OS History and Overview

• Stallings, 2.1 – 2.10

• Section 2.11 is on Android. Important information, but it might be distracting/confusing at this point. First, I want to show you how a general-purpose operating system works.

Main Topics

• Outline:
  – Origins of operating systems
  – Major developments and advances
  – Brief history and key players
    • Multics to Unix to Linux
    • Mach microkernel
    • Windows evolution
    • Apple’s path to MacOS X
  – Preview of virtualization technology
Operating System

- One definition: A program that controls the execution of application programs and acts as an interface between user and the computer hardware
  - OS must be able to relinquish control to user programs and regain it safely and efficiently
  - Decides when and how to execute user programs
  - Manages/allocates hardware and system software resources to applications.

But, electronic computers preceded operating systems...

Origins of Operating Systems

- Serial Processing
- Simple Batch Systems
- Multiprogrammed Batch Systems
- Time Sharing Systems
Serial Processing

- Earliest electronic “com-put-ers” did not have an OS
  - programmers interacted directly with the computer hardware
- Computers ran from a console with display lights, toggle switches, some form of input device, and a printer
  - Load compiler, program (via card reader and tapes)
  - Compile, link, run, in steps
  - Errors indicated by lights on front panel
- Users had access to the computer in “series” (sign-up sheet!)

ENIAC (1946)

- First general-purpose electronic computer
- Developed by UPenn for Army Research Lab
- Project led by
  - John Mauchley (1907-1980)
  - J. Presper Eckert (1919-1995)
- Input: card reader
- Output: card punch; data printed by an accounting machine
- Used vacuum tubes – transistors not invented yet
- Stored data in decimal format
- 357 10-digit multiplications/sec, 35 divisions/sec
Batch Processing

• Much time was being lost configuring a computer between one job and the next (setup and teardown time)

• The earliest “operating systems” were designed to address this problem
  – batches of jobs loaded onto tape, executed sequentially by the OS
  – first OS: GM Research Labs developed OS for IBM 701

A major change in thinking…

• Still, the idea of programmers interacting directly with the machine seemed impractical, even silly.

• Why?

• Timesharing
  – a.k.a., multiprogramming, multitasking
  – Sharing a resource (e.g., computer) among users, such that each seems to have the whole machine
  – Pauses by one user filled with work for another
  – Need to store program state, quickly switch among apps.
  – Basic idea: “Virtualize” the CPU
  – Key mechanism?
Multics Project

- 1966, joint project between MIT, GE, and Bell Labs
- Idea: design a timesharing computer system that could be accessed via telephone from anywhere
  - written mostly in PL/1 (not assembly language!)
  - implemented on GE 645, which supported paged-segmentation memory hardware.
  - virtual address: 18-bit segment number and 16-bit word offset (large!)
- Last known installation shut down in 2000!

Key Players

- Fernando Corbato (1926-2019)
  - MIT professor (retired)
  - Turing Award 1990

- Jack B. Dennis (1931 - )
  - MIT professor (retired), Member NAE

- Peter Denning (1942 - )
  - PhD, MIT
  - Prof at Purdue, then NASA, GMU, NPS…

My OS professor in grad school, 1982
Multics Features/Innovations

• Virtual address spaced merged into tree-like file system; each segment was a file
• Per process stacks
• Concept of a shell, a process you can “replace” with some other process
• Multi-level feedback queue scheduling
• Security: access list associated with each file and set of protection rings of services for executing processes
• Dynamic linking of library routines (on demand)
• Supported multiple CPUs

Multics Ring Structure

• Protected domains numbered 0 to 7
• Processes executing in inner domains have more privileges than those in outer domains
• Separate (kernel) stack for each ring
Moving among Rings

• Each process maintains current-ring-counter, indicating ring in which it is currently executing
• Segmented address space; each segment has ring number and access bits (rwx)
• Process in ring i can access any segment at ring j, where j ≥ i, according to access bits and access control list
• Mechanism to move to lower ring?

Unintended Multics Contribution: Unix!

• Ken Thompson and Dennis Ritchie were Bell Labs engineers who worked on the Multics Project
• They (and others) felt Multics was overly complex and had too many layers…
• Began work on a simpler operating system…
  – OS comprises one layer
  – All application code (end applications, libraries, shells) run in another layer
Unix Beginnings

- 1969 - first Unix system
  - (mostly) Thompson at Bell Labs
  - implemented in assembly language on DEC PDP-7
  (what else happened in 1969?)

- 1970 - development of C
  - Thompson and Dennis Ritchie port Unix to a PDP-11/20
  - Ritchie designs and writes the first C compiler
  - Goal: “high-level” language for writing a portable OS

- 1972 - C implementation of Unix
  - Ritchie and Thompson rewrite the Unix kernel in C.

BTW, PDP-11…
Unix Features

• Monolithic kernel (simple!)
  – Clean design (originally)
  – No complex recovery scheme

• Designed for programmers
  – Many system calls, libraries
  – Shells, scripts, standard I/O
  – Fork/exec, pipes, signals, IPC
  – Windowing with X11 (later)
  – Tree structured file system
  – Simple file access/descriptors
  – And many others…

AT&T’s Unix

• 1970s - free distribution of Unix
  – UNIX source code distributed freely to universities, due to marketing restrictions on AT&T, parent of Bell Labs
  – UNIX gains favor within the academic/research community

• 1984 - AT&T Unix goes commercial
  – January 1, 1984 - Divestiture of Bell System, which had previously been a government-regulated monopoly
  – AT&T can enter new markets, including computers
  – AT&T releases the commercial UNIX System V
BSD Unix

• 1979-1980 - Berkeley enters the picture
  – Ken Thompson had spent a sabbatical at Berkeley, among other things, teaching Unix
  – Bill Joy, a graduate student, becomes a disciple
  – Joy invents vi, adds demand paging to Unix - 3BSD
  – A DARPA research grant supports the implementation of TCP/IP and the Unix interface, sockets

• Berkeley Software Distribution (BSD Unix)
  – Most influential BSD versions will be 4.2 (1983) and 4.3 (1987), widely used in computer science departments
  – Joy will co-found Sun Microsystems,
  – BSD Unix will form the basis of many commercial spinoffs

Unix System V

• 1987 – AT&T releases UNIX System V Release 3
  – Major features include: IPC facilities, Transport Level Interface, Remote File Sharing, STREAMS communication facility
  – Various major hardware vendors, who felt business pressure to have a “Unix” product, based their OS on this version. Examples: HP-UX IBM AIX.

• Late 1980s - AT&T and Sun Microsystems agree to cooperate on UNIX development to merge/unify System V and BSD lines.

• 1990 - System V, Release 4 (AT&T and Sun)
  – Intended as a new standard unifying the UNIX variants.
Unix Wars

• Other vendors (especially DEC, HP, IBM) are threatened by the idea of AT&T and Sun “cornering” the Unix market
• They unite to create the Open Software Foundation (OSF) and an alternative Unix standard
• 1991 - OSF/1, an OS that combines BSD Unix and Mach microkernel, is released
• Nearly all the main computer companies were now marketing some version of Unix…at a price!
• But… Unix had become popular because it was a powerful OS, and free!!

GNU (GNU’s Not Unix!)

• In response to commercialization of Unix, Richard Stallman starts the GNU Project
  – Goal: create “complete Unix-compatible software system” composed entirely of free software
  – 1985 - Stallman creates Free Software Foundation
  – 1989 - Stallman writes GNU General Public License
    • Copyleft license: requires derived works to be available under the same copyleft.
  – By early 1990’s, many programs (libraries, shells, etc) available, but kernel work was stalled.
Linux

- Andrew Tanenbaum (prof in Netherlands) had developed MINIX as a “microkernel-based, Unix-like” OS for education.
- 1991 - Linus Torvalds develops Linux kernel as a student project
  - Based in part on MINIX and Unix System V
  - Wanted a free Unix-like, industrial strength OS (note: google(Tanenbaum-Torvalds debate) on monolithic vs microkernels)
  - Originally developed only for Intel x86 hardware; since then, has been ported to more platforms than any other OS

Linux Impact

- Linux is available under GNU Public License
- Many, many distributions of Linux, packaged with various “userland” configurations, have been available over the past 20 years
- Torvalds continues to oversee kernel work, and Stallman still runs Free Software Foundation
- Linux continues to evolve:
  - multithreaded kernel, demand-loadable modules
  - Support for 64-bit processors
  - Ever-expanding set of modules, new file systems
  - And many others...
Linux Impact

• Although desktop OS market is dominated by Windows versions, Linux has major share of other areas:
  – Top 10 fastest supercomputers run Linux
  – 95% of top 500 supercomputers run Linux
  – over 86% of smartphones sold in world used Android
  – 33% of server market (65% for Unix overall)

So, Monolithic Kernels are Best. Right?

• Unix/Linux has had great success, due to:
  – Simple design (at least originally)
  – Portability (most of OS is machine independent)
  – Economics (became widespread because it was free)
  – It’s continued evolution
  – And, it is F-A-S-T!

• Disadvantages of a monolithic kernel?
Mach Project

- Some researchers considered a monolithic kernel to be unwieldy and inherently insecure/brittle.
- Mach Project at Carnegie Mellon (Prof. Rick Rashid)
  - Started in 1986, but built on earlier projects
  - Features: microkernel, Unix compatibility, better IPC, SMP support, new approach to virtual memory

- The advent of microkernels would unleash a bitter 20-year debate on how to build operating systems

Monolithic vs. Microkernel-based OS

Microkernel

- Very small kernel provides only the most basic services to support processes and threads
  - Low-level process management, memory management, interprocess communication, interrupt handling
- Servers (often running at user-level) provide all other OS functions
  - Full process management, virtual memory management and paging, file service, I/O, networking, system calls and system emulation
  - User process sends message to server, which does work and returns results

Emulation of Other Operating Systems

- Mach microkernel intended to support other OS’s, which are emulated at user level
- Multiple emulators could be running simultaneously, e.g., BSD Unix, Windows, etc. (sound familiar?)
Microkernel vs. Monolithic Kernel?

• Advantages of microkernel approach?

• Main disadvantage?

How to Make $90B

• 1975 - Altair personal computer marketed by MITS of Albuquerque, NM;
  – popular among hobbyists; most run CP/M Operating System. Who wrote version of BASIC for it?

• 1980 - IBM (surprisingly!) joins PC market
  – IBM executives approach Bill Gates for an OS. Gates points them to Digital Research Corp. for CP/M OS

• Digital Research personnel balk!!

• Gates buys a CP/M-like OS for $30K, hires its author, sells it as MS-DOS to IBM for $50K !!*

  * reserves right to sell it to other PC manufacturers…
MS-DOS

- 1981 - MS-DOS version 1.0 on 5.25’ floppy disk
- 1984 - version 2.0 - still in assembler, but incorporated many Unix concepts (tree-based file system, file access routines, redirection…)
- 1985 - first release of Windows, a graphical shell for MS-DOS. Modeled after Apple Lisa, which Jobs & Wozniak developed with ideas from Xerox Parc.
- Early releases of Windows are not popular
- 1990 - Windows 3.0 released – Major Success!
- 1991 - Windows sold in stores for first time

Windows NT

- Despite the popularity of Windows, Bill Gates realized that the underlying MS-DOS was not a long-term solution
- 1988 – Gates hires Dave Cutler from DEC to design a new OS for PCs
- Original design heavily influenced by microkernel research of Rick Rashid (joined MS in 1991)
  - designed from scratch, written mostly in C
  - inherent thread & multiprocessor support
  - object oriented design (though not written in C++)
Original NT Architecture

- NT supports multiple types of processes (POSIX, Win32, OS/2) via user-level **subsystems**

Windows NT Evolution

- Although the name was changed for marketing reasons, NT is the foundation of:
  - Windows 2000, XP, Vista, 7, 8, 10, ...
- Performance was lacking in the original system
  - Trampoline issue with user-level servers
- NT has evolved to be similar to a monolithic OS
  - Subsystems and other user-level code has migrated to kernel-level
Rounding out the BIG THREE: Mac OS X

- For years, Apple and Microsoft were not considered players in “serious” computing (including the OS)
- That changed at Microsoft with NT
- At Apple, it was Mac OS X
- To understand why/how OS X was developed, we need to consider several factors:
  - Apple design philosophy
  - Foundational technologies developed at other places
  - Competition from other companies (e.g., Microsoft)
  - “Failed” projects within Apple

Birth of Apple

- 1976: Steve Wozniak, Steve Jobs and Ronald Wayne founded Apple Computer
- 1976: Apple I introduced, $666.66
  - Based on prototype developed by Wozniak
  - Fully assembled circuit board, as opposed to a kit
  - Systems were hand-built by Wozniak
  - 8-bit 6502 processor, 4KB RAM
  - Firmware-resident monitor (OS)
  - Intended for hobbyists, much less expensive than a Unix time-sharing system (already in its 6th edition)
Apple II

- 1977 – Apple II introduced
- Also based on 6502 processor
- 16KB RAM, completely assembled, games included
- First PC with color graphics, major impact on personal computing
- Apple II had a very long and successful history
  - Production ended only in 1993!

Apple Macintosh

- Jobs unveils Macintosh on January 24, 1984
  - Superbowl pre-announcement commercial two days earlier
  - http://www.youtube.com/watch?v=2zfqw8nhUwA
- 7.83 MHz MC68000, but no MMU and no FPU
- 128K RAM, 3.5” floppy drive
- Mouse, 512x342 B&W 9” display
- Mac OS: still primitive like MS-DOS
  - Single user, single tasking
  - But with graphical display
- A more sophisticated OS for Mac was still many, many years away…
NeXT Chapter

• 1985: Steve Jobs had left Apple with 5 other employees and forms a startup, NeXT Computer.
• NeXT Cube (1988)
• NeXTSTEP OS
  – Kernel: Mach 2.0
  – 4.3BSD “environment”
  – Both graphical and command-line interfaces
  – Always-on “dock” to hold frequently used applications
  – Written in Objective-C, which was inspired by Smalltalk

Mac OS X

• 1996 Apple acquires NeXT!
  – Access to Mach, Objective-C, and everything NeXTSTEP
• 2000 - Jobs again Apple CEO
• New OS development project
  – Main components all inherited from NeXT
  – XNU kernel merges Mach and BSD Unix, plus OO driver framework, in a single address space OS
• Multiple OS interfaces
  – BSD Unix, Classic (MacOS), Carbon (NeXTSTEP)
Original Mac OS X Architecture

- Lower entities support higher ones
- XNU kernel comprises BSD "atop" Mach microkernel
- XNU kernel resembles?

OS Challenges in 1980s and 1990s

- How can one OS support multiple types of processes?
  - Mach microkernel plus user-level emulators
  - Windows NT plus subsystems
  - Mac OS X programming interfaces
  - Problems?
- Fault tolerance and load balancing
  - enable running applications to be paused, checkpointed and migrated among computer systems
  - Difficulties?
Virtual Machines

• Why not just host the full, unmodified OS (and its applications) running on a virtual machine?

• Recall: A regular OS “fakes out” user-level processes
  – makes each one think it has the whole machine
  – virtualizes the processor through time slicing
  – virtualizes memory through page tables

• But how can we fake out a whole operating system?
  – reads and writes all sorts of critical data structures
  – contains special instructions that need to be executed in kernel mode

Virtualization is not new

• Virtual machines were implemented on IBM mainframes way back in the 1960s

• The idea fell out of use with the advent of the PC
  – “If I have my own physical machine, why do I need a virtual machine on a mainframe?”
  – In fact, given certain change in the semantics of instructions, it became “impossible” to virtualize Intel x86 processors…

• “Rediscovery” of virtual machines began with VMware in late 1990s.
“Hosted” Virtual Machines

- Host OS serves applications as usual
- **Unaware** that another piece of kernel code (hypervisor) periodically takes control of the machine and supports multiple VMs
- Any idea how the hypervisor can grab control of the machine?

### VM Ubiquity

- Nowadays, there are many companies offering various types of VMs, as well as support from HW vendors.
- Virtualization is arguably **the** most important development in OS in the past 20 years.
- Cloud Computing would be **very** different without it.
- Details later in the course…
- But first, we need to learn how a “regular” OS works!
- So, let’s get started with Chapter 3, Processes.