

Jin-Soo Kim  
([jinsoo.kim@snu.ac.kr](mailto:jinsoo.kim@snu.ac.kr))

Systems Software &  
Architecture Lab.

Seoul National University

Fall 2026

# Introduction to Operating Systems



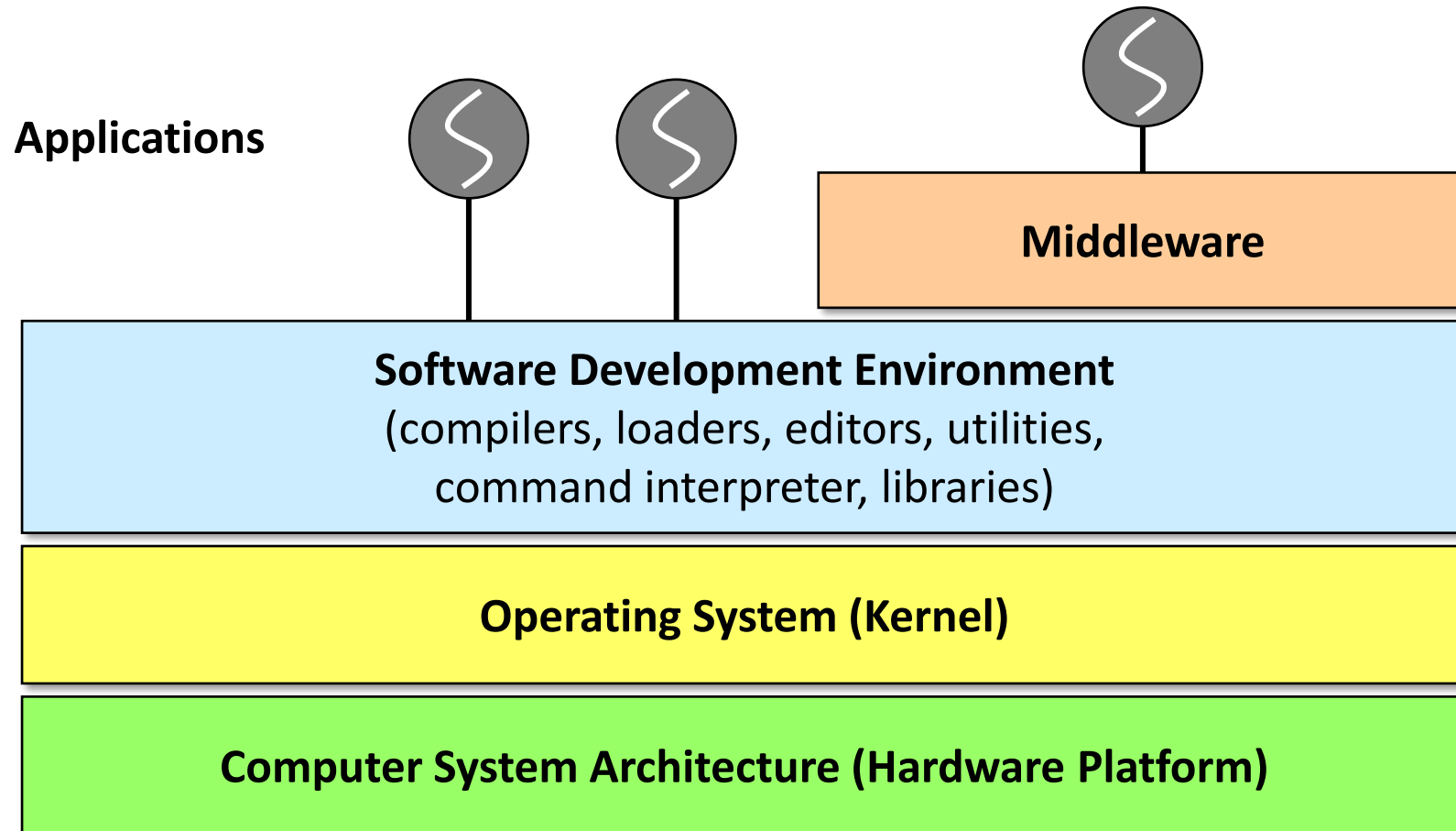
# Why OS?



©RetroPlanet.com

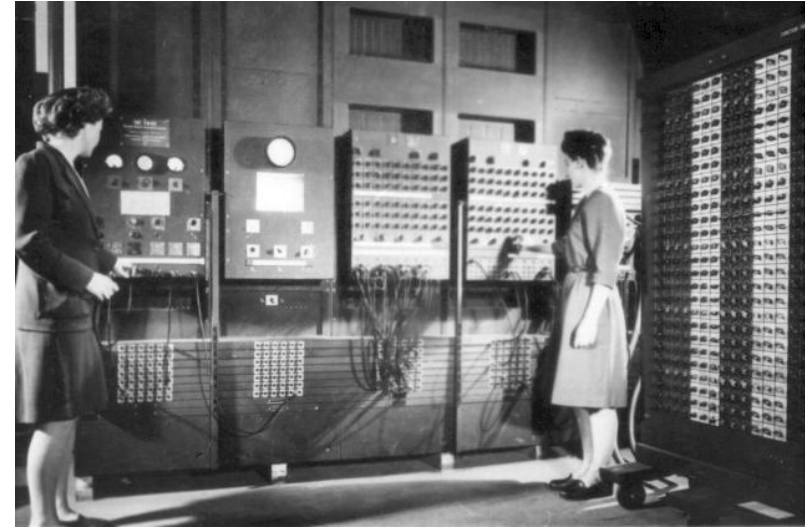
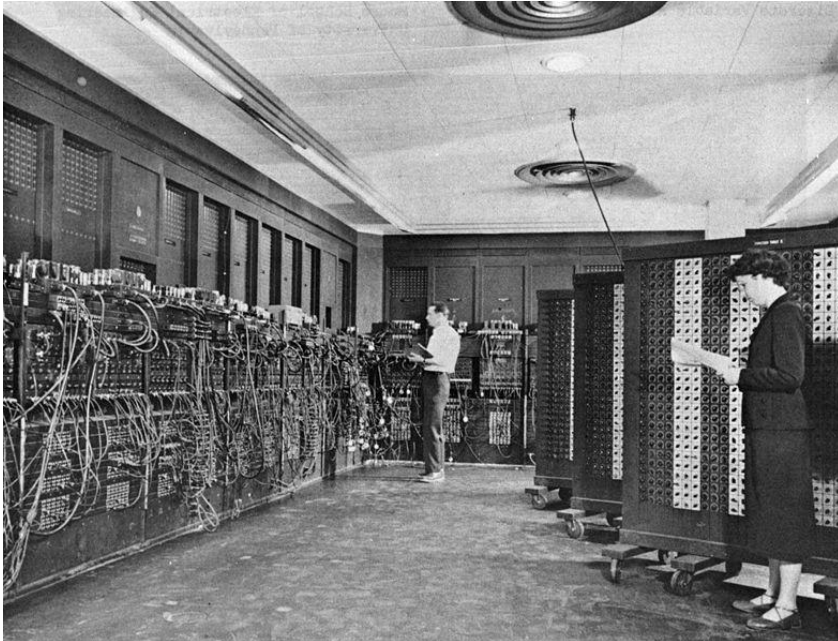
# What is an OS?

- \_\_\_\_\_ that converts hardware into a useful form for \_\_\_\_\_



# IG (1945-55)

- Vacuum tubes and plugboards
  - No OS
  - No programming languages
  - No assembly languages



**ENIAC** (Electronic Numerical Integrator And Computer), 1946

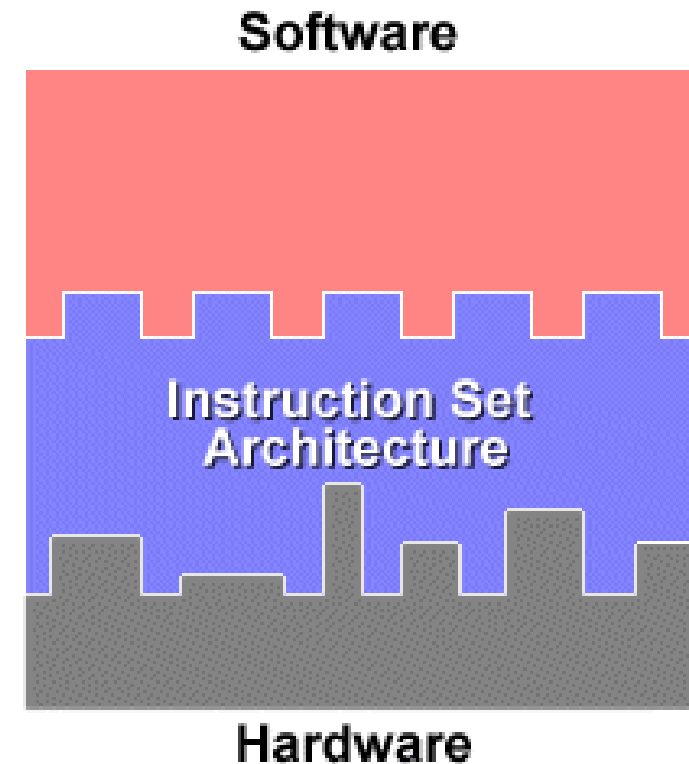
# 2G (1955-65)

- Transistors and mainframes
- \_\_\_\_\_ systems
  - One job at a time
  - Card readers, tape drives, line printers
- OS is always resident in memory and merely transfers control (just a library)
- CPU is underutilized due to the bottleneck in I/O



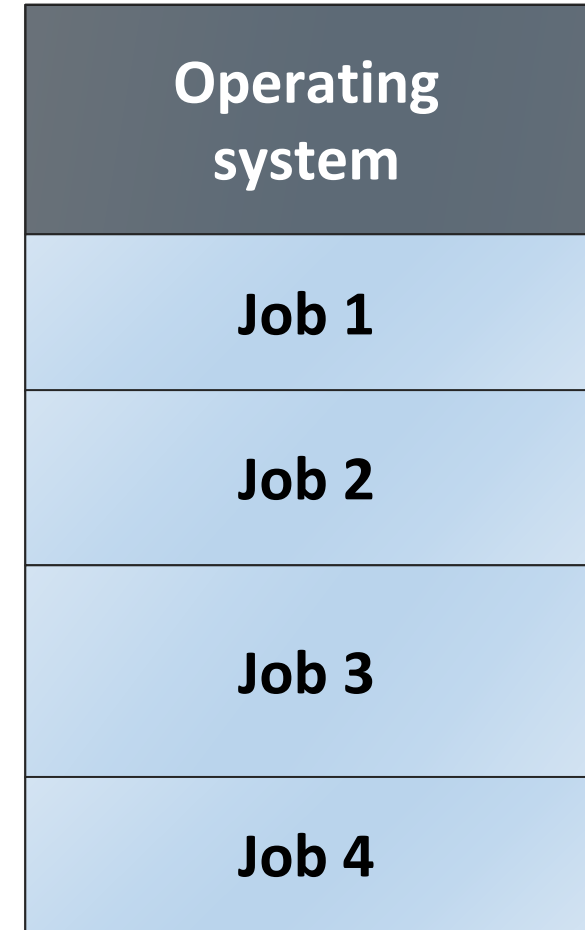
# 3G (1965-80)

- Architectural advances
  - Integrated Circuits (ICs): better price/performance
  - Disk drives
  - On-line terminals
- Established the notion of “Computer Architecture”
  - IBM System/360 Family



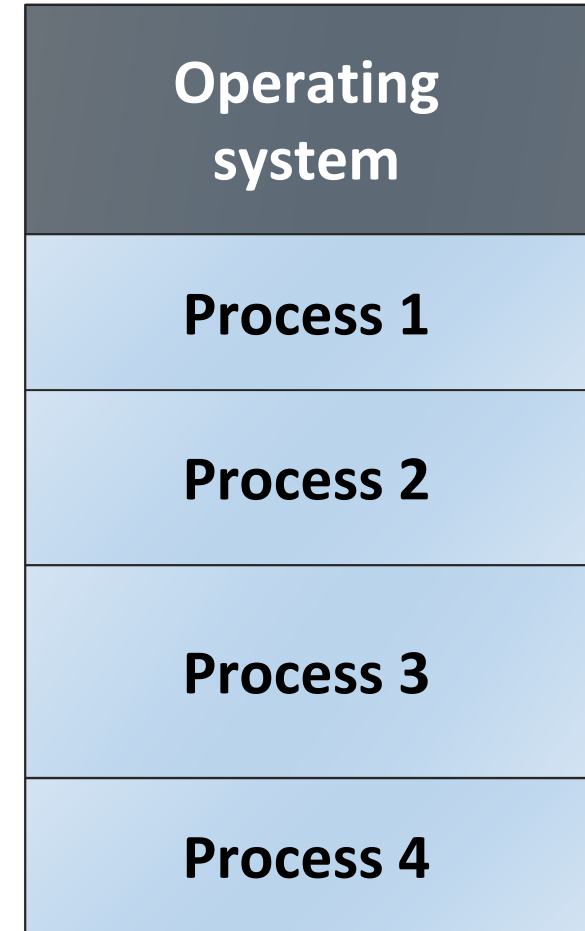
# 3G (1965-80)

- systems
  - Increase CPU utilization
  - IBM OS/360 (1964)
- OS features
  - Job scheduling
  - Memory management
  - CPU scheduling
  - Concurrency
  - Protection
  - Spooling  
(Simultaneous Peripheral Operation On-Line)



# 3G (1965-80)

- systems
  - Improve response time
  - MIT CTSS (1961), Multics (1965), Unix (1969)
- OS features
  - Sophisticated CPU scheduling
  - Virtual memory and swapping
  - File system
  - Synchronization
  - Interprocess communication (IPC)
  - Interactive shell
  - More protection, ...



# 4G (1980-)

- Architectural advances

- Microprocessors (LSIs & VLSIs): smaller and faster
- Storage: larger and faster
- Personal computers
- CPU work is offloaded to I/O devices

- Modern OS features

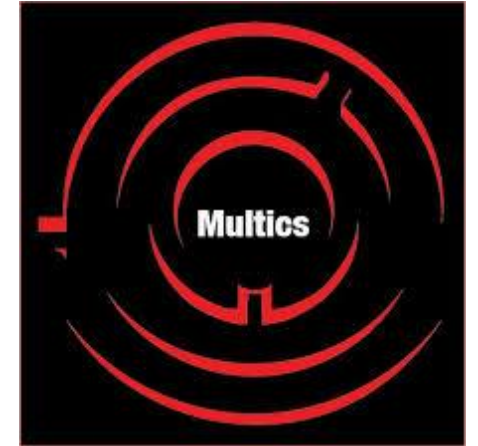
- GUI (Graphical User Interface)
- Multimedia
- Internet & Web
- Mobile / Networked / Distributed
- Virtualization, etc.

# OS History

- CTSS (1961, MIT)
  - Compatible Time-Sharing System
- OS/360 (1964, IBM)
  - Multiprogramming with a fixed/variable number of tasks (MFT/MVT)
- MULTICS (1965, MIT, AT&T Bell Labs, GE)
  - MULTiplexed Information and Computing Service
- Unix (1969, AT&T Bell Labs)

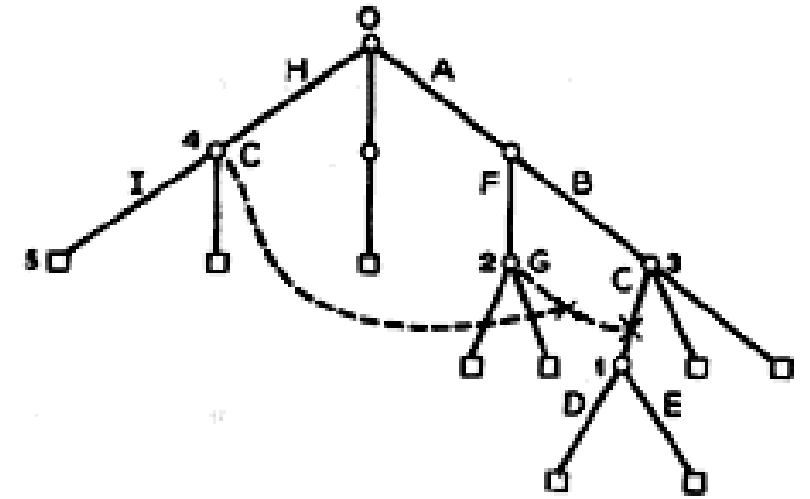
# Multics

- Multiplexed Information and Computing Service
- A time-shared, multi-processor mainframe “computing facility”
- Originally started by MIT, GE, and Bell Labs in 1965
  - For GE-645, a 36-bit system
  - Bell Labs quit in 1969 and built Unix
  - GE’s computer business, including Multics, was taken over by Honeywell in 1970
  - Last system shutdown on 10/31/2000
- <http://www.multicians.org>

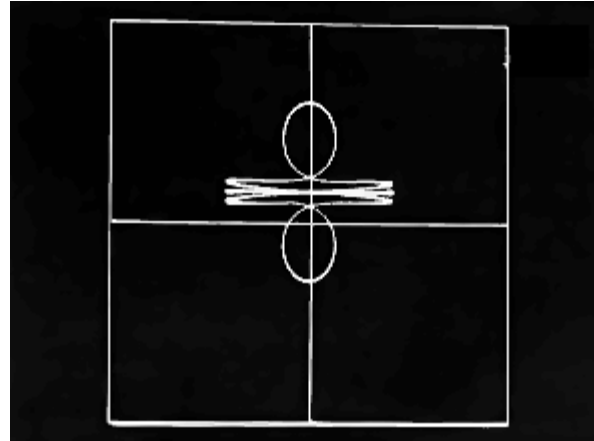


# Multics Innovations

- Hierarchical file system
  - ACLs, long names, hard & symbolic links, quota, ...
- Virtual memory (segmentation and paging)
- User-level command shell
- Dynamic linking, shared memory
- Implementation in high-level language (PL/I)
- Mapping of logical disk volumes onto physical volumes
- Support for BCPL, APL, Fortran, Lisp, C, Cobol, Algol, Pascal, ...
- Multics Relational Data Store (MRDS), Spreadsheets
- Rated B2 by NCSC (National Computer Security Center)



# Unix



“... When BTL (Bell Telephone Laboratories) withdrew from the Multics project, they needed to rewrite an operating system in order to play **space travel** on another smaller machine (a DEC PDP-7 with 4K memory for user programs). The result was a system which a punning colleague called UNICS (UNIplicated Information and Computing Services) – an ‘emasculated Multics’; no one recalls whose idea the change to UNIX was.”

– Peter H. Salus, *A Quarter Century of Unix*, Addison-Wesley, 1994.

“... It was the summer of '69. In fact, my wife went on vacation to my family's place in California.... I allocated a week each to the operating system, the shell, the editor, and the assembler, to reproduce itself, and during the month she was gone, it was totally rewritten in a form that looked like an operating system, with tools that were sort of known, you know, assembler, editor, and shell .... Yeh, essentially one person for a month.”

– Ken Thompson

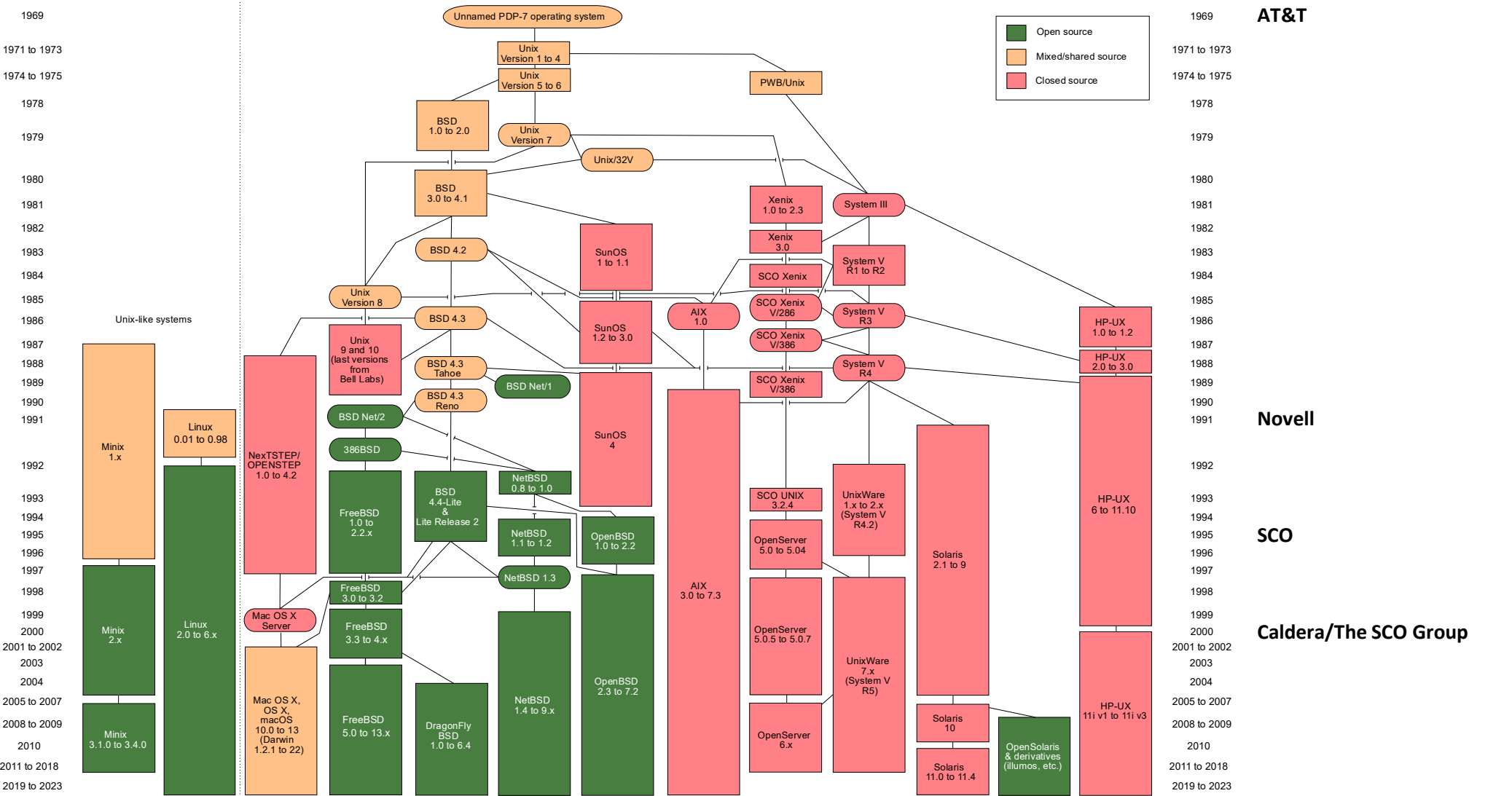
# Unix Features

- Hierarchical file system
  - Special files: uniform I/O, naming, and protection
  - Removable file systems via mount/umount
  - i-node
- Process control
  - `fork()`, `exec()`, `wait()`, `exit()`
  - Pipes for inter-process communication
- Shells
  - Standard I/O and I/O redirection
  - Filters, command separators, shell scripts
- Signals



AT&T Archives: The UNIX Operating System  
<https://www.youtube.com/watch?v=tc4ROCJYbm0>

# Unix Family Tree



# Multics vs. Unix

## ■ Multics

- Top-down approach
- 150 MY for design and system programming, another 50 MY for improvements
- Too complicated, too costly hardware
- Many novel ideas had a great impact

## ■ Unix

- Bottom-up approach
- 2 MY: simplicity, elegance, and ease of use
- Low-cost hardware, university adoption
- The root of the modern operating systems

# OS: Application View

- OS provides an execution environment for running programs
- OS provides a(an) \_\_\_\_\_ view of the underlying computer system
  - What are the correct abstractions?
  - How much of hardware should be exposed?
- Typical OS abstractions
  - Processors → Processes, Threads
  - Memory → Address space (virtual memory)
  - Storage → Volumes, Directories, Files
  - I/O Devices → Files (+ ioctls)
  - Networks → Files (sockets, pipes, ...)



# OS: System View

- OS manages various resources of a computer system

- Sharing

- \_\_\_\_\_

- Fairness

- Efficiency

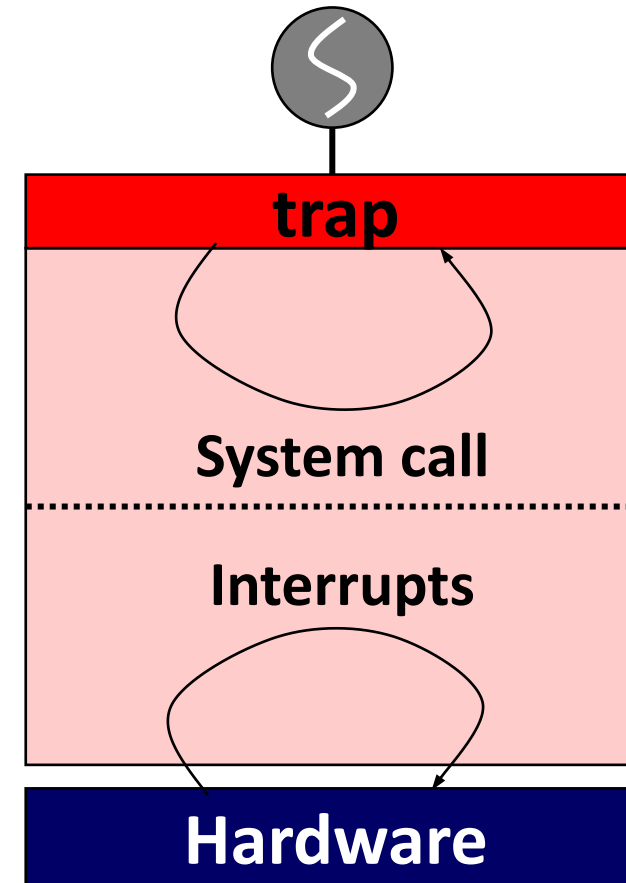
- ...

## Resources

- CPU
- Memory
- I/O devices
- Queues
- Energy
- ...

# OS: Implementation View

- OS is highly-concurrent, \_\_\_\_\_ software
- Two kinds of events
  - System calls
  - Interrupts



# Three Pieces

- Virtualization

- How to make each application believe it has each resource to itself?

- Concurrency

- How to handle concurrent events correctly and efficiently?

- Persistence

- How to make information survive power loss?