CS202 – Advanced Operating Systems

Background

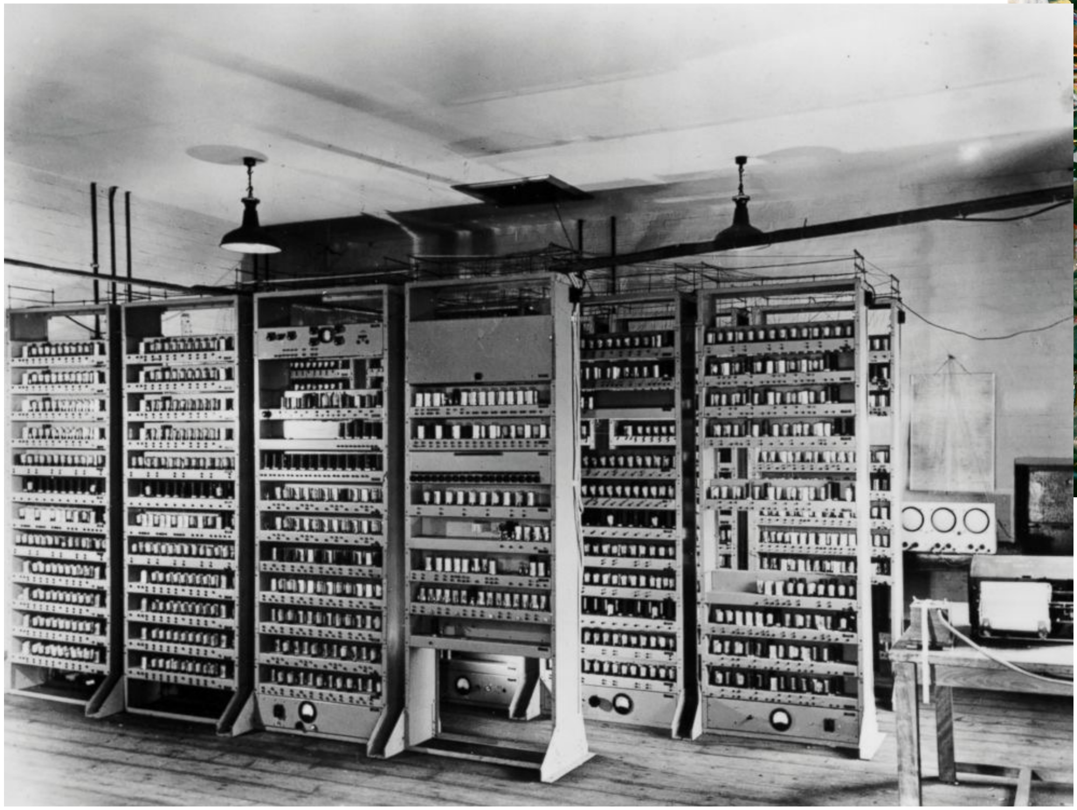
January 8, 2025

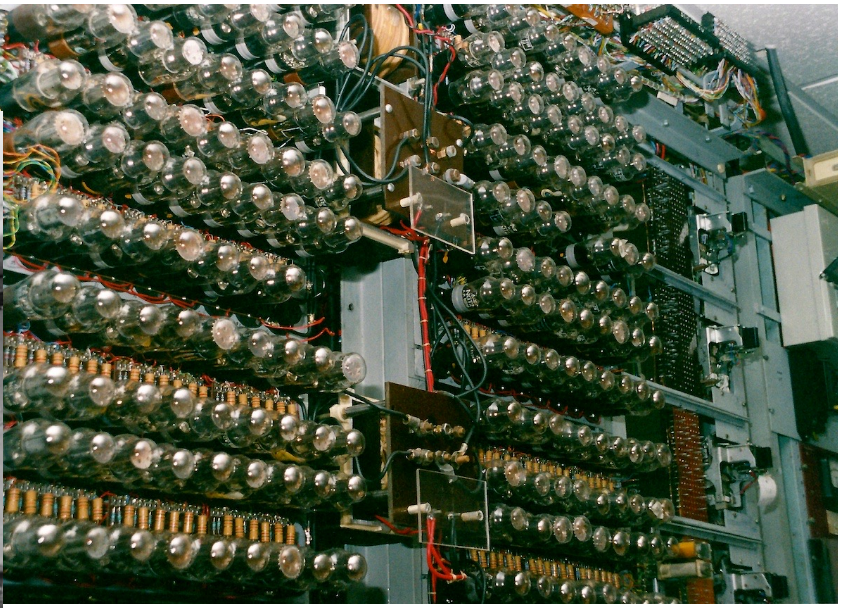
Operating Systems Models

- Our first topic will explore OS models
  - Why do Operating Systems look the way they do?
  - What drives the decisions? What else is possible?

- To set the table
  - Today, we will do a walk-through of the historical evolution of OS
  - Next, we will discuss the OS's process abstraction of the CPU

Dawn of Computing

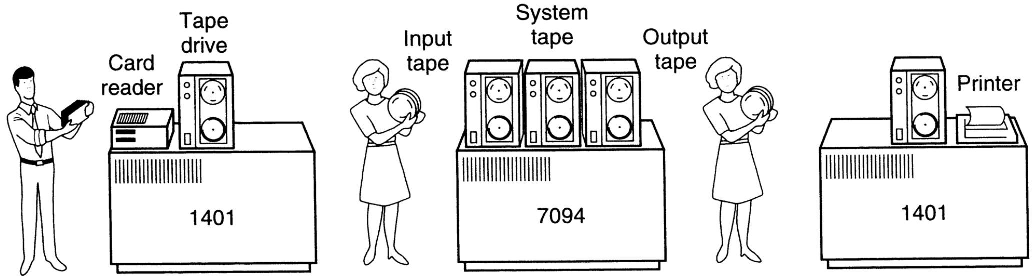




Phase 1: 1955-1970

- Computers expensive; people cheap
  - Use computers efficiently – move people away from machine
  - OS becomes a batch monitor
    - Loads a job, runs it, then moves on to next
    - If a program fails, OS records memory contents somewhere
    - More efficient use of hardware but increasingly difficult to debug

- Batch systems on *mainframe* computers
  - collections of jobs made up into a *batch*
  - example: IBM 1401/7094
    - card decks spooled onto magnetic tape and from tape to printer



  - example: English Electric Leo KDF9
    - 32K 48-bit words, 2μsec cycle time
    - punched paper-tape input ‘walk-up’ service or spooling via mag tape

IBM 7094, thought to be first computer singing (1961)
https://youtu.be/yIwhx3NQSLg

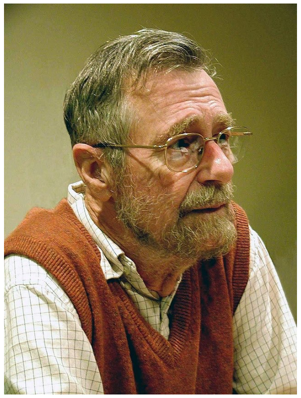
Phase 1, problems

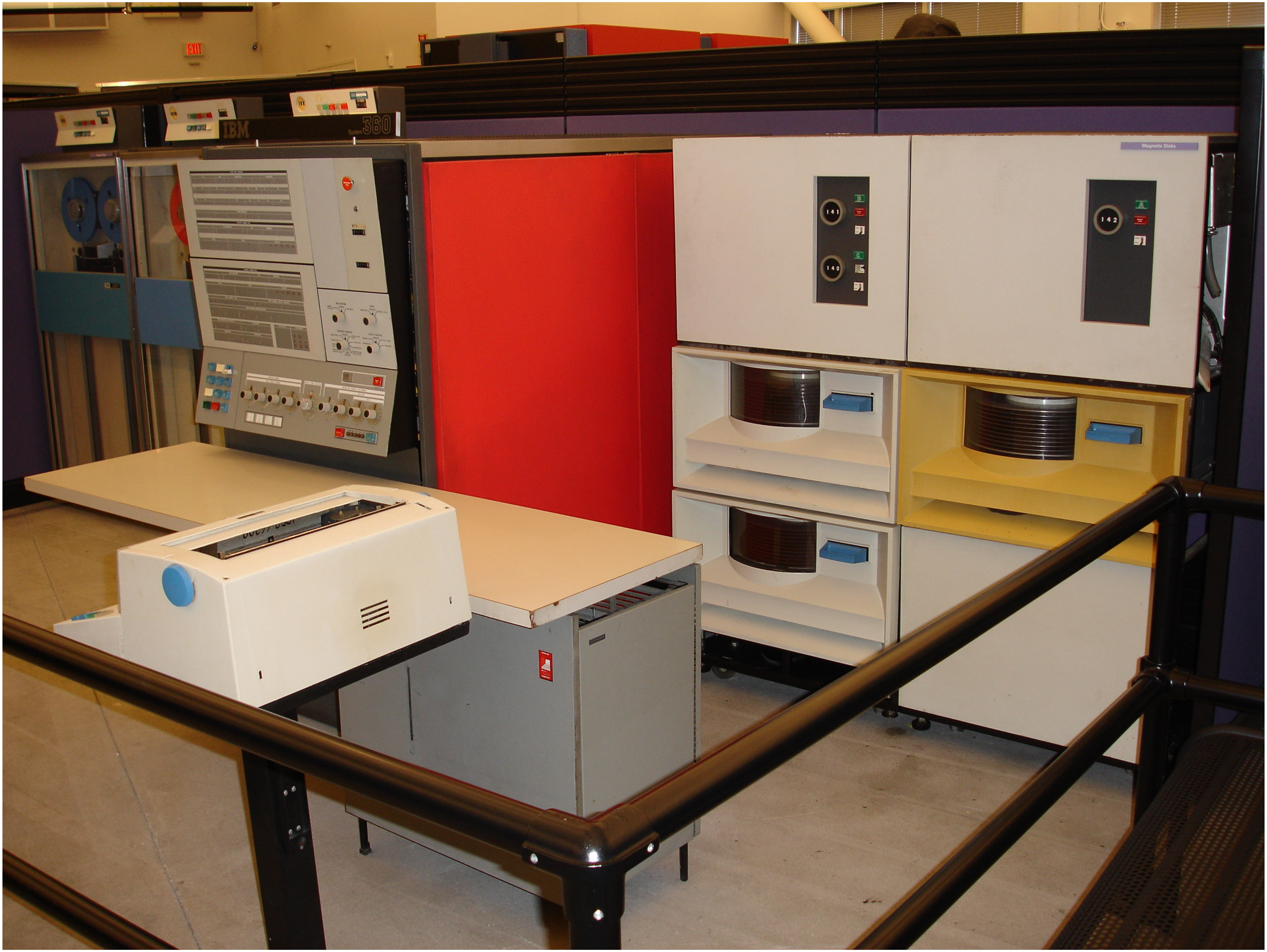
- Utilization is low (one job at a time)
- No protection between jobs
- Short jobs wait behind long jobs
  - So, we can only run one job at a time
- Coordinating concurrent activities
- Still painful and slow (but less so?)

Some important projects

- Atlas computer/OS from Manchester U. (late 50s/early 60s)
  - First recognizable OS
  - Separate address space for kernel
  - Early virtual memory

- THE Multiprogramming system (early 60s)
  - Introduced semaphores
  - Attempt at proving systems correct; interesting software engineering insights
  - Project lead by Dijkstra (Turing award winner)




EXIT
IBM
System 360
Magnetic Disks
141
140
142

Phase 2: 1970s

- Computers and people are expensive
  - Help people be more productive
  - Interactive time sharing: let many people use the same machine at the same time
  - Emergence of minicomputers
    - Terminals are cheap
  - Keep data online on fancy file systems
  - Attempt to provide reasonable response times (Avoid thrashing)

Important advances and systems

- Compatible Time-Sharing System (CTSS)
  - MIT project (demonstrated in 1961)
  - One of the first time-sharing systems
  - Corbato won Turing award in 1990
  - Pioneered much of the work in scheduling
  - Motivated MULTICS

Multics

- Jointly developed by MIT, Bell Labs and GE
- Envisioned one main computer to support everyone
    - People use computing like a utility like electricity – sound familiar? Ideas get recycled
- Many, many fundamental ideas: protection rings, hierarchical file systems, devices as files, mandatory security, etc.
- Building it was more difficult than expected
- Technology caught up

Unix appears

- Ken Thompson, who worked on Multics, wanted to use an old PDP-7 laying around in Bell Labs
- He and Dennis Ritchie built a system designed by programmers for programmers
- Originally in assembly. Rewritten in C
  - If you notice for the paper, they are defending this decision
  - However, this is a new and important advance: portable operating systems!
- Shared code with everyone (particularly universities)

Ken Thompson

Dennis M. Ritchie

1983 Turing Award for Unix

Unix (cont’d)

- Berkeley added support for virtual memory for the VAX
- DARPA selected Unix as its networking platform in Arpanet
- Unix became commercial
  - …which eventually lead Linus Torvalds to develop Linux

Some important ideas in Unix

- OS written in a high-level language
- OS portable across hardware platforms
  - Computing is no longer a pipe stove/vertical system
- Pipes
  - E.g., grep foo file.txt | wc -l
- Mountable file systems
- Many more (we'll talk about Unix a lot)

Phase 3: 1980s

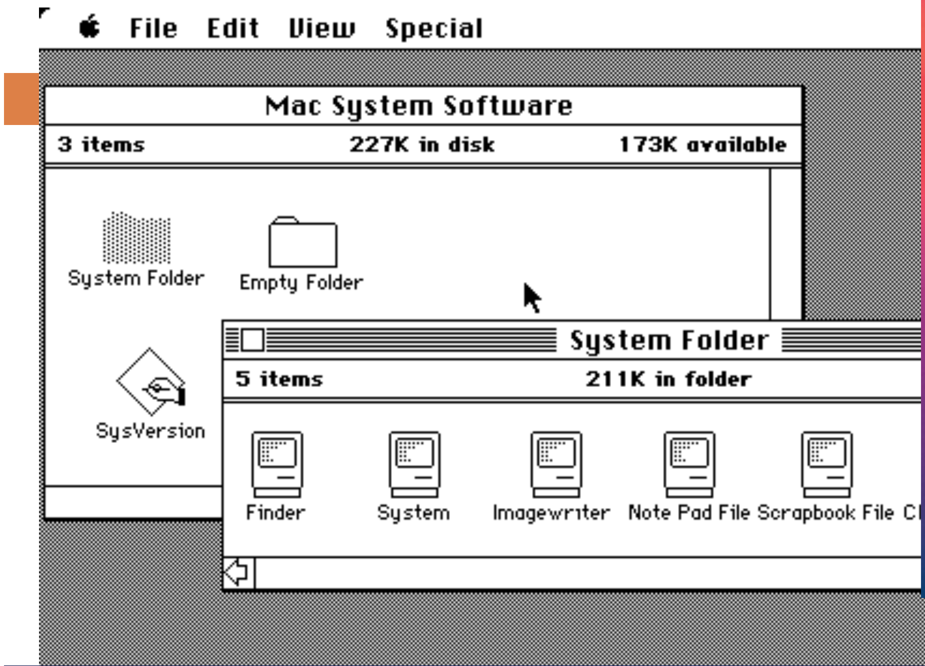
- Computers are cheap, people expensive
  - Put a computer in each terminal
  - CP/M from DEC first personal computer OS (for 8080/85) processors
  - IBM needed software for their PCs, but CP/M was behind schedule
  - Approached Bill Gates to see if he can build one
  - Gates approached Seattle computer products, bought 86-DOS and created MS-DOS
  - Goal: finish quickly and run existing CP/M software
  - OS becomes subroutine library and command executive

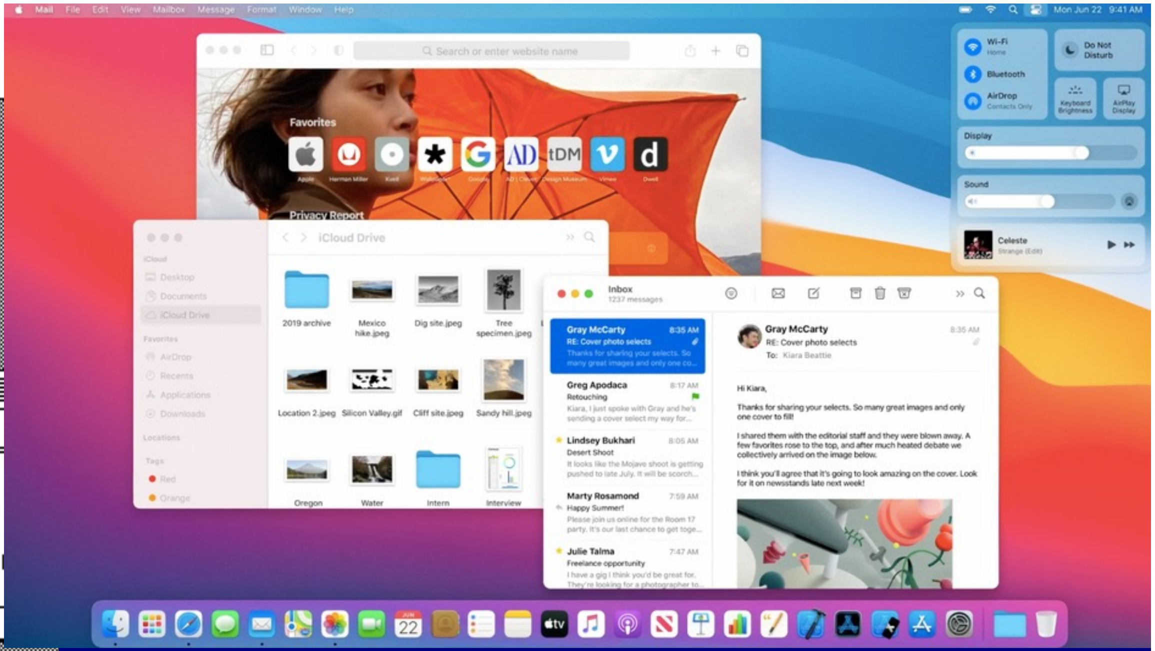
Disk Operating System (DOS)

- Introduced in 1981 for IBM PC based on 8086/8088
- Only 640KB memory available for applications
  - No virtual memory
  - Need quite a few tricks to use all memory that you installed on the computer
- No multi-user, no multi-tasking, no multi-threading
- Notorious 8.3 filename restrictions
- No GUI
  - Now the command line environment of Windows
  - Windows is originally a graphic user interface running on DOS — like X-Windows

New advances in OS

- PC OS was a regression for OS
  - Stepped back to primitive Phase 1 style OS leaving the cool developments that occurred in Phase 2
- Academia was still active, and some developments still occurred in mainframe and workstation space
- Eventually, Windows, Linux, MacOS, took over
  - Phase 2 OS’s making it to PCs
  - GUIs!





Gray McCarty 8:35 AM
RE: Cover photo selects
Thanks for sharing your selects. So many great images and only one co...

Greg Apodaca 8:17 AM
Retouching
Kiara, I just spoke with Gray and he's sending a cover select my way for...

Lindsey Bukhari 8:05 AM
Desert Shoot
It looks like the Mojave shoot is getting pushed to late July. It will be scorch...

Marty Rosamond 7:59 AM
Happy Summer!
Please join us online for the Room 17 party. It's our last chance to get toge...

Julie Talma 7:47 AM
Freelance opportunity
I have a gig I think you'd be great for. They're looking for a photographer to...

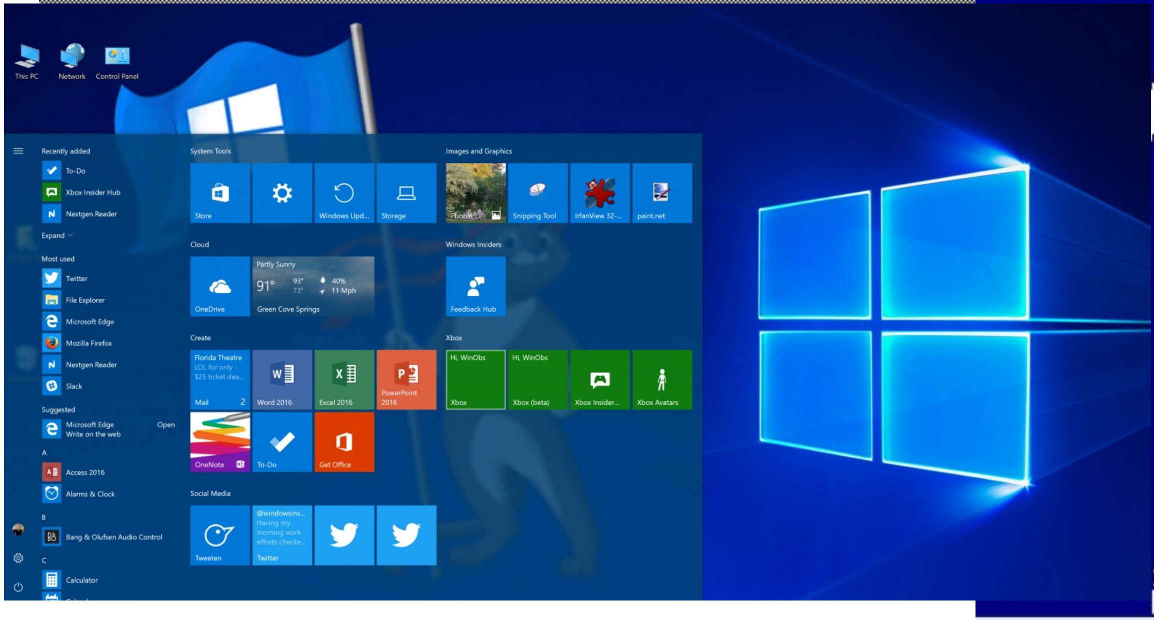
Gray McCarty 8:35 AM
RE: Cover photo selects
To: Kiara Beattie

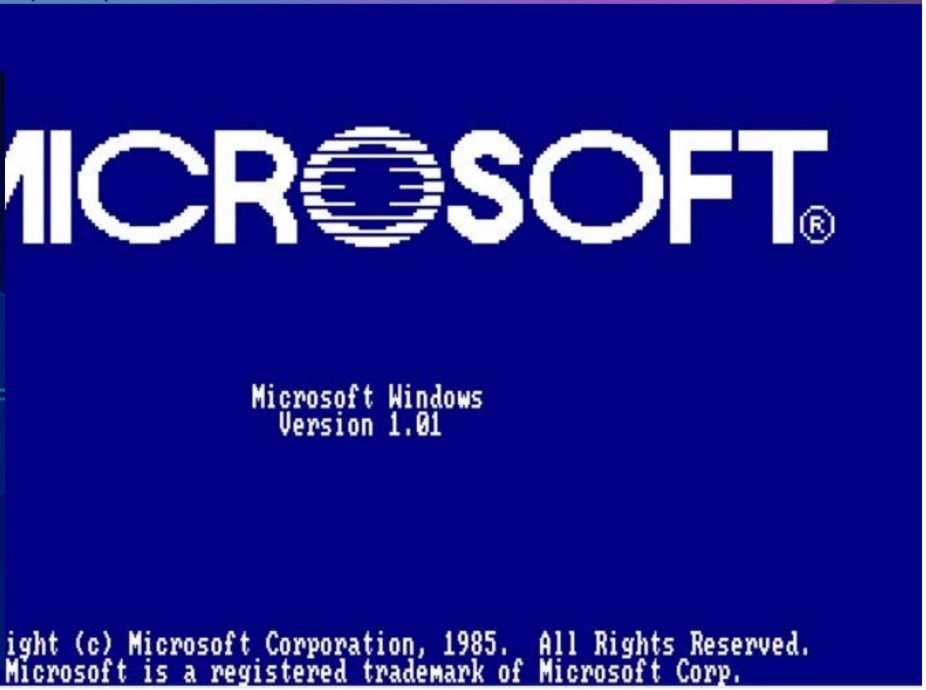
Hi Kiara,

Thanks for sharing your selects. So many great images and only one cover to fill!

I shared them with the editorial staff and they were blown away. A few favorites rose to the top, and after much heated debate we collectively arrived on the image below.

I think you'll agree that it's going to look amazing on the cover. Look for it on newsstands late next week!





Microsoft Windows
Version 1.01

ight (c) Microsoft Corporation, 1985. All Rights Reserved.
Microsoft is a registered trademark of Microsoft Corp.

Phase 4: Networked systems 1990s to 2010s

- Machines can talk to each other
  - its all about connectivity
- We want to share data not hardware
- Networked applications drive everything
  - Web, email, messaging, social networks, …
- Multiuser less important for personal machines
  - But more important for servers
- Security becomes a significant issue
  - Even in single-user systems

Phase 4, continued

- Market place continued horizontal stratification
  - ISPs (service between OS and applications)
  - Information is a commodity
  - Advertising a new marketplace

- New network-based architectures
  - Client-server
  - Clusters
  - Grids
  - Distributed operating systems
  - Cloud computing (or is that phase 5?)

New problems

- Large scale
  - Google file system, mapreduce, …
- Concurrency at large scale
  - ACID (Atomicity, Consistency, Isolation and Durability) in Internet Scale systems
    - Very large delays
    - Partitioning
- Security and Privacy

Phase 5: 2010s -- ??

- New generation?
- Mobile devices that are powerful
- Sensing: location, motion, …
- Cyber-physical systems
- Machine learning everywhere
- Computing evolving beyond networked systems
  - But OS for them looks largely the same
  - Is that a good idea?

Reading Research Papers

- What is the purpose of reading research papers?
  - How do you read research papers?

Understanding What You Read

- Things you should be getting out of a paper
  - What central idea is proposed/explored in the paper?
    - Abstract
    - Introduction
    - Conclusions

These are the best areas to find an overview of the **Contribution**

  - **Motivation**: What is the problem being addressed?
    - How does this work fit into others in the area?
  - **Related work**: often a separate section, sometimes not, every paper should detail the relevant literature.
    - Papers that do a limited or superficial job are almost sure to be bad.
  - An informed reader should be able to read the related work and understand the basic approaches in the area, and why they do not solve the problem effectively

Understanding What You Read

- What scientific devices are the authors using to communicate their point?
- **Methodology** – how they describe their solution.
  - Theoretical papers typically describe a model using mathematical arguments (e.g., proofs)
  - Experimental papers design a test apparatus (e.g., build a system that aims to satisfy some claims)
  - Empirical research measures a claimed scenario
- Modern systems papers tend to have extensive evaluations to assess its claims

Understanding What You Read

- What is the impact of the paper?
  - **Results** - statement of new scientific discovery
    - Typically some abbreviated form of the results will be present in the abstract, introduction, and/or conclusions
    - Note: just because a result was accepted into a conference or journal does necessarily not mean that it is true. Always be circumspect.
- What should you remember about this paper?
  - **Take away** - what general lesson should you take away from the paper?
  - Note that really good papers will have take-aways that are more general than the paper topic

Summarizing an Article

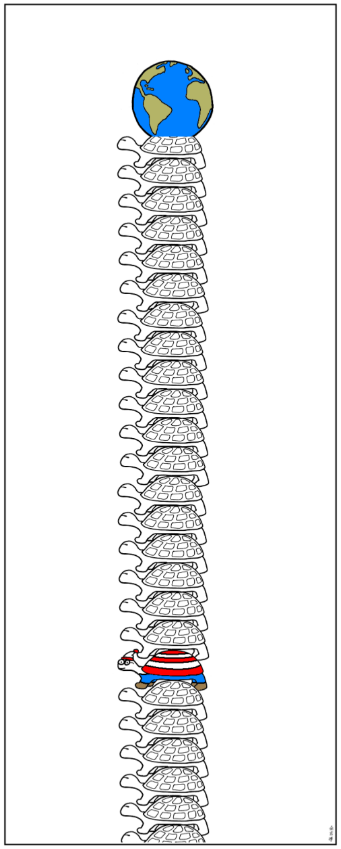
- □ Contribution
- □ Motivation
- □ Related work
- □ Methodology
- □ Results
- □ Take away

Summarizing Thompson Article

- **Contribution**: Ken Thompson shows how hard it is to trust the security of software in this paper. He describes an approach whereby he can embed a Trojan horse in a compiler that can insert malicious code (e.g., in a login program).
- **Motivation**: People need to recognize the security limitations of systems and tools you use.
- **Related Work**: This approach is an example of a Trojan horse program.
  - A Trojan horse is a program that serves a legitimate purpose on the surface, but includes malicious code that will be executed with it. Examples include the Sony/BMG rootkit: the program provided music legitimately, but also installed spyware.
- **Methodology**: The approach works by generating a malicious binary that is used to compile compilers. Since the compiler code looks OK and the malice is in the binary compiler compiler, it is difficult to detect.
- **Results**: The system identifies construction of login programs and miscompiles the command to accept a particular password known to the attacker.
- **Take away**: What is the transcendent truth????? (see next slide)

Summarizing Thompson Article

- **Take away:** Thompson states the “obvious” moral that “you cannot trust code that you did not totally create yourself.” We all depend on code, but constructing a basis for trusting it is very hard, even today.
- ... or trust is an infinite regression ...



“A well-known scientist (some say it was Bertrand Russell) once gave a public lecture on astronomy. He described how the earth orbits around the sun and how the sun, in turn, orbits around the center of a vast collection of stars called our galaxy. At the end of the lecture, a little old lady at the back of the room got up and said: "What you have told us is rubbish. The world is really a flat plate supported on the back of a giant tortoise." The scientist gave a superior smile before replying, "What is the tortoise standing on?" "You're very clever, young man, very clever", said the old lady. "But it's turtles all the way down!"
- Hawking, Stephen (1988). A Brief History of Time.

Conclusions

- Today was a whirlwind history of OS eras
- OS designs and features have evolved in phases as people discovered what support their programs need to operate efficiently and reliably
  - It has been a long windy road
- Overview of processes next time
  - But please read on your own if you need more
  - From the Three Easy Steps textbook

Questions