

Introduction

To do ...

- Welcome to OS
- Administrivia
- OS overview and history
- Next time: OS components and structure



symbian
OS

 debian

The logo for Debian, which consists of a red spiral icon followed by the word "debian".

 solaris™

The Solaris logo, which features a stylized orange sun icon above the word "solaris" in a blue serif font.

iOS

 ubuntu

The Ubuntu logo, which features the word "ubuntu" in white lowercase letters on a brown background with colorful circles around it.

Why study operating systems?

- Tangible reasons
 - Build/modify one – OSs are everywhere
 - Administer and use them well
 - Tune your favorite application performance
 - Great capstone course
 - Common source for job interview questions
- Intangible reasons
 - Curiosity
 - Use/gain knowledge from other areas
 - Challenge of designing large, complex systems

What is an operating system?

- An *extended machine* and a resource manager
- Consider a disk
 - Manual for using a Serial ATA hard disk – 450pp ... device driver
 - Devices drivers still too low-level – read/write blocks ... file systems
- Extended machine – top-down/user-view
 - The *illusionist* – Hides the mess, presenting a virtual machine that's easier to program than the hardware
 - *How much memory do you have?*
 - *How is your disk formatted?*
 - *How many processors do you have?*



What is an operating system?

- Resource manager – bottom-up/system-view
 - The *referee* – Everybody gets a fair-share of time/ space from a resource (multiplexing in space/time)
 - A control program that prevent errors & improper use (CP/M?)

Loading CPM.SYS...

CP/M-86 for the IBM PC/XT/AT, Vers. 1.1 (F)
Copyright (C) 1983, Digital Research

Hardware Supported :

Diskette Drive(s) : 3
Hard Disk Drive(s) : 1
Parallel Printer(s) : 1
Serial Port(s) : 1
Memory (Kb) : 640

CP/M Control Program/Monitor

For Intel 8080/85 by Kildall's Digital Research, Inc.
IBM tried to license it for their “new” IBM PC. *Failing to get a non-disclosure agreement they used instead Microsoft to provide an OS*

D>a:
A>dir
A: PIP CMD : STAT CMD : SUBMIT CMD : ASM86 CMD
A: GENCMD CMD : DDT86 CMD : TOD CMD : ED CMD
A: HELP CMD : HELP HLP : SYS CMD : ASSIGN CMD
A: FORMAT CMD : CLDIR CMD : WRTLDR CMD : BOOTPCDS SYS
A: BOOTWIN SYS : CPM H86 : WINSTALL SUB : PD CMD
A: WCPM SYS : DISKUTIL CMD
A>_ User 0 0:00:11 Jan. 1, 2000

What is an operating system?

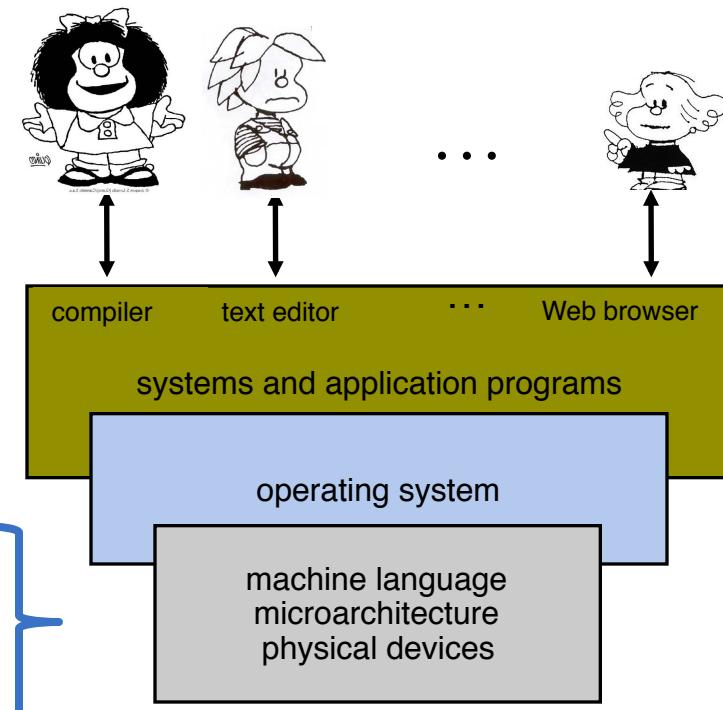
- A bundle of helpful, commonly used things
- Goals and evaluation criteria
 - Reliability – the system does what is designed to do
 - Security and privacy – operation cannot be compromised by an attacker, data stored is only accessible to authorized users
 - Convenience – make solving user problems easier
 - Efficiency – use hardware efficiently (\$\$ machines!)
 - Easy to modify/evolve/port



A computer system - Where's the OS?

- Hardware provides basic computing resources
- Applications define ways in which resources are used to solve users' problems
- OS *controls & coordinates use of hardware by users' applications*
- A few vantage points
 - End user
 - Programmer
 - OS Designer

1+ processors, main memory, disks,
keyboards, display, camera, network
interfaces, mouse pad, ...



What's part of the OS?

- Trickier than you think: file system, device drivers, shells, window systems, browser, ...
- Everything a vendor ships with your order?
- Program running at all times, in kernel mode
 - Everything else is either a system program (ships with the OS) or an application program
 - Can the user change it?
- Why does it matter? In 1998 the US Dept. of Justice filed suit against MS claiming its OS was too big

The course

- Everything you need to know

<http://www.aqualab.cs.northwestern.edu/classes/eecs-343-f16/>

- Course staff

- Fabián Bustamante
 - Andrew Weiner (TA)

Course overview ...

- ~~Homework~~ In-class quizzes (8)
 - In part, reading enforcers – textbook
- Readings
 - Optional set of papers; outline in the webpage
 - Basis for extra-credit questions in exams
- Projects (4) *all in xv6*
 - To do in groups of 2-3
- Exams (2)
 - Final (not cumulative)
- *Whatever you have heard, it's not that bad!*

On reading

- Optional papers (others as well, but these few for exams)
 1. D.D. Redell et al., ***Pilot: An Operating System for a Personal Computer***, CACM, Feb. 1980.
 2. C. Waldspurger et al., ***Lottery scheduling: Flexible proportional-share resource management***, OSDI, 1994.
 3. H. Levy et al., ***Virtual Memory Management in the VAX/VMS Operating System***, IEEE Computer, Mar. 1982.
 4. T. Anderson et al., ***Scheduler activations: effective kernel support for the user-level management of parallelism***, ACM TOCS, 10(1): 53-79, Feb. 1992.
 5. B. Lampson et al., ***Experiences with Processes and Monitors in Mesa***, CACM, Feb. 1980.
 6. M. Rosenblum et al., ***Design and implementation of the log-structured file system***, SOSP, 1991.
- Mandatory
 - S. Peter et al., ***Arakis: The Operating System is the Control Plane***, OSDI, 2014.

On xv6 projects

- A new model for OS @ Northwestern
- Hacking MIT's xv6
 - What's that? A reimplementation of Unix 6 (1971) for x86 architectures
 - Inspired by Lions's Commentary on UNIX 6
- Why?
 - You get to play with a real OS
 - Without dying under 15 million LOC (<80k)
 - More reading and understanding code than writing
 - Hacking is how you internalize systems concepts

On projects – Code walkthrough

- For every project, code walkthrough by randomly selected groups with the staff

To get full credit for your project you must be able to carry the walkthrough, showing you understand your own code

- A few points
 - Sampling with replacement
 - To be done in first week after submission
 - Every team member must be there

Your TODO list

- Project 1 will posted **TOMORROW**
 - Due Sep. 30th
- Check you have access to T-Lab / Wilkinson
 - Email help@eecs otherwise
- Start reading book, papers and code
 - Xv6, a simple Unix-like teaching operating system
<http://www.aqualab.cs.northwestern.edu/class/333-eecs343-xv6>
- Start looking for partners for your project!

On class participation

- A Wong-Baker's like scale



Contributes to the discussion (A)



Seems to follow the discussion (B)



I've seen her in class (C)



She's in class, but probably on FB (D)



Is he actually in this class? (F)

On that topic ...



Computers & Education

Volume 62, March 2013, Pages 24–31



Laptop multitasking hinders classroom learning for both users and nearby peers

Faria Sana^a,  , Tina Weston^{b, c},  , Nicholas J. Cepeda^{b, c},  , 

The Pen Is Mightier Than the Keyboard Advantages of Longhand Over Laptop Note Taking

Pam A. Mueller¹

Daniel M. Oppenheimer²

¹Princeton University

²University of California, Los Angeles

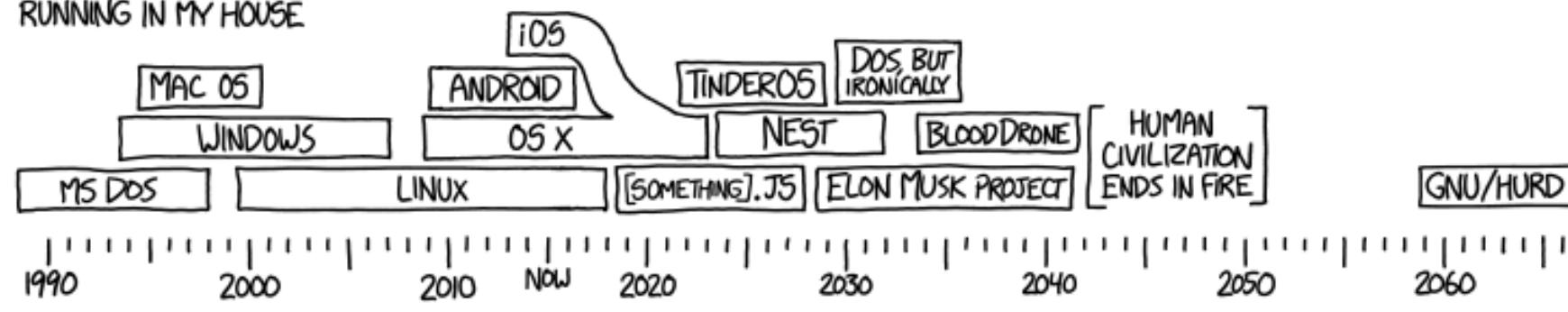
Pam A. Mueller, Princeton University, Psychology Department, Princeton, NJ 08544 E-mail: pamuelle@princeton.edu

On cheating/plagiarism

- Collaboration is good
- Cheating is a very serious offense
- It's OK to
 - Meet with colleagues
 - Study for exams together
 - Discuss assignments with them
- But, *what you turn in must be your own work*
 - Copying code, solution sets, etc. from other people or any other sources (Web included!) is strictly prohibited

And now a short break ...

OPERATING SYSTEMS RUNNING IN MY HOUSE

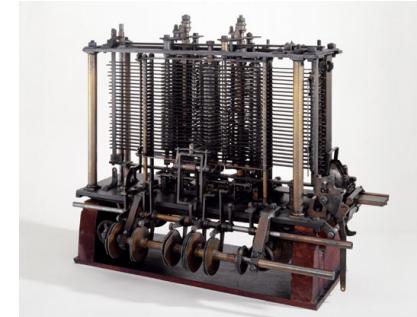


Hardware/Software evolution/change

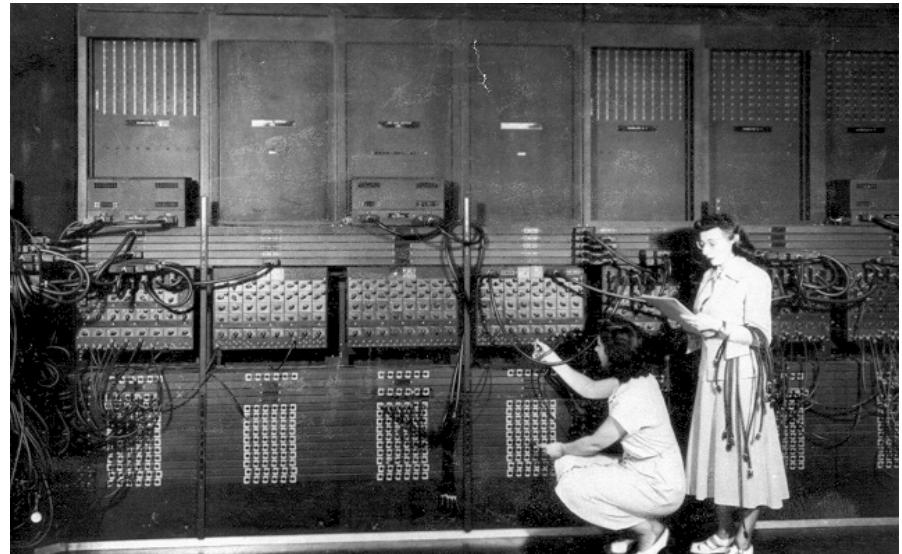
- 1960s Mainframes
- 1970s Minicomputers
- 1980s Microprocessors, workstations, LANs, Internet
- 1990s PC, Web
- 2000s Internet services, mobile computing
- 2010s Pervasive, sensor networks, cloud computing, data-intensive computing
- *It's up to you*

The evolution of operating systems

- Early attempts – Babbage's (1702-1871)
 - Analytical Engine (Ada Lovelace – World's first programmer)



- 1945-55 – Vacuum tubes & plugboards
 - ABC, MARK 1, ENIAC, later IBM 701 (1952)
 - No programming languages, no OS
 - A big problem
 - Scheduling – signup sheet on the wall



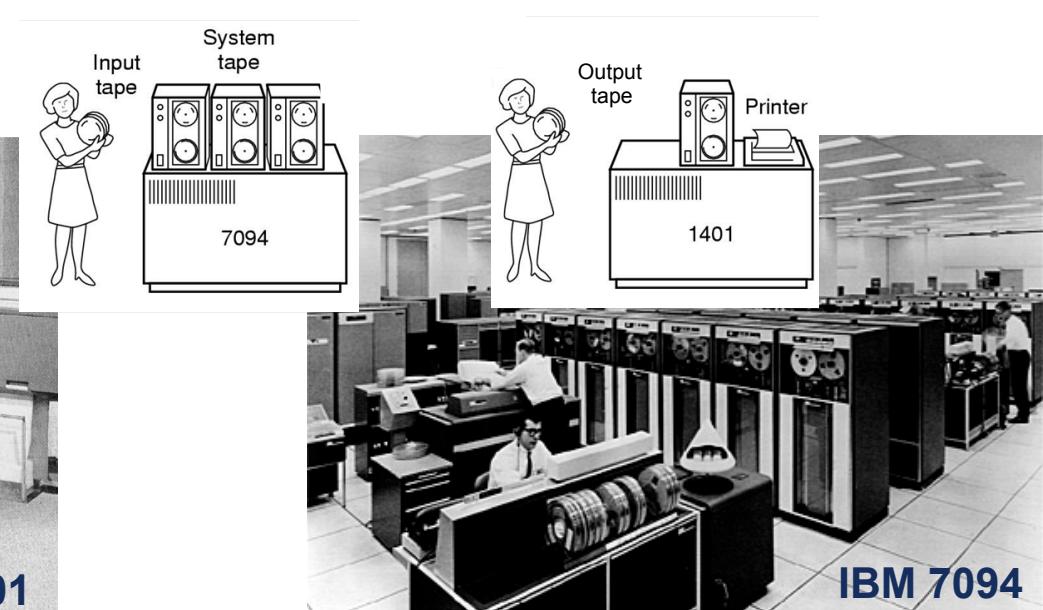
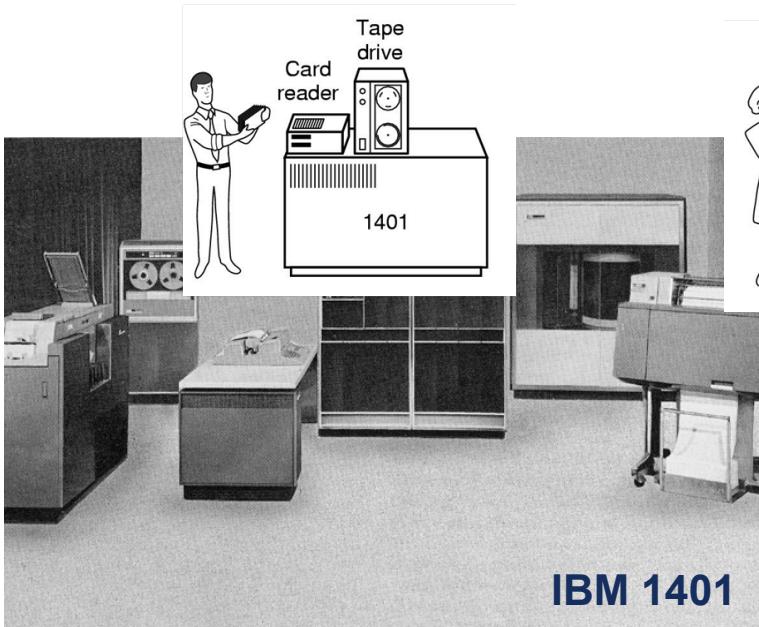
Evolution ... Batch systems (1955)

- Transistors → machs. reliable enough to sell
 - Separation of builders & programmers
- Getting your program to run
 - Write it in paper (maybe in FORTRAN)
 - Punch it on cards & drop cards in input room
 - Operator may have to mount/dismount tapes,
setting up card decks, ... setup time!
- Batch systems ...

Evolution ... Batch systems (1955)

• Batch systems

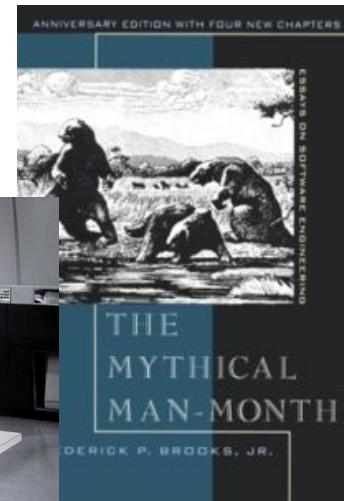
- Collect a tray of full jobs, read them into tape with a cheap computer
- Bring them to the main computer where the “OS” will go over each job one at a time
- Print output offline



Evolution ... Multiprogramming (1960s)

- To increase system utilization
 - Keeps multiple runnable jobs in memory at once
 - Overlap I/O of a job with computing of another
 - Needs asynchronous I/O devices
 - Some way to know when devices are done
 - Interrupt or polling
 - Goal- optimize system throughput
 - Maybe at the cost of response time

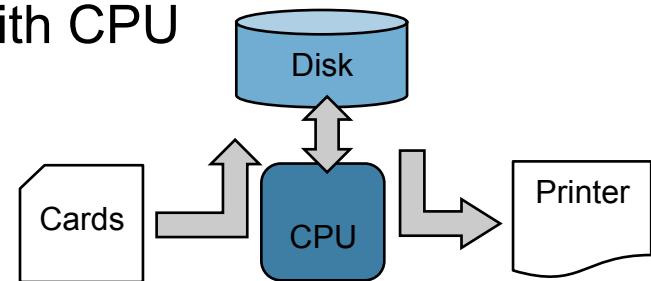
IBM OS/360 &
the tar pit



Evolution ... Spooling (1960s)



- Disks, random access, much faster than card readers & printers
- Simultaneous Peripheral Operations On-Line
 - While one job runs, spool next one from card reader onto disk
 - Slow card reader I/O overlapped with CPU
 - With multiple programs on disk, scheduling
- But
 - CPU still idle when program interact with a peripheral during execution
 - Interactivity was gone



Evolution ... Timesharing (1960s)

- To support interactive use
 - Multiple users on terminals, one machine
 - Each user has the illusion of owing the machine
- Time-slicing
 - Dividing CPU equally among users
 - If jobs are interactive, CPU can jump between them without users noticing it
 - Optimize response time maybe at the cost of throughput
 - Recovers interactivity for the user (So what?)
- CTSS (Compatible Time Sharing System), MULTICS (second-system effect) and UNIX

Evolution ... Parallel systems (1970s)

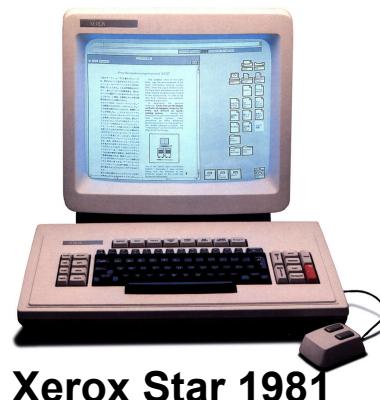
- Some apps can be written as multiple tasks
 - Speed up when running on several CPUs
 - Need OS and language primitives for dividing/organizing the multiple tasks
 - Need OS primitives for fast comm. between tasks
 - Speedup dictated by communication/computation ratio
 - Many flavors of parallel computers today
 - SMPs – symmetric multi-processors, multi-core
 - SMT – simultaneous multithreading [“hyperthreading”]
 - MPPs – massively parallel processors
 - NOWs – networks of workstations, clusters
 - Computational grid (SETI @home)

Evolution ... PCs (1980s)

- Large-scale integration >> small & cheap machines
- 1974 – Intel's 8080 & Gary Kildall's CP/M
- Early 80s – IBM PC, BASIC, CP/M & MS-DOS
- User interfaces, XEROX Altos, MACs, Windows



Xerox Alto 1973



Xerox Star 1981

IBM PC circa 1981



Evolution ... Distributed and pervasive

- Facilitate use of geographically distributed resources
 - Workstations on a LAN or across the Internet
- Support communication between programs
- Speed up is not always the issue, but access to resources (including information)
- Architectures
 - Client/servers
 - Mail server, print server, web server
 - Peer-to-peer
 - (Most) everybody is both, server and client

Evolution ... Embedded and pervasive

- Pervasive computing
 - Cheap processors embedded everywhere
 - Wearables, glasses, thermostats, light bulbs...
- Typically very constrained hardware resources
 - Slow processors (all relative)
 - Small amount of memory
 - No disk or tiny disk
 - Typically only one dedicated application
 - Limited power
- But technology changes fast

Major OS issues

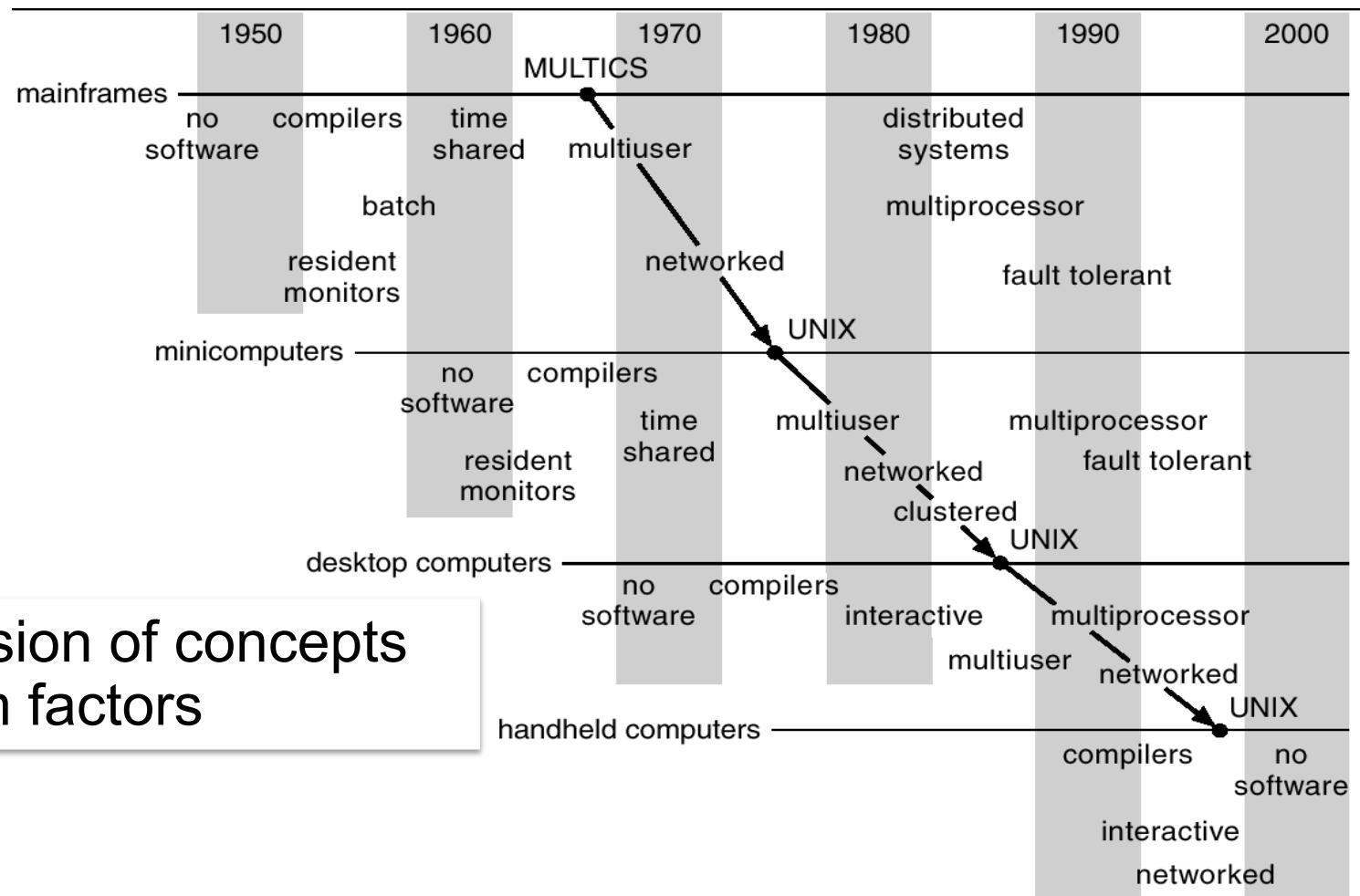
- Structure: *how is the OS organized?*
- Sharing: *how are resources shared?*
- Naming: *how are resources named?*
- Security: *how is integrity of the OS and its resources ensured?*
- Protection: *how is one user/program protected from another?*
- Performance: *how do we make it all go fast?*
- Reliability: *what happens if something goes wrong?*
- Extensibility: *can we add new features?*
- Communication: *how do programs exchange information, including across a network?*

Other OS issues

- Concurrency: *how are parallel activities created and controlled?*
- Scale and growth: *what happens as demands or resources increase?*
- Persistence: *how do you make data last longer than program executions?*
- Distribution: *how do multiple computers interact with each other? how do we make distribution invisible?*
- Accounting: *how do we keep track of resource usage, and perhaps charge for it?*

There are a huge number of engineering tradeoffs in dealing with these issues!

“Ontogeny recapitulates phylogeny”*



Progression of concepts
and form factors

The development of an embryo repeats the
evolution of the species (* Ernst Haeckel)

Has it all been done then?

- **Constant new challenges**
 - Embedded computing and sensor networks
 - Multiple cores
 - Cloud services
 - Peer-to-peer computing
 - Software defined networking
- **Old problems redefine themselves**
 - Evolution of smart phones recapitulated that of PC, which recapitulated that of minicomputers which ...
 - But the ubiquity of PC's redefined the issues of protections, phones are doing this again ...

Summary

- In this class you will learn
 - OS components and structures
 - The most important interfaces
 - Common policies in an OS
 - Algorithms to implement those policies
- Philosophy
 - You may never build an OS, but
 - You need to understand the foundations
 - OSs exemplify the sorts of engineering tradeoffs you'll need to make throughout your careers