CA170: Week 9

What is an Operating System

Intro

- List of operating systems
- Comparison of operating systems

What is an Operating System?

1. Provide an environment in which programs and users can work.
    - Extract problems that are common to multiple programs and users on the system, and provide the solutions to them, so that programs and users do not have to.
        - e.g. The programmer writing a program does not need to know about the physical layout of the hard disk
        - he wants to be able to work with a concept called "files" (as does the user).
    - Not wanting programmers to have to reinvent the wheel each time:
        - File systems
            - Where does this file go on the physical disk? What if it grows bigger than the slot initially assigned to it?
        - User interfaces
            - If I move a window, how do I re-draw the window underneath? What pixels do I set to draw the letter 'A'?
    - These are not questions to be solved by each programmer.
    - They have other things to do. These are questions for the OS.
    - Network utilities - provide API for applications to use.
    - Bundling utilities so that users have many/most programs they need already installed. Editor, browser, calculator, image editor. Could go on.
2. Clever OS algorithms squeeze the most usage out of slow machines and limited resources
    - Wanting to run multiple programs on one machine, ideally overlapping in time. e.g. A web browser and a text editor. Take it for granted.
    - Wanting multiple users to be able to share one machine at the same time (web server, ssh to mainframe).
3. HIDE the reality of the system from the programs and users.
    - Hide the fact that my program is scattered all over memory, is being moved around in memory, and is being swapped out to disk.
    - Hide the fact that my file is physically scattered all over the disk, and is being moved around the disk.
    - Hide the fact that my program shares memory with lots of running system programs and multiple other user's programs, all of which are being moved around.
    - Hide the fact that my program is not actually running constantly, but is actually getting little timeslices of the CPU, in between it going off to service other programs.

Principles that an OS will work by

1. Respond to the user interface as quick as possible. Everything else can wait.
2. Only load into memory the absolute minimum that we need to work with. We can load more if and when needed.
    - For example, just load the initial code to display a program and its menus. Don't load the code to handle specific menu items until they are called (which often never happens).
3. Only write to disk the absolute minimum we need to make the changes.
    - Example: Editing a 5 M file. Added a few lines to the end. Press "Save".
4. When deleting large areas of memory or disk, just put markers at the start of block and end of block showing it is free to be overwritten if and when needed in the future. There is no need to actually go through every location and scrub it clean.
    - Example: Delete a 10 G file.
    - See un-deleting files.

"Never Enough"

- The whole history of Operating Systems / Computers can be summarised as:
- "Never enough".
- Never enough CPU speed.
- Never enough memory (RAM).
- Never enough disk space. (Perhaps solved for PCs. 10 T now less than $300.)
- Never enough disk space on reliable, redundant, always-on, disk-always-spinning, high-use server. (Much more expensive. 10 T costs around $10,000.)
- Never fast enough disk access.
- Never enough screen size.
- Never enough bandwidth.
- Never enough battery life time.

PC Operating Systems

PC Operating Systems (Single User)

- Microsoft dominance. Been that way for a long time.
    - Usage share of operating systems - link in notes
    - Desktop and laptop computers - link in notes
-
- Desktop Operating System Market Share 2020 by netmarketshare.com.
- Do one month span only to get bar chart.
- Microsoft dominance, but not quite monopoly.
-
- Desktop Operating System Market Share 2020.
- From StatCounter.

PC Operating Systems

- DOS / Windows - link in notes
- Mac OS X - link in notes
- UNIX / Linux - link in notes
- Historical
    - OS/2 - link in notes
    - Mac OS - link in notes
- Comparison of operating systems - - link in notes
- Comparison of operating system kernels - link in notes
- Comparison of open-source operating systems - link in notes

Server Operating Systems

Mainframe / Server operating systems (multi-user)

- Split between UNIX/Linux and Microsoft. Been that way for a long time.
- Usage share of operating systems
    - Servers
    - Mainframes
- Usage of operating systems for websites 2020.
- Usage share of server operating systems for web servers only (not all servers/mainframes).
- "Unix" family includes Linux.
- IDC's Worldwide Quarterly Server Tracker tracks all servers (not just web servers)
    - Pay to view, so hard to link to.
    - In 2014 this had server OS share (by percentage of revenue):
        - Windows 45.7%
        - Linux 28.5%
        - Unix 13.6%
        - z/OS 8.0%
    - Anything more recent on a public link?

Server operating systems

- mainframes / servers (multi-user),
- file servers (shared filespace in LAN),
- web servers
- UNIX / Linux
- Windows Server
- z/OS
- VM
- VMS

Supercomputers

- Supercomputer operating systems
- TOP500 list
- Dominated by Linux.
- As at 2020, top 10 are all Linux.

Mobile operating systems

Mobile operating systems

- A formerly diverse market has settled into Android v. iOS.
- Market share listings may not agree because of differences in recording "OS of phones" v. "OS of phones that access websites"
- Usage share of operating systems
    - Mobile
-
- Sales stats show a formerly diverse market ...
-
- ... changing to Android dominance.
-
- Mobile Operating System Market Share 2020 by netmarketshare.com.
-
- Mobile Operating System Market Share 2020.
- From StatCounter.
-

Mobile operating systems

- Mobile operating systems (smartphone, PDA, tablet)
- Comparison of mobile operating systems
- iOS (Apple, closed source, UNIX family)
- Android (Google, open source, Linux, UNIX family)

App stores

- App stores
- Comparison of mobile software distribution platforms
- App Store (iOS)
- Google Play (Android)
- Tablet Operating System Market Share 2020 by netmarketshare.com.

Tablets

- Tablets
- Usage share of operating systems
- Tablets
- iOS and Android split the market
- i.e. Unix family dominates the market

History Of Operating Systems

History of Operating Systems

- History of Operating Systems
- Timeline of Operating Systems

(1) In 1940s-50s, Program = Computer

- At start, there were no Operating Systems.
- Computers were like the abstract model of a machine,
- running one program literally and nothing else.
- 1 user at a time
- 1 program at a time
- Programmer operates machine himself.
- Manually load program, run until crashed (dump memory and registers) or finished (print output), revise program, run again, or run next program.
- Interactive (great if you're the lone programmer) but CPU idle for long periods (e.g. while program being revised, or when program halts/crashes when programmer not watching).
- Long wait to use machine for other programmers.
- DRIVING FORCES FOR CHANGE:
- LOT MORE PROGRAMMERS WANTING TO USE MACHINE
- COMPUTERS EXPENSIVE (ANY CPU IDLE TIME BAD)

(2) In 1950s-60s, Operator hired to run computer.

- Programmers have to submit jobs, receive results later.
- Operator schedules jobs.
- Jobs still stored on sequential (tape) access (no random access medium yet).
- Sequential tapes of jobs are prepared by operator for the CPU to run through once.
- Resident monitor, always in memory (first OS).
- Doesn't decide order of jobs, but does co-ordinate their sequencing, automatic starting and terminating.
- Jobs come with control cards to say what they need to run.
- Downside for programmer: long queues, not interactive any more,
- if error in program may have to wait days to retry it.
- Have to think things through in advance!
- DRIVING FORCE FOR CHANGE:
- RANDOM (DISK) ACCESS BECOMES AVAILABLE.

(3) Pool of jobs on disk, random access.

- OS can now implement the human operator's algorithms in deciding sequence of jobs to run.
- Scheduling of jobs now totally automated (true OS).
- DRIVING FORCE FOR CHANGE:
- I/O DEVICE SPEED IS MUCH SLOWER THAN CPU SPEED,
- SO CPU STILL OFTEN IDLE WHILE I/O GOING ON.

(4) Some parallelisation of I/O and computation.

- Device controller is a piece of hardware that can work in parallel with the CPU.
- Not a parallel computer
- - it is just a specialised device for data transfer, not a general-purpose computer.

- Spooling - Print device controller is still copying data from disk to printer
- while CPU has already begun working on next job.
- Next job can begin while previous is still printing out.
- CPU can store output on disk if printer fulland move on.
- DRIVING FORCES FOR CHANGE:
- WAIT TIMES STILL TOO LONG.
- Long jobs delay everything else. Run short jobs first?
- But if we have a small job that must be run once a day, then we cannot EVER run a program that will take 1.1 days.
- Also, program might do I/O half-way through its execution (rather than only at start/end).
- When program stops to wait for this I/O, CPU is idle.
- Maybe only short time, but it all adds up.

(5) Multi-programming.

- Multiple jobs in memory (and running) at the same time.
- Each memory space protected from each other.
- OS picks one, executes it for a while, stops (e.g. when program waits for input, or else randomly), picks other to run.
- CPU busy, but still batch model, not interactive.
- Scheduling:
- Job scheduling (or long-term scheduling)
- - which jobs get into memory at all
- CPU scheduling (or short-term scheduling)
- - which to run at each step (many may be ready)
- DRIVING FORCES FOR CHANGE:
- INCREASING NUMBER OF -USERS- WHO WANT TO
- INTERACT WITH PROGRAMS WHILE THEY ARE RUNNING.
- COMPUTERS CHEAPER - CHEAP DUMB TERMINALS AVAILABLE.
- HUMANS' TIME IS EXPENSIVE - DON'T WANT THEM TO WAIT

(6) 1970s-80s. Interactive time-sharing on mainframes

- Multi-programming where the program may be waiting on a USER.
- OS will in the meantime service another program, which may be interacting with another user.
- Result: Multiple users share the CPU.
- If the time-slicing is quick enough, they all feel as if they have their own dedicated machine!
- Programmers delighted - CPU kept busy, yet interactive again, just like in (1).
- User interaction at run-time allows a whole world of programs that were never possible before.
- In practice shared systems can get overloaded and slow down.
- DEC VT100 terminal (1978).
- This kind of terminal would be used to run programs on a mainframe shared with other users.
- Users can now interact with programs at run-time.
- DRIVING FORCES FOR CHANGE:
- REAL COMPUTERS GET VERY CHEAP

(7) 1980s. Standalone PCs.

- To some extent a return to simplicity of (5) and earlier
- - no traffic problems.
-
- Internet still unimportant.

- LANs beginning to be used to coordinate file sharing.
-
- Solitaire, bundled with Windows from 1990.
- - How office workers wasted time on PCs before the Internet.
- DRIVING FORCES FOR CHANGE:
- INTERNET BECOMES USABLE AND USEFUL.
- STANDALONE MACHINE NOW SEEN AS TOO ISOLATED.

(8) 1990s. Internet.

- Web is killer app for Internet 1993.
- Much use of information on shared remote web servers.
- Shared file systems.
- Shared email systems.
- Return to some of the problems of (6) - traffic problems.
- The shared mainframe returns:
- The Web.
-
- DRIVING FORCES FOR CHANGE:
- BROADBAND (AT HOME).
- THEN LATER: MOBILE BROADBAND.

(9) 2000s

- Multimedia - video, audio, photo.
- YouTube (2005) probably the defining application of 2000-10 period.
- Mobile - Internet on mobile devices.
- Proper OSes (with proper file systems) on mobile devices.
- iPhone (2007) probably the defining invention of 2000-10 period
-
- The decline of dialup, 2006-08 (in Ireland).
-
- The mobile world before 2005:
- No or limited Internet.
- Image from here.
- **video of steve jobs
- Steve Jobs introduces the iPhone at MacWorld 2007.
- The modern mobile world begins.
- Click through to video.
- Go to 5:20 where he introduces the modern concept of touch.