Software systems, buzzwords, issues

- **operating systems**
  - runs programs, controls the computer, stores information, communicates
- **applications ("apps")**
  - programs that do things
- **cloud computing, virtual machines, ...**
  - where boundaries become even less clear

- **intellectual property**
  - copyrights, patents, licenses
- **interfaces, standards, antitrust, ...**
  - agreements on how to communicate and inter-operate
- **open source software**
  - freely available, non-proprietary
- **jurisdiction**
  - where are the computers? where is the data? who has access to it?

Programs come in lots of sizes

- **programs come in different sizes**
  - cos109 psets and labs: tiny, like a response paragraph
  - 10-20 lines
- **projects in COS courses like 333**
  - like a term paper
  - 2000-5000 lines
- **significant applications**
  - like a book, maybe a very big book
  - 100,000 – 10,000,000 lines
- **operating systems, major applications**
  - like a multi-volume book ?
  - 10,000,000 and up

- **a typical programmer produces at most a few thousand lines of production code per year**

Operating system

- **a program that controls the resources of a computer**
  - interface between hardware and all other software
  - examples: DOS, Windows 3.0/3.1/95/98/NT/ME/2000/XP/Vista/7/8/10
    Unix/Linux, Mac OS X, iOS, Android, ...

- **runs other programs ("applications", your programs, ...)**
- **manages information on disk (file system)**
- **controls peripheral devices, communicates with outside world**
- **keeps things from interfering with each other**

- **provides a level of abstraction above the raw hardware**
  - makes the hardware appear to provide higher-level services than it really does
  - makes programming much easier

What an operating system does

- **manages CPUs, schedules and coordinates running programs**
  - switches CPU among programs that are actually computing
  - suspends programs that are waiting for something (e.g., disk, network)
  - keeps individual programs from hogging resources
- **manages memory (RAM)**
  - loads programs in memory so they can run
  - swaps them to disk and back if there isn't enough RAM (virtual memory)
  - keeps separate programs from interfering with each other
  - and with the operating system itself (protection)
- **manages and coordinates input/output to devices**
  - disks, display, keyboard, mouse, network, ...
  - keeps separate uses of shared devices from interfering with each other
  - provides uniform interface to disparate devices
- **manages files on disk (file system)**
  - provides hierarchy of directories and files for storing information

History of general-purpose operating systems

- **1950's: signup sheets**
- **1960's: batch operating systems**
  - operators running batches of jobs
  - OS/360 (IBM)
- **1970's: time-sharing**
  - simultaneous access for multiple users
  - Unix (Bell Labs; Ken Thompson & Dennis Ritchie)
- **1980's: personal computers, single user systems**
  - DOS, Windows, MacOS, Unix
- **1990's: personal computers, PDA's, …**
  - PalmOS, Windows CE, Unix / Linux
- **2000's: Windows, Unix/Linux, MacOSX (a Unix variant)**
- **2010's: Apple vs. Google vs. Microsoft**
  - iOS, Android, Chrome-OS, … (all Unix/Linux-based)
  - cloud computing
- **not all computers have general-purpose operating systems**
  - "embedded systems": small, specialized, but increasingly general (often Unix/Linux)

Unix operating system

- **developed ~1971 at Bell Labs**
  - by Ken Thompson and Dennis Ritchie
- **clean, elegant design**
  - at least in the early days
- **efficient, robust, easy to adapt, fun**
  - widely adopted in universities, spread from there
- **written in C, so easily ported to new machines**
  - runs on everything (not just PC's)

- **influence**
  - languages, tools, de facto standard environment
  - enabled workstation hardware business (e.g., Sun Microsystems)
  - supports a lot of Internet services and infrastructure
    - often Linux

Linux

- **a version of Unix written from scratch**
  - by Linus Torvalds, Finnish student (started 1991)

- **source code freely available (kernel.org)**
  - large group of volunteers making contributions
  - anyone can modify it, fix bugs, add features
  - Torvalds approves, sets standard
  - commercial versions make money by packaging and support,
    not by selling the code itself

- **used by many major sites, including**
  - Google, Amazon, Facebook, Twitter, YouTube, ABC, CBS, CNN, ...

To run programs, the operating system must

- **fetch program to be run (usually from disk)**
- **load it into RAM**
  - maybe only part, with more loaded as it runs (dynamic libraries)
- **transfer control to it**
- **provide services to it while it runs**
  - reading and writing info on disk
  - communications with other devices
- **regain control and recover resources when program is finished**
- **protect itself from errant program behavior**
- **share memory & other resources among multiple programs running "at the same time"**
  - manage memory, disks, network, ...
  - protect programs from each other
  - manage allocation of CPUs among multiple activities

Memory management

- **what's in memory? over-simplified pictures:**

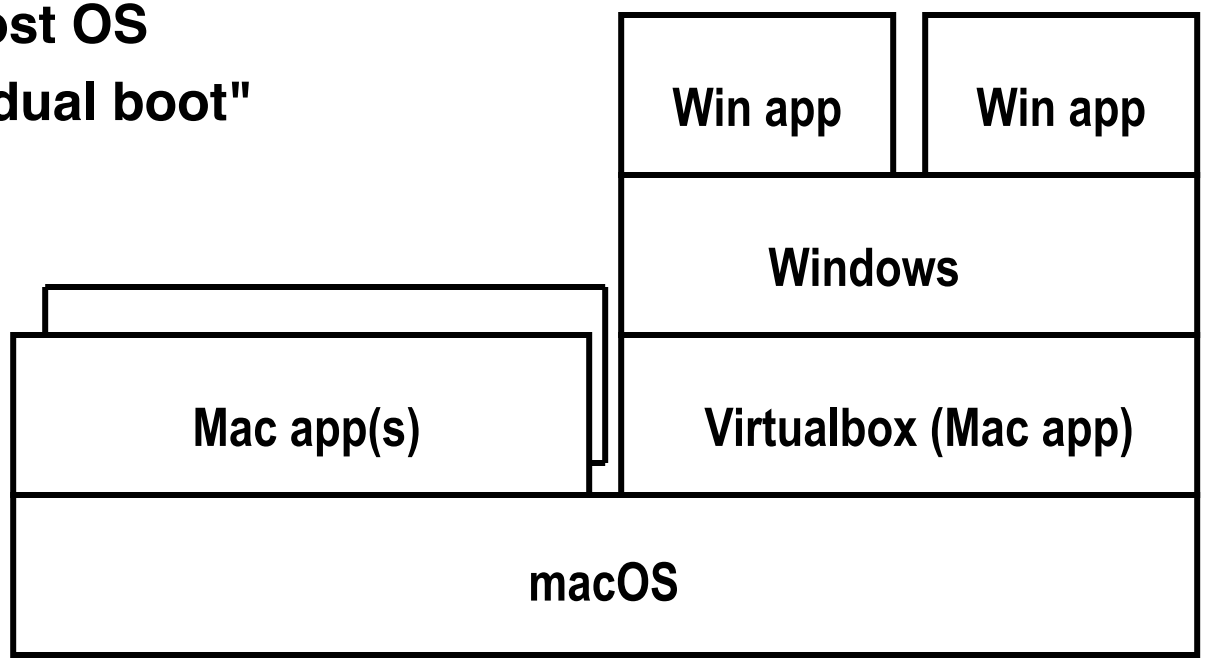
Unix:

Windows:

- **reality is more complicated**
  - pieces of programs are partly in RAM, partly on disk
    can only execute instructions that are in RAM
- **memory protection:**
  - making sure that one program can't damage another or the OS
- **virtual memory:**
  - making it look like there is more RAM than there really is

Virtual machines

- **running other operating systems on top of an OS**
  - e.g., VMWare, VirtualBox, Xen, HyperV, ...
- **system calls from applications to "guest" OS are intercepted by "host" OS**
  - e.g., guest == Windows 10 or Linux, host == macOS
- **passed to guest OS, which handles them by converting into system calls to host OS**
- **not the same as "dual boot"**

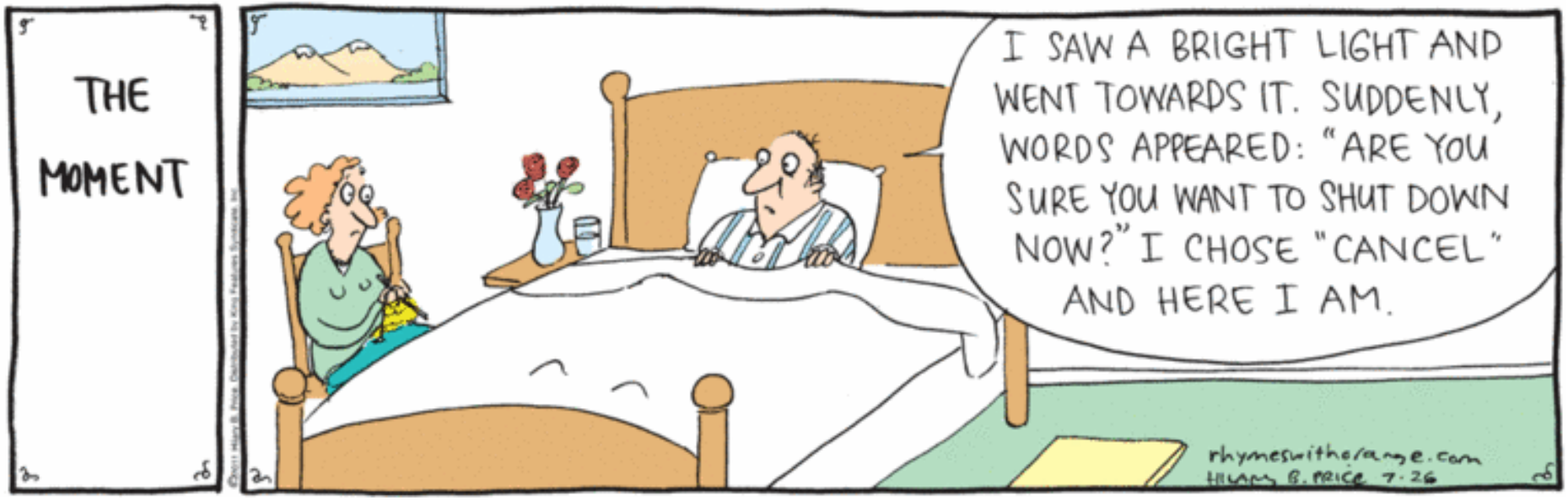


Bootstrapping: how does it all get started?

- **when turned on, user sees screen turn on, desktop appear, mouse/keyboard come alive**
- **underneath, CPU begins executing at a specific memory location when turned on**
  - location is defined by the hardware: part of machine's design
  - often in ROM (read-only memor) so not volatile but changeable

- **"bootstrap" instructions place there read more instructions**
  - CPU tries to read first block from disk, which has instructions to read more of the operating system
  - if that fails, tries to read bootstrap instructions from somewhere else

    e.g., USB drive, network, ...


THE MOMENT
I SAW A BRIGHT LIGHT AND WENT TOWARDS IT. SUDDENLY, WORDS APPEARED: "ARE YOU SURE YOU WANT TO SHUT DOWN NOW?" I CHOSE "CANCEL" AND HERE I AM.
rhymeswithorange.com
HILARY B. PRICE 7-26