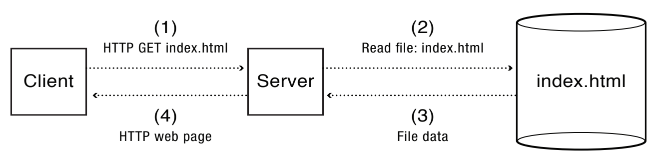
Operating System Roles

- Referee
  - Resource allocation among users, applications
  - Protection/isolation of users, applications from one another
- Illusionist
  - Each application appears to have the entire machine to itself
    - Processor/processors, all of memory (and in fact vastly more than all of physical memory), reliable storage, reliable network transport
- Glue
  - Libraries, user interface widgets, …
  - Communication between users, applications

Example: File Systems

- Referee role
  - Prevent users from accessing others' files without permission
- Illusionist role
  - Files can grow (nearly) arbitrarily large
  - User program doesn't need to know where the file data are or how they are organized or are accessed by the processor
  - Files persist even if machine crashes in the middle of a save
- Glue role
  - Named directories, printf, …

Example: Web Application



- How does the server manage many simultaneous client requests and share CPU and other resources among them?
- How do we keep the client safe from spyware embedded in scripts on a web site?

Example: Reading from Disks

- Different types of disks, with very different structures
  - Floppy, various kinds of hard drives, Flash, IDE, …
- Different hardware mechanisms to read, different layouts of data on disk, different mechanics
- Floppy disk has ~20 commands to interact with it
- Read/write have 13 parameters; controller returns 23 codes
- Motor may be on or off, don't read when motor off, etc.
- And this is only one disk type
- Instead, a simple abstraction: data are in files, you read from and write to files using simple interfaces
- OS manages all the rest

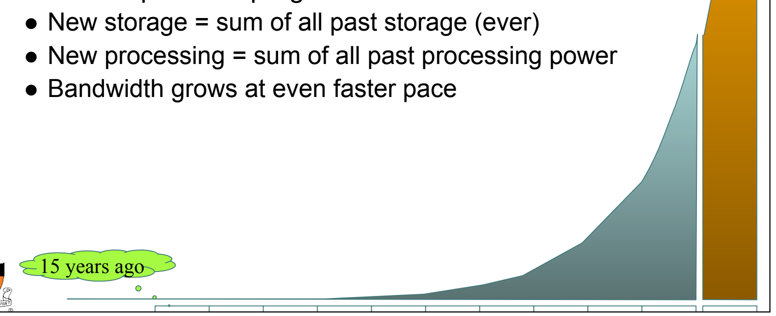
20

Today

- Course information and logistics
- What is an operating system?
- Evolution of operating systems
- Why study operating systems?

Exponential Growth in Computing and Communications (Courtesy Jim Gray)

- #transistors on chip doubles every 18 months
- 100x per decade
- Progress in next 18 months
  = ALL previous progress
  - New storage = sum of all past storage (ever)
  - New processing = sum of all past processing power
  - Bandwidth grows at even faster pace



Personal Computers Then and Now

- Osborne Executive PC (1982) vs Apple iPhone
  - 100x weight, 500x volume, 10x cost (adjusted), 1/100 clock frequency

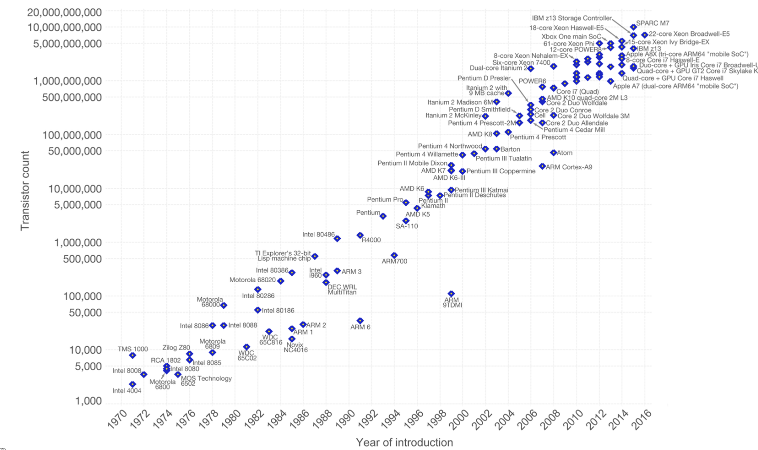
A Typical Academic Computer (1981 vs. 2011)

	1981	2011	Ratio
Intel CPU transistors	0.1M	1.9B	~20000x
Intel CPU core x clock	10Mhz	10×2.4Ghz	~2,400x
DRAM	1MB	64GB	64,000x
Disk	5MB	1TB	200,000x
Network BW	10Mbits/sec	10GBits/sec	1000x
Address bits	32	64	2x
Users/machine	10s	< 1	>10x
$/machine	$30K	$1.5K	1/20x
$/Mhz	$30,000	$1,500/24,000	1/4,800x

Computer performance over time

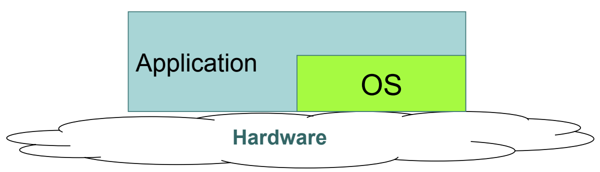
	1981	1997	2014	Factor (2014/1981)
Uniprocessor speed (MIPS)	1	200	2500	2.5K
CPUs per computer	1	1	10+	10+
Processor MIPS/$	$100K	$25	$0.20	500K
DRAM Capacity (MiB)/$	0.002	2	1K	500K
Disk Capacity (GiB)/$	0.003	7	25K	10M
Home Internet	300 bps	256 Kbps	20 Mbps	100K
Machine room network	10 Mbps (shared)	100 Mbps (switched)	10 Gbps (switched)	1000
Ratio of users to computers	100:1	1:1	1:several	100+

Transistor Count on Processor Chips over Time



Phase 1: Hardware Expensive, Human Cheap

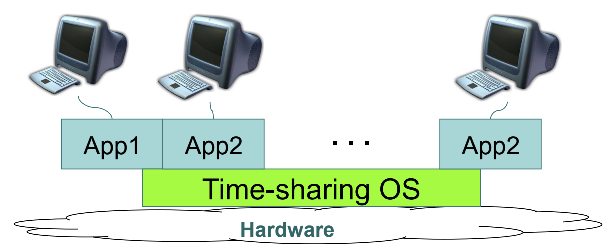
- User at console, OS as subroutine library
- Batch monitor (no protection): load, run, print
- A lot of the (expensive) hardware sits idle a lot. Developments:
  - Interrupts; overlap I/O and CPU
  - Direct Memory Access (DMA)
  - Memory protection: keep bugs to individual programs
  - Multics: designed in 1963 and run in 1969; multiprogramming
- Assumption: No bad people. No bad programs. Minimum interactions



27

Phase 2: Hardware gets Cheaper, Human Expensive

- Use cheap terminals to share a computer
- Time-sharing OS
- Unix enters the mainstream as hardware got cheaper
- Problems: thrashing as the number of users increases



28

Phase 3: HW Cheaper, Human More Expensive

- Personal computer
  - Altos OS, Ethernet, Bitmap display, laser printer (79)
  - Pop-menu window interface, email, publishing SW, spreadsheet, FTP, Telnet
  - Became >200M units per year
- PC operating system
  - Memory protection
  - Multiprogramming
  - Networking

First PC at Xerox PARC

29

Now: > 1 Machines per User

- Pervasive computers
  - Wearable computers
  - Communication devices
  - Entertainment equipment
  - Computerized vehicle
  - Phones: billions units /year
- OS are specialized
  - Embedded OS
  - Specialtly general-purpose OS (e.g. iOS, Android)

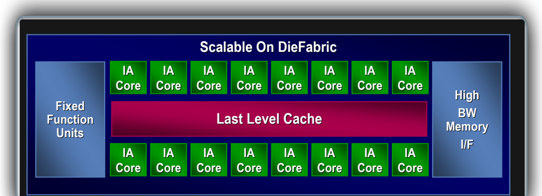
30

Now: Multiple Processors per "Machine"

- Multiprocessors
  - SMP: Symmetric MultiProcessor
  - ccNUMA: Cache-Coherent Non-Uniform Memory Access
  - General-purpose, single-image OS with multiproccesor support
- Multicomputers
  - Supercomputer with many CPUs and high-speed communication
  - Specialized OS with special message-passing support
- Clusters
  - A network of PCs
  - Server OS w/ cluster abstraction (e.g. MapReduce)

Now: Multiple "Cores" per Processor

- Multicore or Manycore transition
  - Intel Xeon processor has 10 cores / 20 threads
  - New Intel Xeon Phi has 72 cores, Core X goes up to 18 cores
  - nVidia GPUs has thousands of FPUs
- Accelerated need for software support
  - OS support for many cores
  - Parallel programming of applications



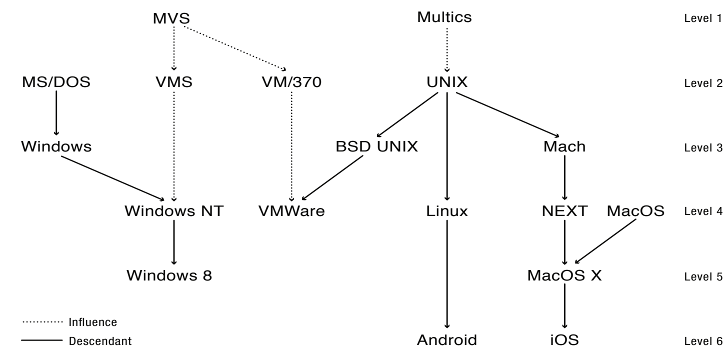
32

Now: Datacenter as A Computer

- Cloud computing
  - Hosting data in the cloud
  - Software as services
  - Examples:
    - Google, Microsoft, Salesforce, VoIP telephony, …
- Utility computing
  - Pay as you go for computing resources
  - Outsourced warehouse-scale hardware and software
  - Examples:
    - Amazon, Google, Micros

33

OS history



Today

- Course information and logistics
- What is an operating system?
- Evolution of operating systems
- Why study operating systems?

35

Why Study OS?

- OS is a key part of a computer system
  - It makes our life better (or worse)
  - It is “magic” to realize what we want
  - It gives us “power” (reduce fear factor)
- Learn how computer systems really work, who does what, how
- Learn key CS concepts: abstraction, layering, virtualization, indirection
- Learn about concurrency
  - Parallel programs run on OS
  - OS runs on parallel hardware
  - Best way to learn concurrent programming
- Understand how a system works
  - How many procedures does a key stroke invoke?
  - What happens when your application references 0 as a pointer?

36

Why Study OS?

- Basic knowledge for many areas
  - Networking, distributed systems, security, …
- Build an OS
  - Real OS is huge, but building a small OS will go a long way
- More employable
  - Become someone who "understands systems"
  - Join the top group of "athletes"
  - Ability to build things from ground up
  - Deeply understand abstractions and concurrency

37

Does COS318 Require A Lot of Time?

- Yes
- But less than a few years ago
- But yes

38

Why is Writing an OS Hard?

- Concurrent programming is hard
- Difficult to use high-level programming languages for OS
  - device drivers are inherently low-level
  - lack of debugging support (use simulation)
  - real-time requirements
- Tension between functionality and performance
- Different contexts (mobile devices, data centers, embedded)
- Portability and backward compatibility
  - many APIs are already fixed (e.g., GUI, networking)
  - OS design tradeoffs change as hardware changes

Why is Writing an OS Hard

- Needs to be reliable
  - Does the system do what it was designed to do?
- Needs to keep the system available
  - What portion of the time is the system working?
  - Mean Time To Failure (MTTF), Mean Time to Repair
- Needs to keep the system secure
  - Can the system be compromised by an attacker?
- Needs to provide privacy
  - Data is accessible only to authorized users

Main Techniques and Design Principles

- Keep things simple
- Use abstraction
  - hide implementation complexity behind simple interface
- Use modularity
  - decompose system into isolated pieces
- What about performance?
  - find bottlenecks --- the 80-20 rule
  - use prediction and exploits locality (cache)
- What about security and reliability?

Continuing research, particularly in light of new contexts

Things to Do

- Today's material
  - Read *MOS 1.1-1.3*
  - Lecture available online
- Next lecture
  - Read MOS 1.4-1.5
- Make "tent" with your name
  - Use from now on till the end of the semester
- Use Piazza to find a partner
  - Find a partner before the end of next lecture for projects 1, 2 and 3

42