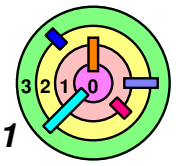
# Ch 1: Introduction

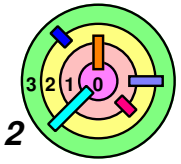
Bill Cheng

*http://merlot.usc.edu/cs402-s16*



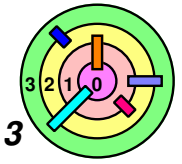
# What are Operating Systems?

- **Possible definitions:**
  - **the code that {Microsoft, Apple, Linus, Google} provides**
  - **the code that you didn't write**
  - **the code that runs in privileged mode**
  - **the code that makes things work**
  - **the code that makes things crash**
  - **etc.**



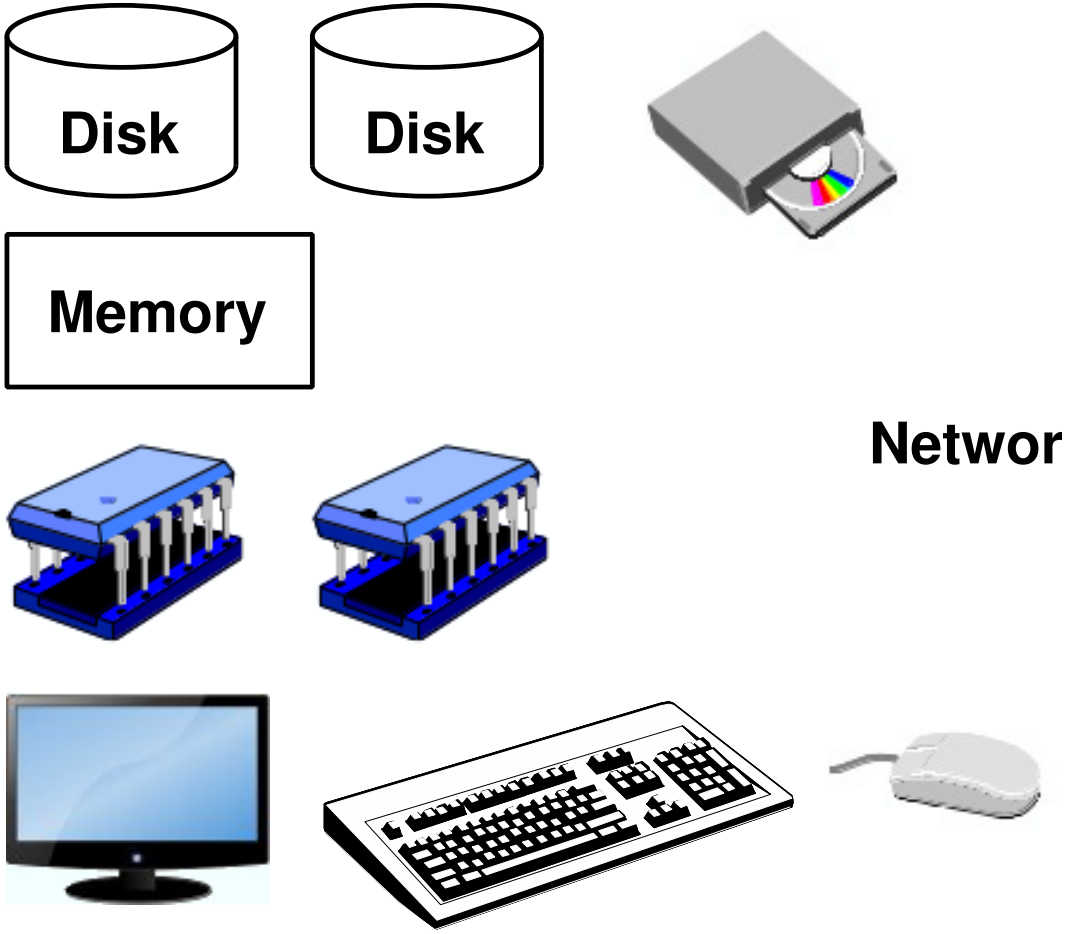
# Operating Systems

- **Abstraction**
  - providing an "appropriate" interface for applications
  - but abstraction to what? (next slide)
- **Concerns**
  - performance
    - time, space, energy
  - sharing and resource management
  - failure tolerance
  - security
  - marketability



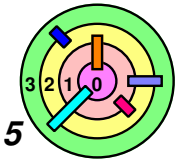
# Hardware

- Hardware
  - disks
    - hard drives
    - optical drives
  - memory
  - processors
  - network
    - ethernet
    - modem
  - monitor
  - keyboard
  - mouse

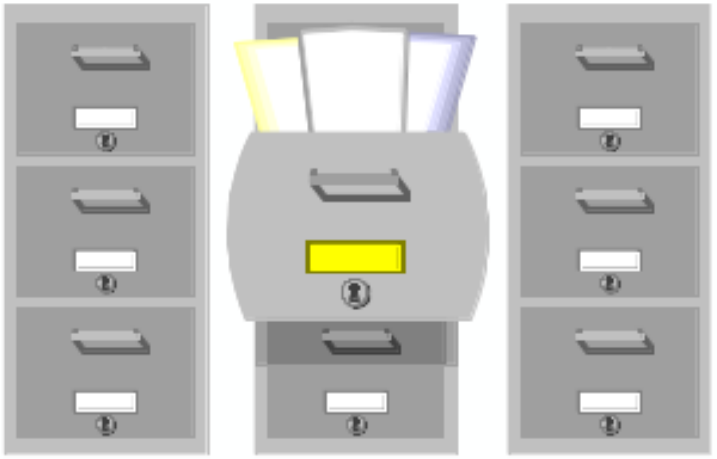


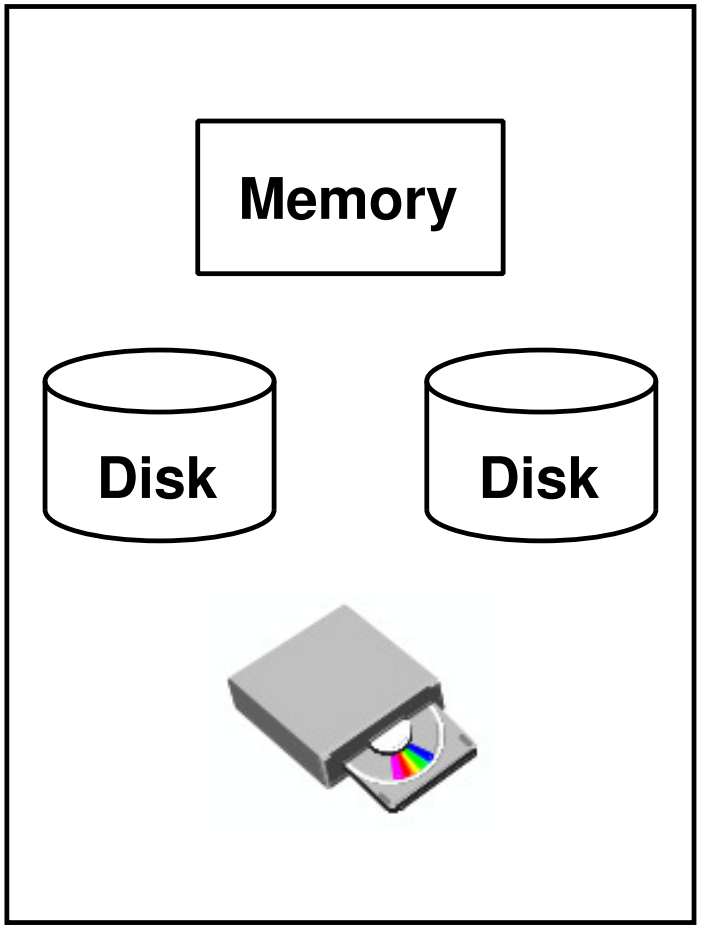
# OS Abstractions

- Hardware
  - disks
  - memory
  - processors
  - network
  - monitor
  - keyboard
  - mouse
- Operating system
  - *files (file system)*
  - *programs (processes)*
  - *threads of control*
  - communication
  - windows, graphics
  - *input*
  - locator
- For those who knows about "processes", we use the word "program" to mean "process" in the introductory material



# Abstraction Example: Files

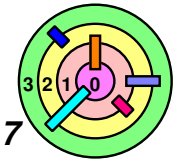





- It's nice to have a simple abstraction
- Abstraction did not come for free
  - it introduces problems that need to be solved and issues to be addressed

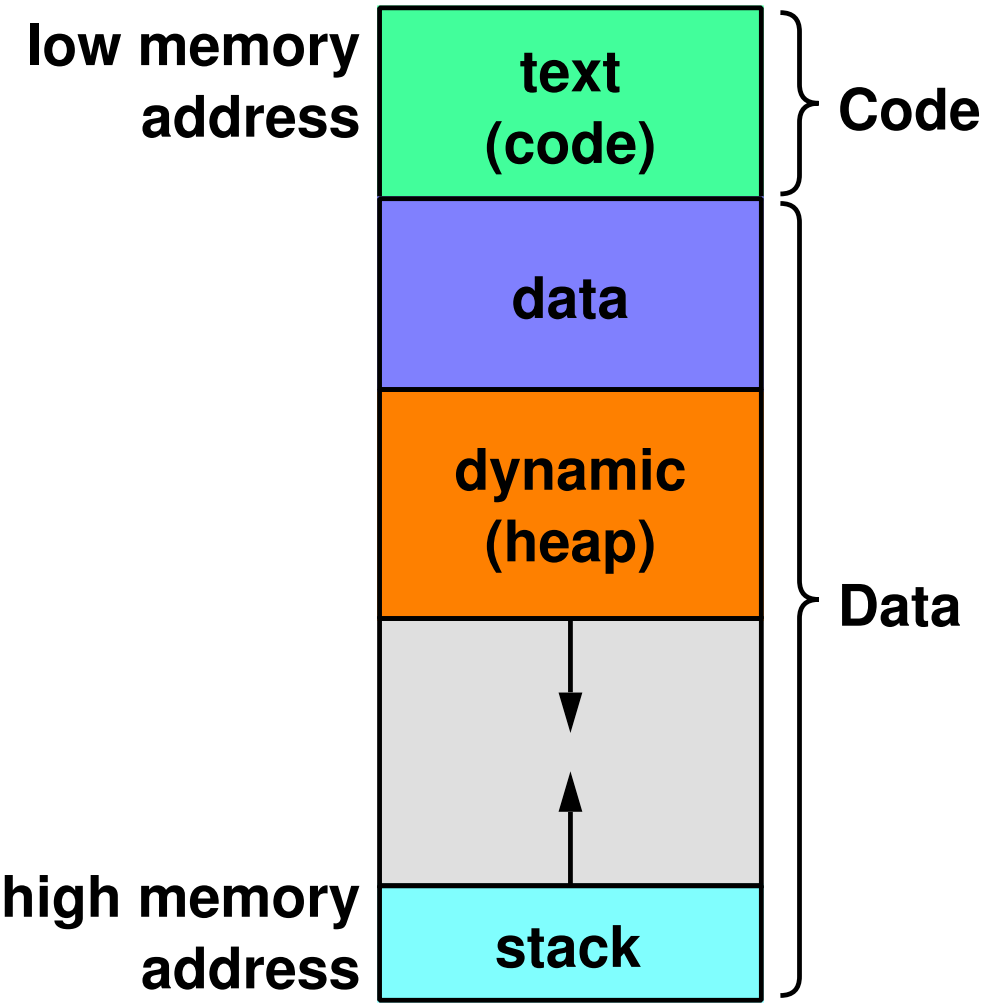
# Issues With The Files Abstraction

- Naming
  - device-independence
- Allocating space on disk (permanent storage)
  - organized for fast access
  - minimize waste
- Shuffling data between disk and memory (high-speed temporary storage)
- Coping with crashes



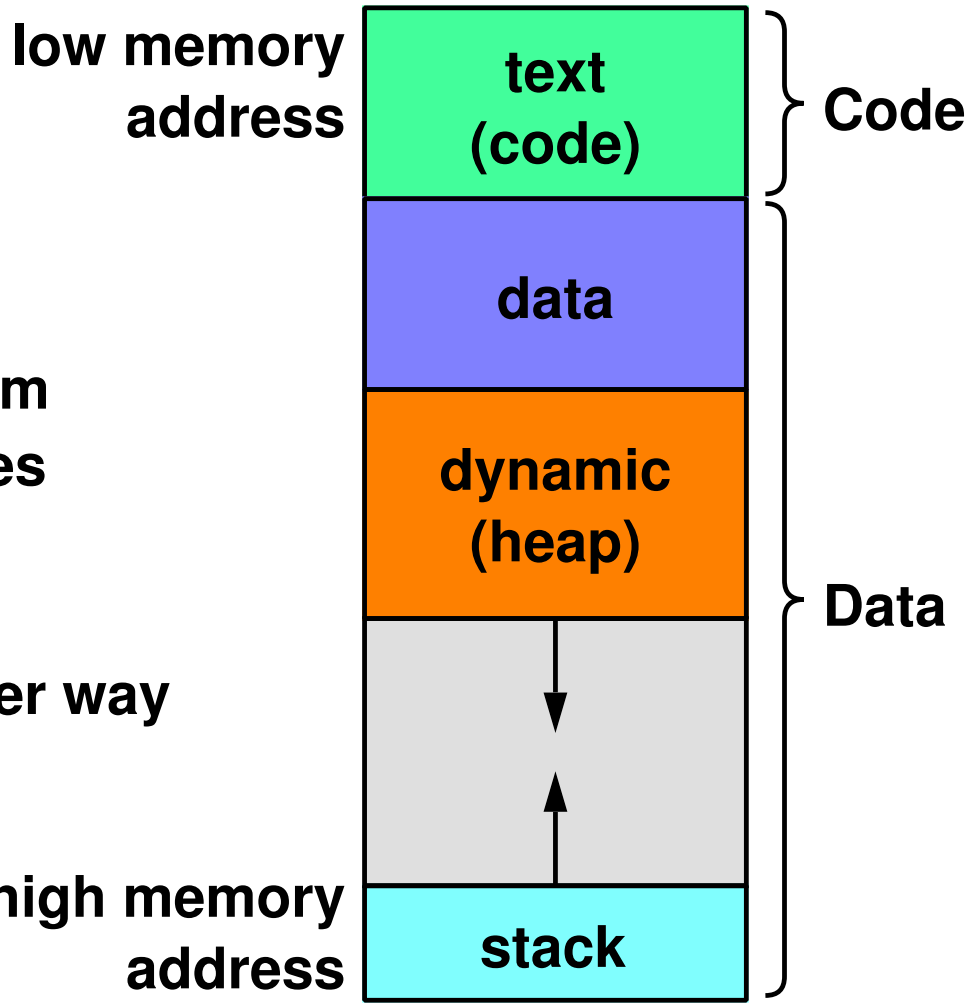
# Abstraction Example: Programs

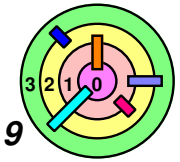
- Application programmers use the ***Address Space*** abstraction:



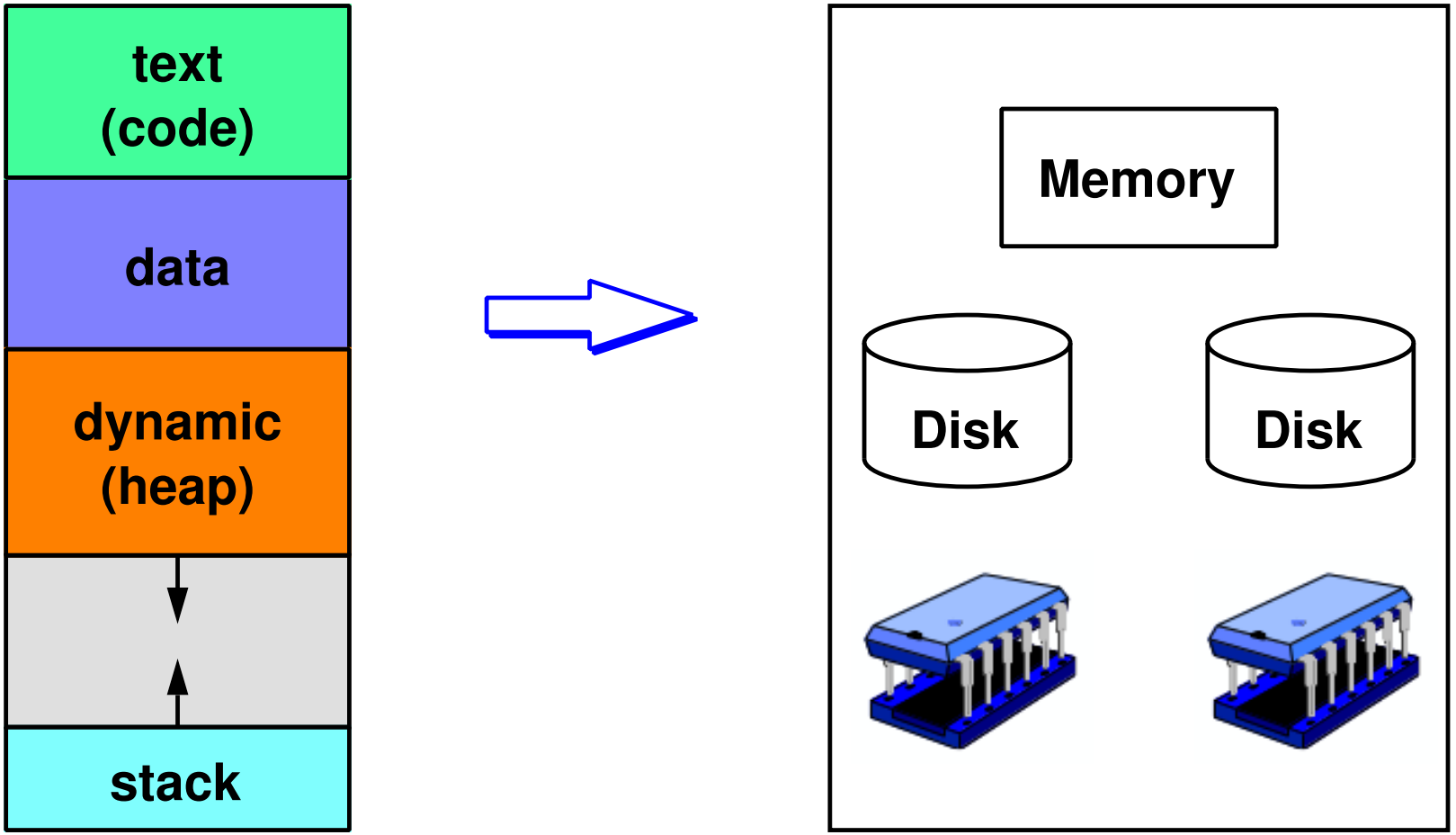
# Abstraction Example: Programs

- Application programmers use the *Address Space* abstraction:
- *Very important:*
  - our address space is *up-side-down* (compared with the textbook)
    - low address at the top
    - high address at the bottom
      - memory layout matches an array
    - stack looks like a "stack"
  - our textbook does it the other way

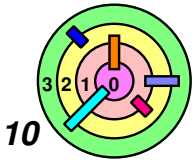


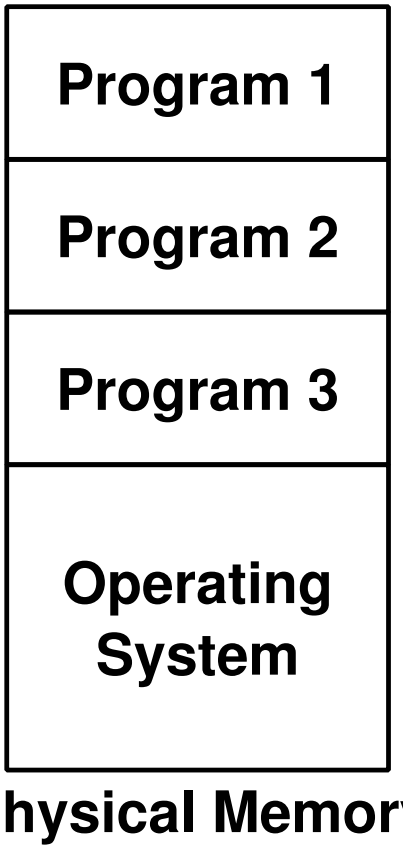
# Abstraction Example: Programs



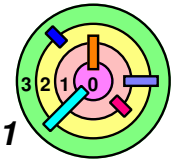

Application programmers do not have to worry about any *sharing* that's going on



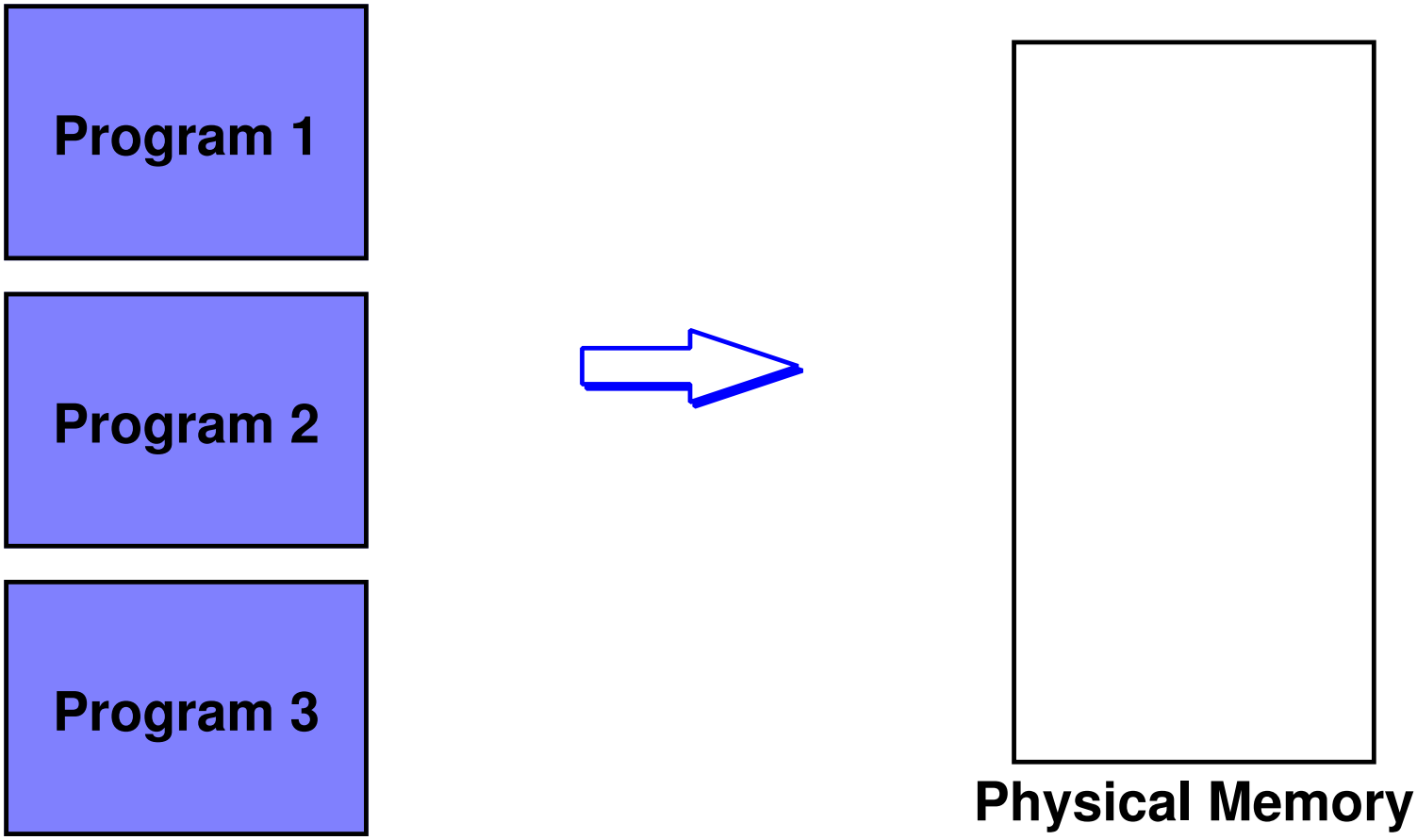
# Memory Sharing Option 1



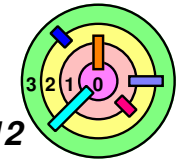

**Does not appear to be very flexible**



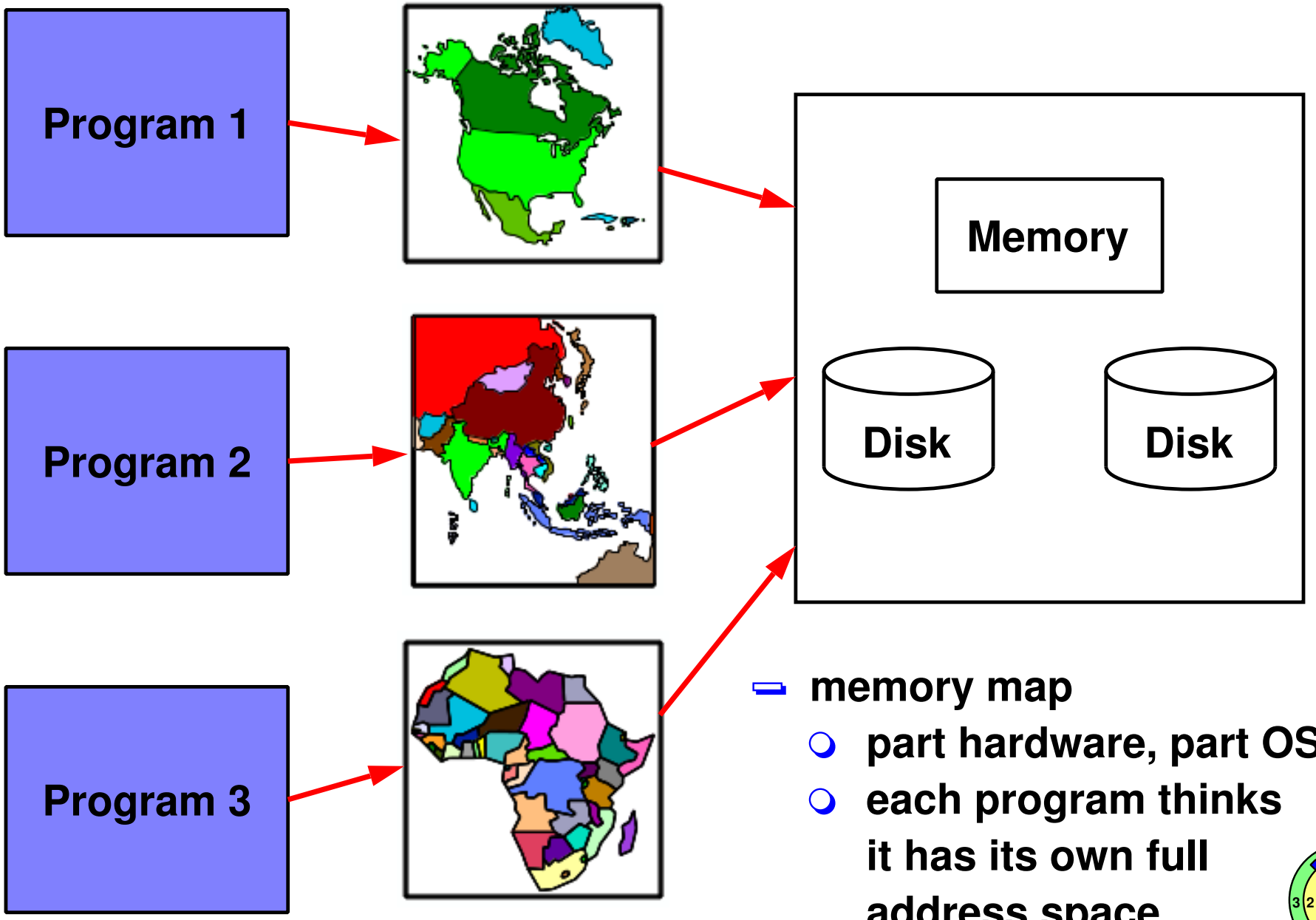
# Memory Sharing Option 2




What if programs take up too much space (more than physical memory)?

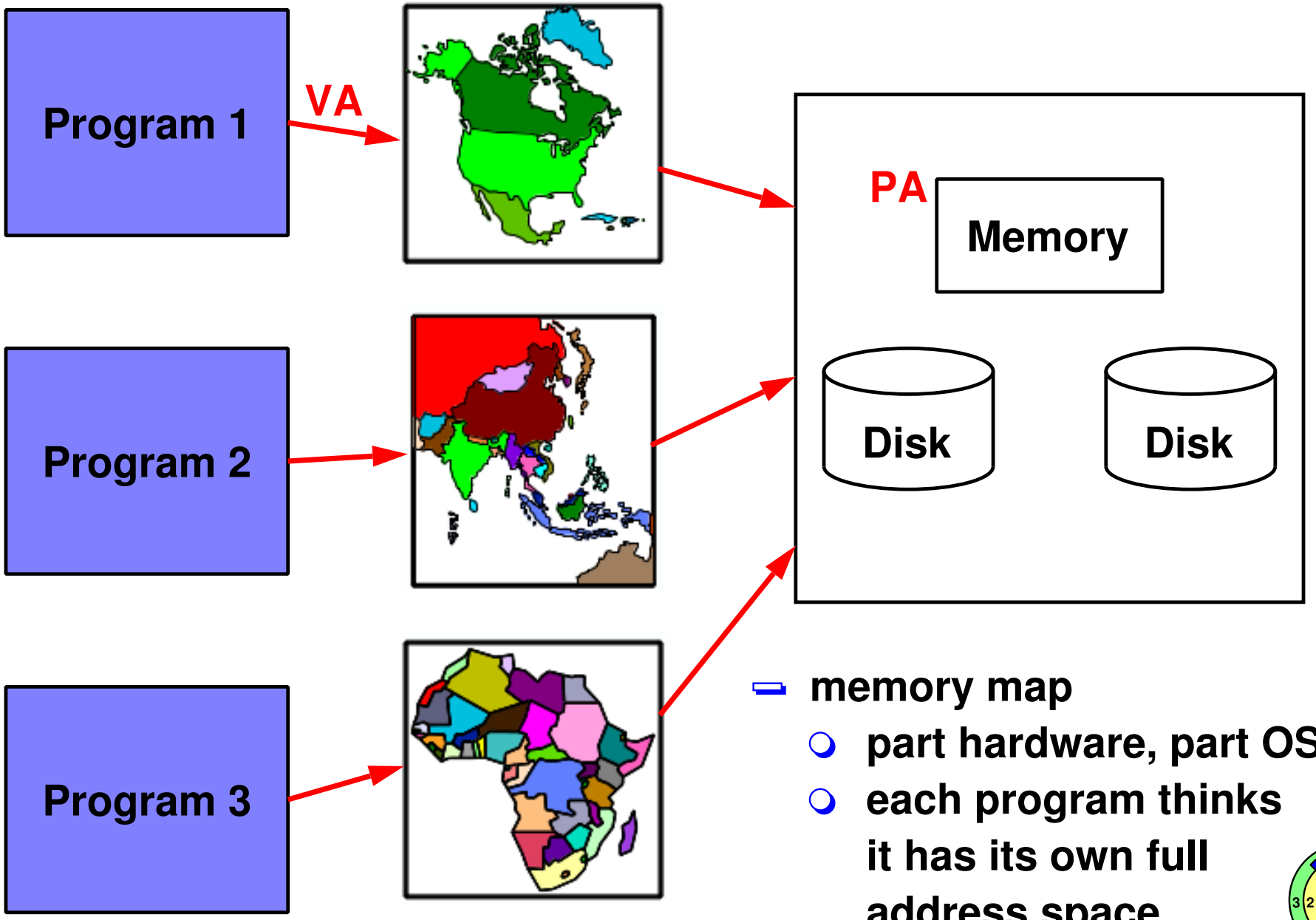


# Virtual Memory




- memory map
  - part hardware, part OS
  - each program thinks it has its own full address space

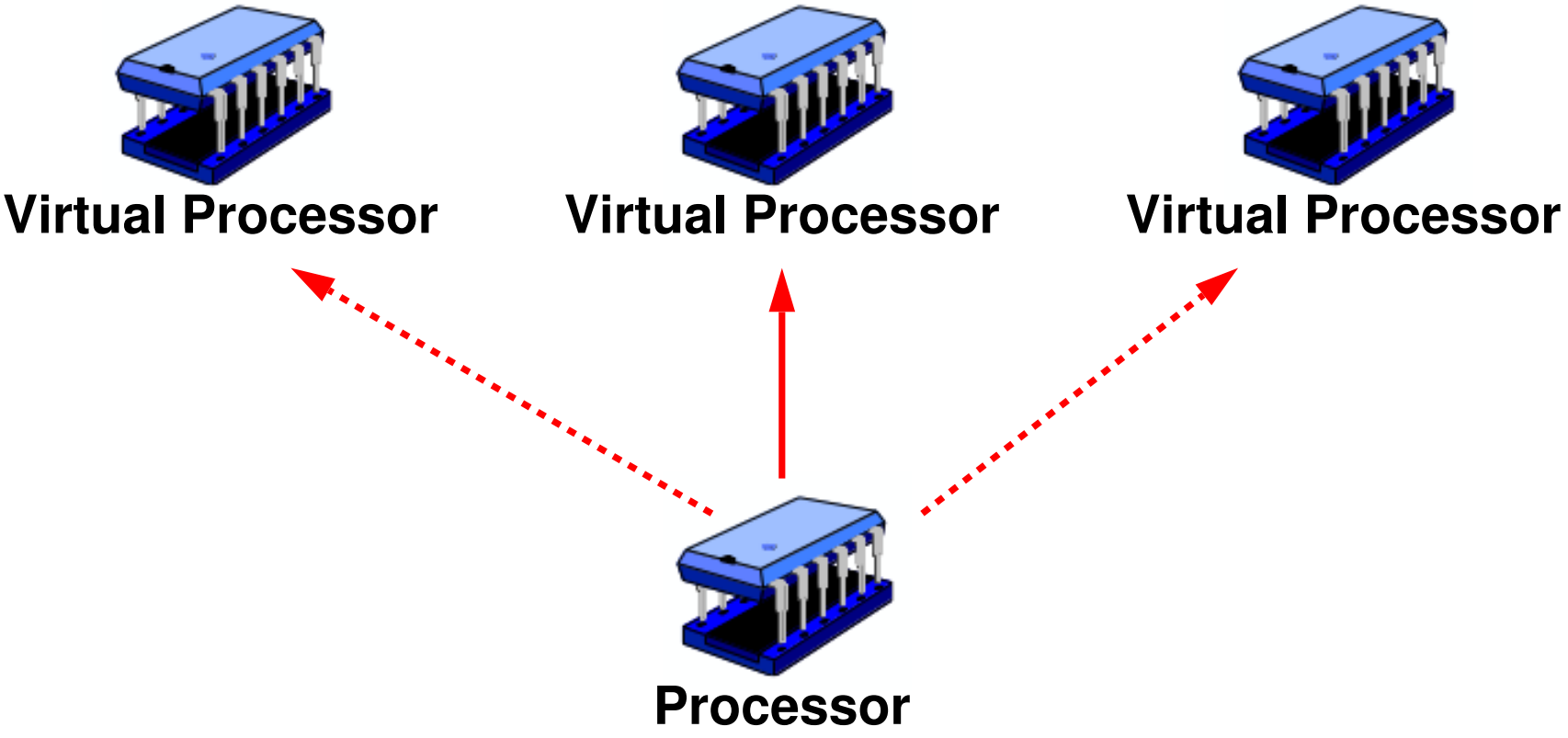
# Virtual Memory




- memory map
  - part hardware, part OS
  - each program thinks it has its own full address space

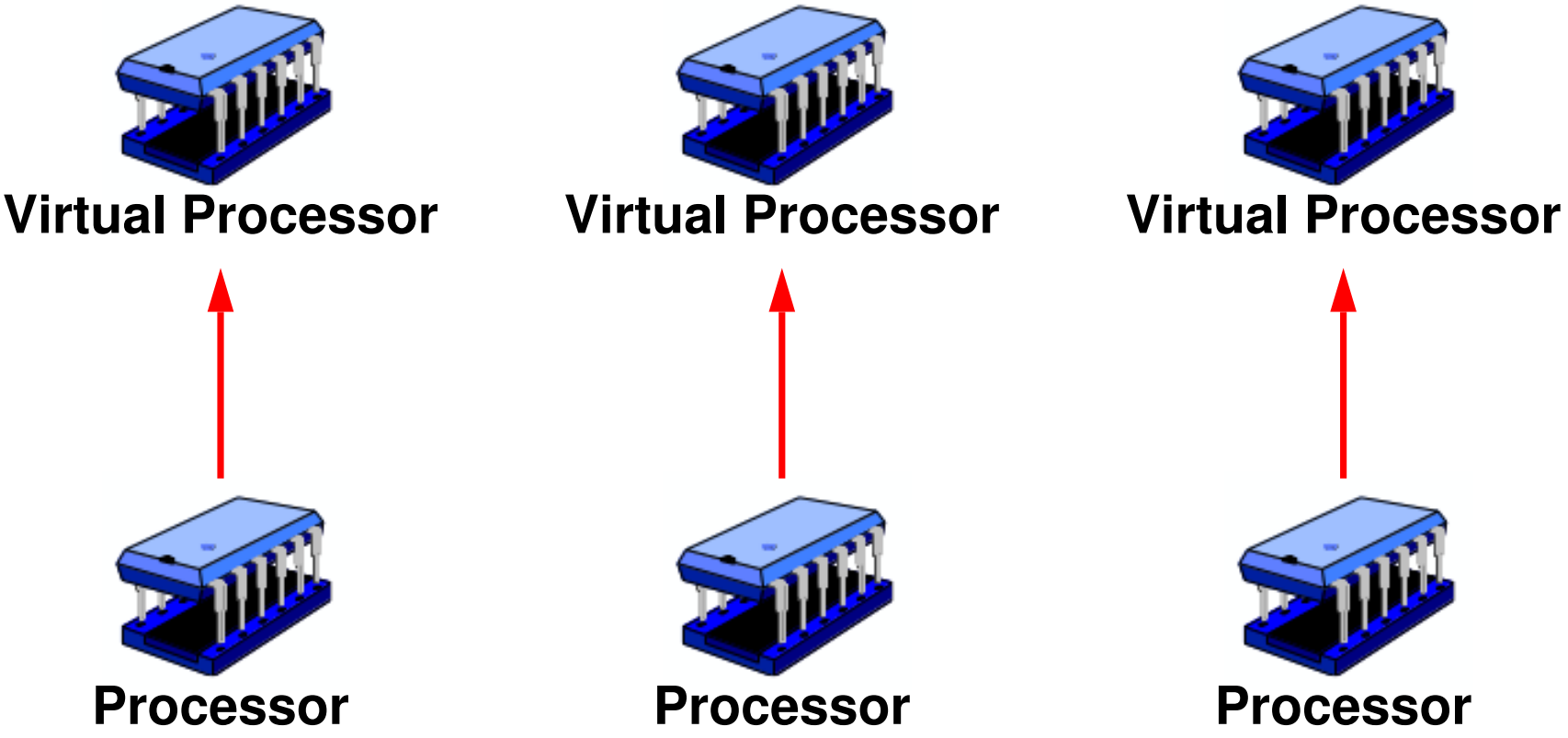
# Sharing of Processor: Concurrency

- If you only have one processor, how do you run multiple "programs" and every program thinks it owns the processor?
  - abstraction: threads (or "threads of execution")



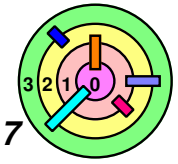
# Sharing of Processors: Parallelism

- What if you have a multicore processor or multiple processors?
  - we don't distinguish the two cases
  - can still use threads
    - but we need to worry about how well we do resource (processor) management/allocation



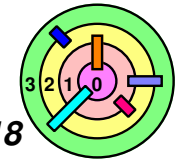
# 1960s OS Issues

- Multiprogramming (i.e., running things "in parallel" with one CPU)
- Time sharing (i.e., support interactive users)
- Software complexity
- Security



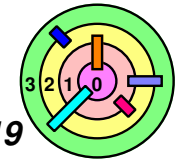
# 2010s OS Issues

- Multiprogramming (i.e., running things "in parallel" with one CPU)
  - not just one computer, but server farms
- Time sharing (i.e., support interactive users)
  - voice, video, sound, etc.
- Software complexity
  - a bigger problem than could be imagined in the 1960s
- Security
  - ditto



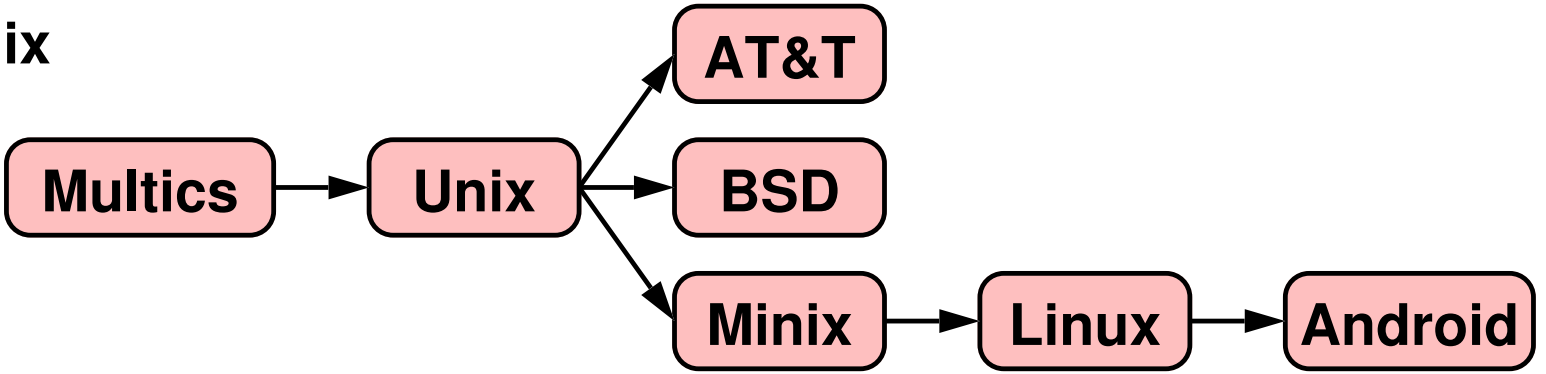
# 1.2 A Brief History of Operating Systems

- The 1950's: The Birth of the Concept
- The 1980's: The Modern OS Takes Form
- Minicomputers & Unix
- The Personal Computer

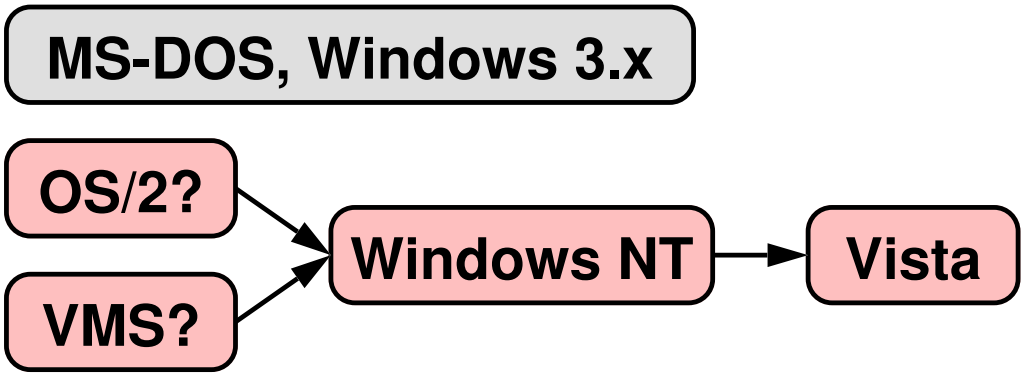


# Where Do Things Evolve From?

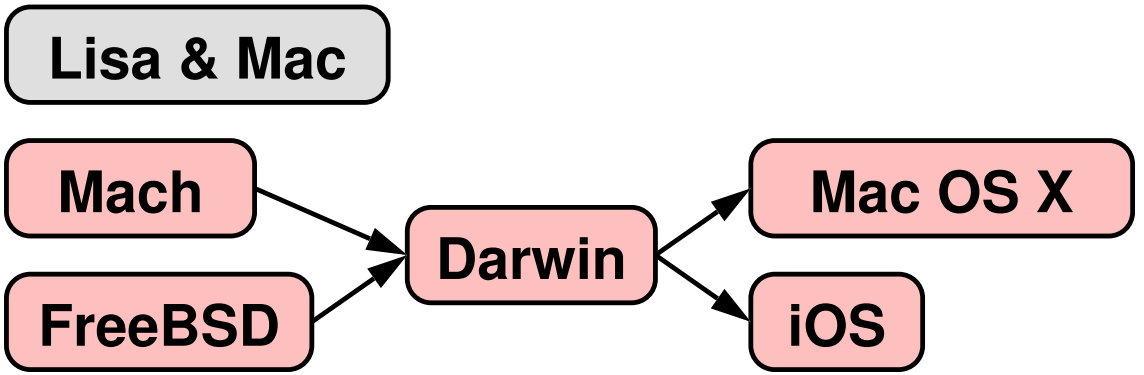
- Linux & Unix

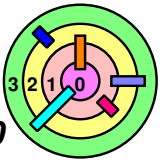



- Microsoft



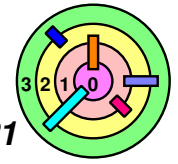

- Apple



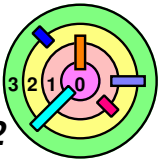



# History of C

- Early 1960s: CPL (Combined Programming Language)
  - developed at Cambridge University and University of London
- 1966: BCPL (Basic CPL): simplified CPL
  - intended for systems programming
- 1969: B: simplified BCPL (stripped down so its compiler would run on minicomputer)
  - used to implement earliest Unix
- Early 1970s: C: expanded from B
  - motivation: they wanted to play "Space Travel" on minicomputer
  - used to implement all subsequent Unix OSes
- Unix has been written in C ever since

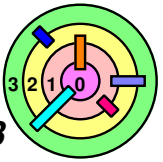


# Extra Slides

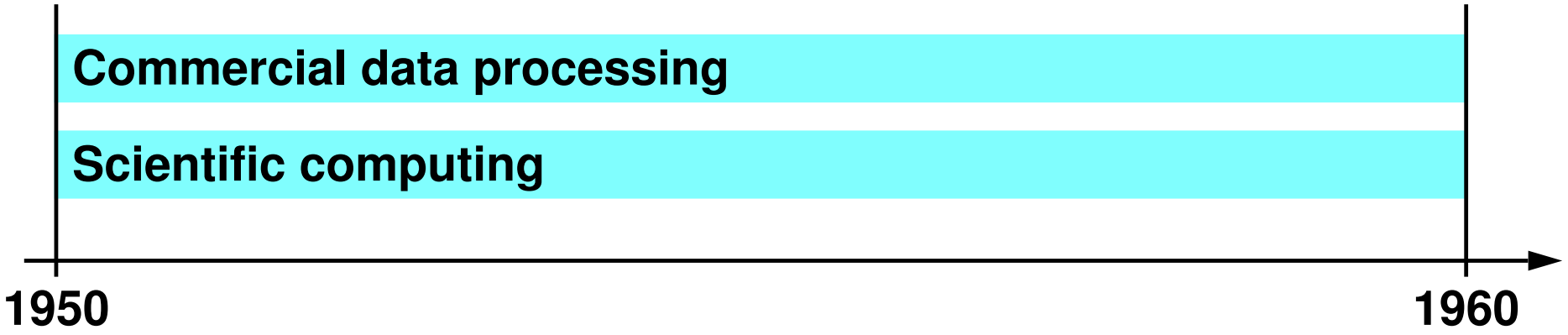


# In the Beginning ...

- There was hardware
  - processor
  - storage
  - card reader
  - tape drive
  - drum
- And not much else
  - no operating system
  - no libraries
  - no compilers
  - very little software in the beginning

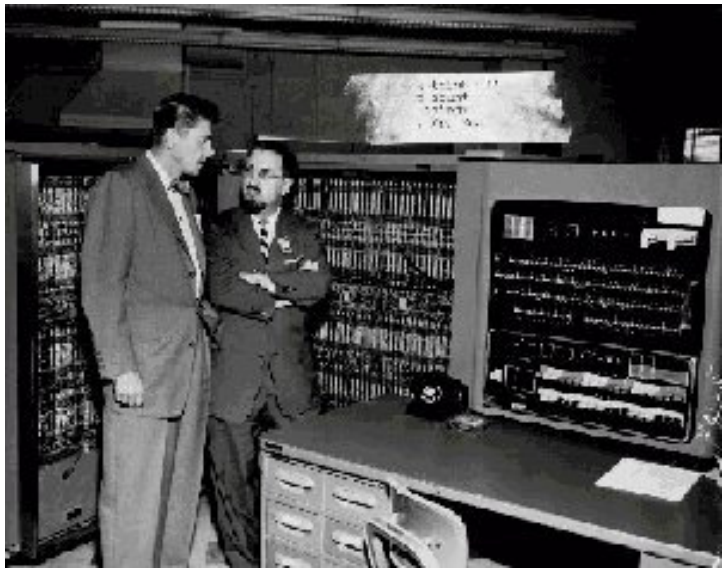


# 1950s




## IBM 701

- OS: Initially, none

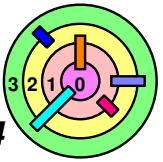


http://www.columbia.edu/cu/computinghistory/grosch.html

## IBM 650

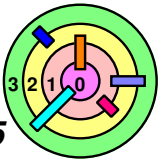
- OS: none

http://www-03.ibm.com/ibm/history/exhibits/650/650_ph10.html



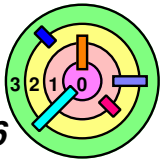
# Programming Without an OS

- Assemble all software into a deck of punched cards
- Get 15-minute computer slot
  1) pay $75 ($611 in 2010 dollars)
  2) mount tapes containing data
  3) read cards into computer
  4) run program
  5) it probably crashes
  6) output (possibly a dump) goes to printer
- Steps 1, 2, 3, and 5 take 10 minutes
  - leaving 5 minutes for step 4

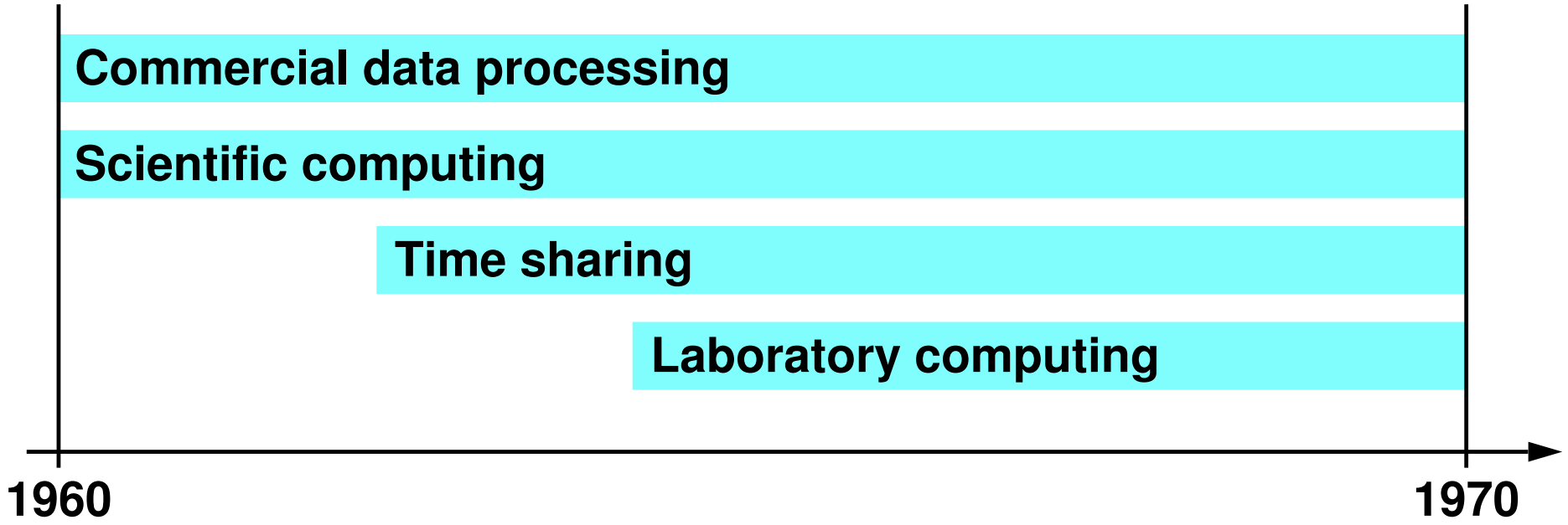


# Enter the OS ...

- Group jobs into batches
- Setup done for all collectively
- Software doing this called *Input/Output System*
  - the first operating system
  - "operating system" is the software that automate things



# 1960s




- According to Doeppner, "The most interesting decade of OS development"
  - starts with the first Virtual Memory system
  - ends with the earliest Unix
  - in between came IBM 360 and Multics

# 1960s

- Goal of OS is to provide the illusion:
  - programmers could write software as if there was more memory than the size of the physical "core"
- *Memory Hierarchy:*
  - core memory (fast and expensive)
  - disks/drums (slower and cheaper)
  - tapes (very slow and a lot cheaper)

## Atlas Computer

http://www.chilton-computing.org.uk/acl/technology/atlas/p002.htm

# 1960s

## IBM 7094

- OS: CTSS (among others)

http://www-03.ibm.com/ibm/history/exhibits/mainframe/mainframe_PP7094.html

## Multics

- OS: Multics

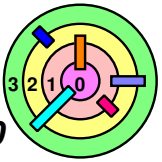
http://www.multicians.org/multics-stories.html

⇨ **CTSS was written by MIT for the IBM 7094**

- persued the idea of ***time sharing***

# 1960s

- Importance of Multics (although its a commercial failure)
  - 1965: probably the first OS written in a high-level language
    - PL/1
    - demonstrated that compiler can generate code that's efficient enough
      - hand-written assembly code is not the only way to go
  - goals were ambitious (and relevant today): reliable storage, security, high throughput for batch jobs, interactive processing, evolvability
    - and got most of the way there
      - way too complex!
    - much work in computer security was on Multics



# 1960s

- Main idea of IBM/360 OS
  - one OS can run on different hardware
    - from small machines to large machines
  - application can be portable to run on different machines
- Didn't work out that way
  - OS needs to be tuned to hardware to have good performance

## The IBM Mainframe

- OS: OS/360

http://www-03.ibm.com/ibm/history/exhibits/mainframe/mainframe_intro2.html

- Became evident that to achieve the original goal would require an anormous effort by a large number of people
  - Fred Brooks, the project leader, later wrote the famous book, "The Mythical Man-Month"
    - a task requiring 12 months of one person's time cannot be done in 1 month by 12 people

# DEC PDP-8


- The first *minicomputer*
- OS:
  - many: ranging from primitive to interesting (a multi-user time-sharing system; a virtual-machine system)

# Unix

http://histoire.info.free.fr/images/pdp11-unix.jpeg

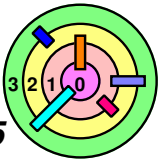
# Unix

http://en.wikipedia.org/wiki/Dennis_Ritchie

- Developed by Ken Thompson & Dennis Ritchie
  - Turing Award (given once per year) in 1983
  - National Medal of Technology (given to multiple technologists every year) in 1998

# History of Concurrency

- Multiprogramming
  - 1961, 1962: Atlas, B5000
  - 1965: OS/360 MFT, MVT
- Timesharing
  - 1961: CTSS (developed by MIT for IBM 7094); BBN time-sharing system for DEC PDP-1
  - mid 60s
    - Dartmouth Timesharing System (DTSS)
    - TOPS-10 (DEC)
  - late 60s
    - Multics (MIT, GE, Bell Labs)
    - Unix (Bell Labs)



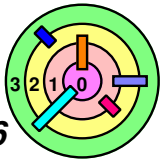
# Apple's Multitasking Announcement

## With Preemptive Multitasking, Everything Happens at Once

In today's fast-paced world, you rarely get to do one thing at a time. Even in the middle of transforming, say, a Photoshop file, you may need to find a crucial piece of information on the web while you compose an urgent reply to a customer. What you need is a computer that can handle several different tasks at once, giving priority to your primary application, but still crunching away at other jobs in the background. ...

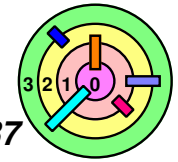
Darwin makes this possible by incorporating a powerful concept called preemptive multitasking. ...

Apple website, September 2000



# History of Virtual Memory

- 1961: Atlas computer, University of Manchester, UK
- 1962: Burroughs B5000
- 1972: IBM OS/370
- 1979: 3 BSD Unix, UC Berkeley
- 1993: Microsoft Windows NT 3.1
- 2000: Apple Macintosh OS X



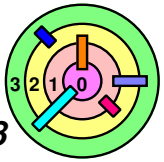
# Apple's VM Announcement...

## Welcome to the Brave New World of Crash-Resistant Computing

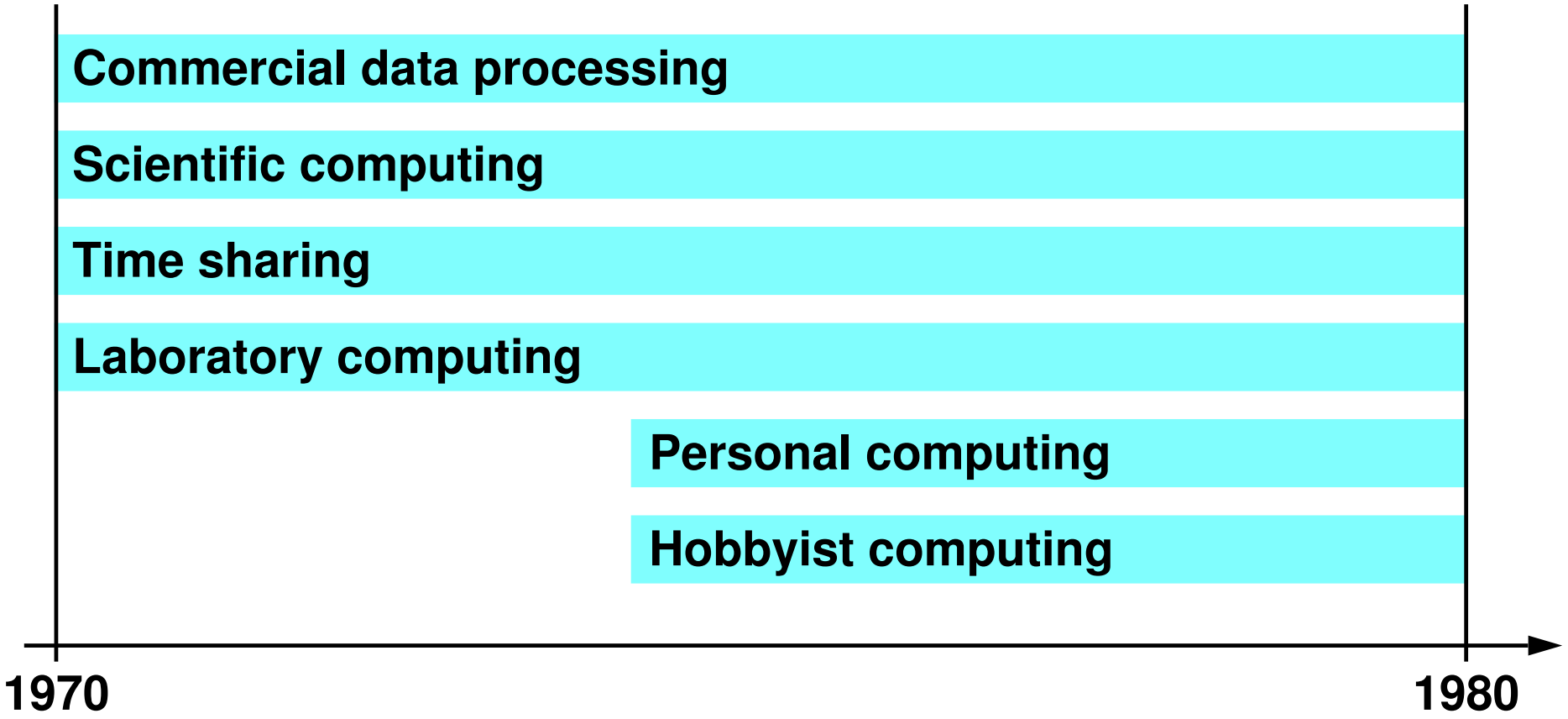
**Let's start with the notion of protected memory, and why it's so important. ... One of the ways an operating system ensures reliability is by protecting applications through a mechanism called protected memory (essentially walling off applications from each other). ...**

**Along with the protected memory mechanism, Darwin provides a super-efficient virtual memory manager to handle that protected memory space. So you no longer have to worry about how much memory an application like Photoshop needs to open large files. ...**

**Apple website, September 2000**



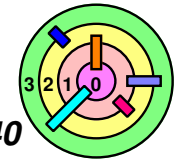
# 1970s



# IBM's Dominance Continues

http://www-03.ibm.com/ibm/history/exhibits/mainframe/mainframe_2423PH3168.html

- OS:
  - OS/370



# Scientific Computing

## Cray-1

- OS: COS
  - single job at a time

http://www.geek.com/articles/chips/hacker-creates-110th-scale-cray-1-supercomputer-20100830/

# Xerox Alto

http://commons.wikimedia.org/wiki/File:Xerox_Alto_full.jpg

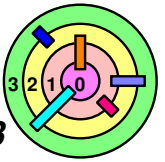
OS:

- single-user, single-computation

1996 PBS documentary, "Triumph of the Nerds", Steve Jobs said, in an interview: "Good artists copy and great artists steal. We have always been shameless about stealing great ideas." This was referring to his visit to Xerox PARC.

# CP/M

- Control Program for Microcomputers, developed by Digital Research
  - 1974
  - first hobbyist OS
  - supported Intel 8080 and other systems
  - clear separation of architecture-dependent code
  - no multiprogramming
  - no protection



# Apple II


http://commons.wikimedia.org/wiki/File:Apple_II.jpg

**OS:**

- **none**
  **later: similar functionality as CP/M (not much)**
- **its sister computer, Lisa, had an OS**

# Microsoft

- Microsoft started out to be a programming-language company selling a ***Basic*** interpreter
  - first was to run on the MITS ALTAIR 8800 which has no OS

http://www.vintage-computer.com/altair8800.shtml

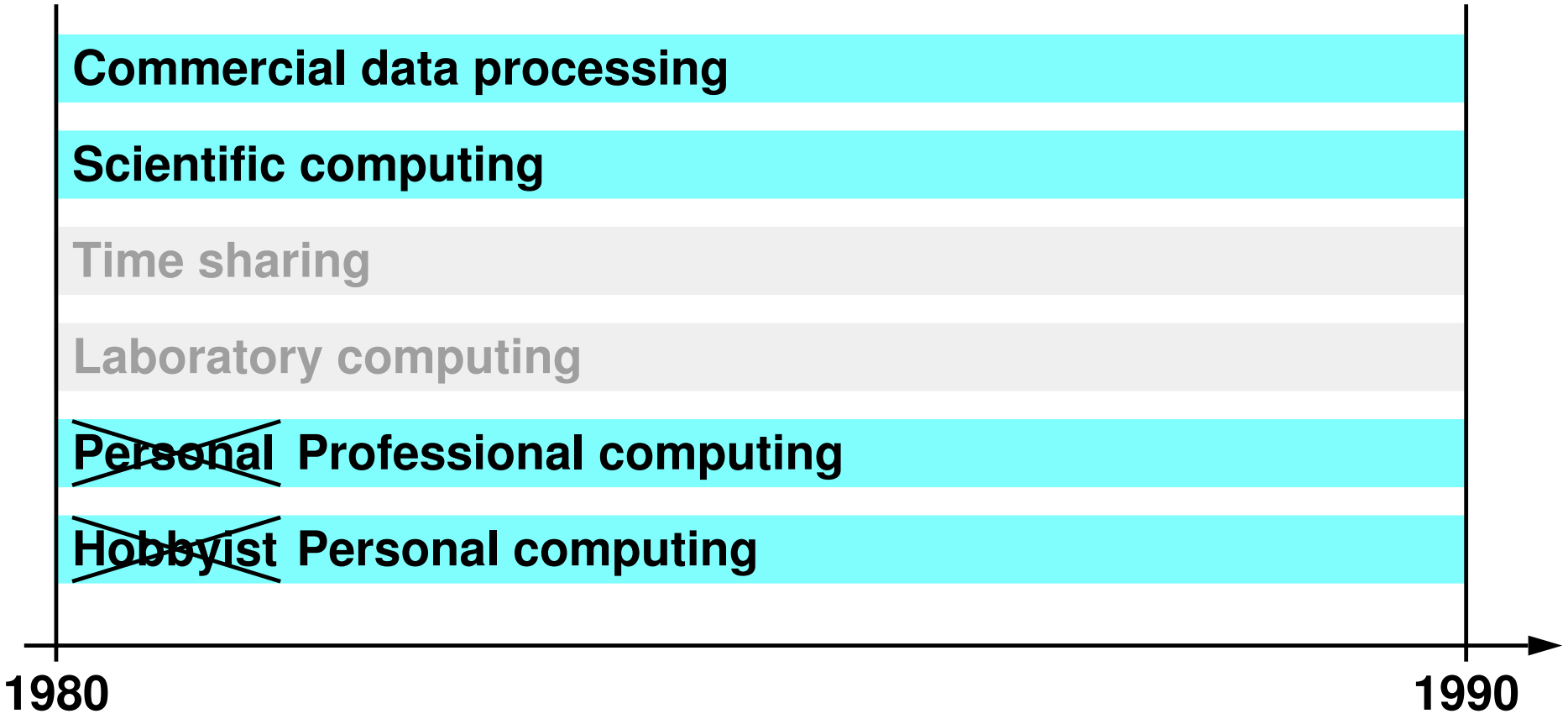
- Microsoft enters the OS business in late 1970s
  - bought a ***Unix*** license
  - Xenix
    - a version of Unix
    - predominant version of Unix in the 1980s
    - used by MS internally into the 1990s

# VAX-11/780


OS:

- VMS
- Unix
- Both:
  - time sharing
  - virtual memory
  - access protection
  - concurrency

# 1980s




- Two OSes Take Off
  - Linux
  - MS-DOS

# IBM PC

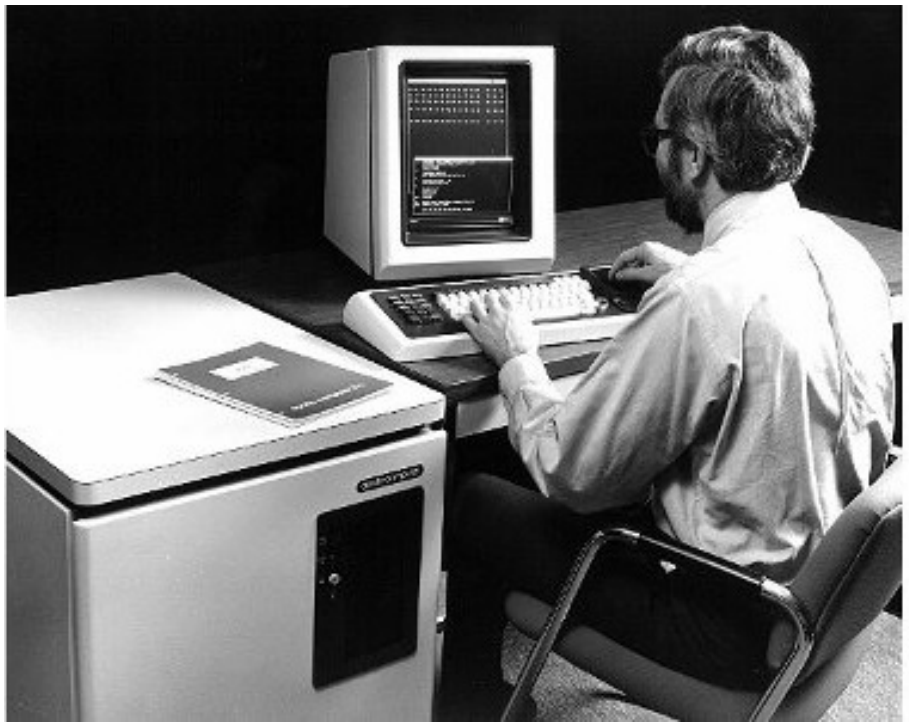
http://en.wikipedia.org/wiki/IBM_Personal_Computer

OS:

- PC-DOS
  (aka MS-DOS)
  (remarkably like CP/M)

- IBM wanted Microsoft to provide Basic for it's up coming IBM PC
- IBM PC had no OS
  - IBM cannot come to agreement with Digital Research to license CP/M
  - Microsoft told IBM, "We'll do it"
  - Microsoft bought QDOS and call it MS-DOS
    - delivered to IBM and sold as PC-DOS

# The Computer Workstation



http://www.computerhistory.org/revolution/computer-graphics-music-and-art/15/217

## The Apollo Workstation

- OS: Aegis
  - virtual memory
  - distributed file system
  - access protection
  - concurrency

# The Computer Workstation

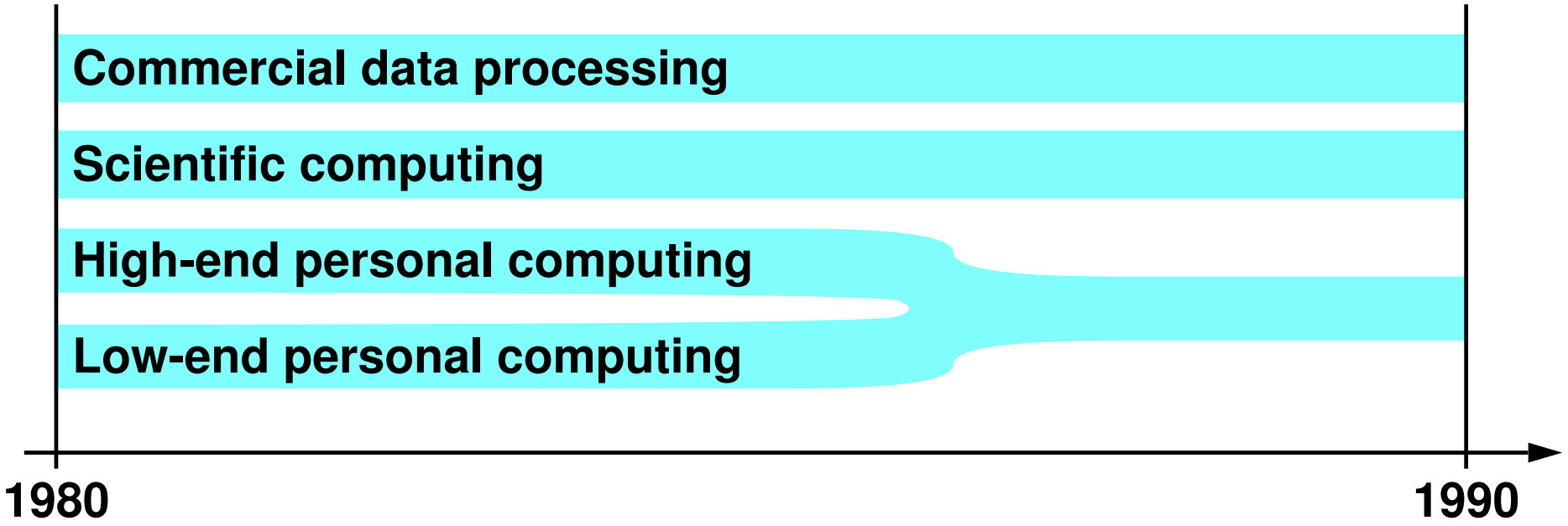

http://en.wikipedia.org/wiki/Sun-2

**Sun Microsystem**

- **OS: SunOS**
  - **derived from BSD 4.3**
    - **one of the founders was Bill Joy**
  - **introduced NFS**

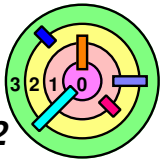
**It says, "The first workstation for under $10,000"**

# 1990s



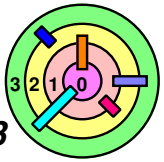
# Microsoft Windows

- Initially an application under MS-DOS
  - even till Windows 3.1, no protection between applications running on top of Windows
  - provided "cooperative multitasking"
  - not a real OS
- Windows 95
  - provided preemptive multitasking
  - but MS-DOS still present, and is part of Windows
    - WIN32 application can "thunk" into WIN16 (i.e., MS-DOS) and die (and bring down the whole OS)
    - same with Windows 98 and Windows ME
  - famous Bill Gates memo that with Windows 95, everything that runs on Windows need to be Internet-aware
    - put TCP/IP on every Windows 95 machine and thus standardized TCP/IP



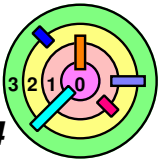
# Toy Operating Systems

- 1987: Andrew Tanenbaum of Vrije Universiteit, Amsterdam, publishes Operating Systems: Design and Implementation
  - included is source code for a complete, though toy, operating system: ***Minix***, sort of based on Unix
- 1991: Linus Torvalds buys an Intel 386 PC
  - MS-DOS doesn't support all its features (e.g., memory protection, multi-tasking)
  - "soups up" Minix to support all this
- January 1992: Torvalds releases Linux 0.12
- January 1992: Tanenbaum declares Linux obsolete



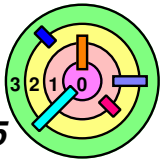
# Late 80s / Early 90s

- 1988: Most major Unix vendors get together and form OSF to produce a common Unix: OSF/1, based on IBM's AIX
- 1989: Microsoft begins work on NT
  - based on VAX-11's VMS architecture (David Cutler was the principle architect of VMS at DEC)
- 1990: OSF abandons AIX, restarts with Mach
- 1991: OSF releases OSF/1
- 1992: Sun releases Solaris 2
  - many SunOS (Solaris 1) programs are broken
- 1993: All major players but DEC have abandoned OSF/1
- 1993: Microsoft releases Windows NT 3.1
- 1994: Linux 1.0 released



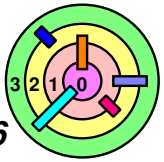
# Late 90s

- IBM has three different versions of Unix, all called "AIX"
- 1996: DEC renames its OSF/1 "Digital Unix"
- 1996: Microsoft releases Windows NT 4
- 1996: Linux 2.0 released
- 1998: DEC is purchased by Compaq; "Digital Unix" is renamed "Tru64 Unix"
- 1999: Sun's follow-on to Solaris 2.6 is called Solaris 7



# The ’00s Part 1

- 2000: Microsoft releases Windows 2000 and Windows ME
- 2000: Linux 2.2 is released
- 2000: IBM "commits" to Linux (on servers)
- ~2000: Apple releases OS X, based on Unix (in particular, OSF/1)
- 2001: Linux 2.4 is released
- 2001: Microsoft releases Windows XP
- 2002: Compaq is purchased by HP
- 2003: SCO claims their code is in Linux, sues IBM; IBM countersues
  - August 10, 2007: judge rules that SCO is not the rightful owner of the Unix copyright, Novell is
  - Novell says there is no Unix in Linux
  - September 2007: SCO files for Chapter 11 bankruptcy protection



# The ’00s Part 2

- 2004: Linux 2.6 is released
- 2005: IBM sells PC business to Lenovo
- July 2005: Microsoft announces Windows Vista
- January 2007: Microsoft releases Windows Vista
- later in 2007: Microsoft starts hinting at Windows 7
- April 2009: Oracle announces purchase of Sun Microsystems
- July 2009: Google announces Chrome OS
- October 2009: Microsoft releases Windows 7