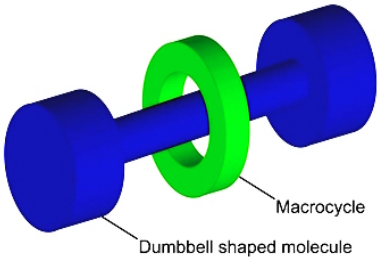
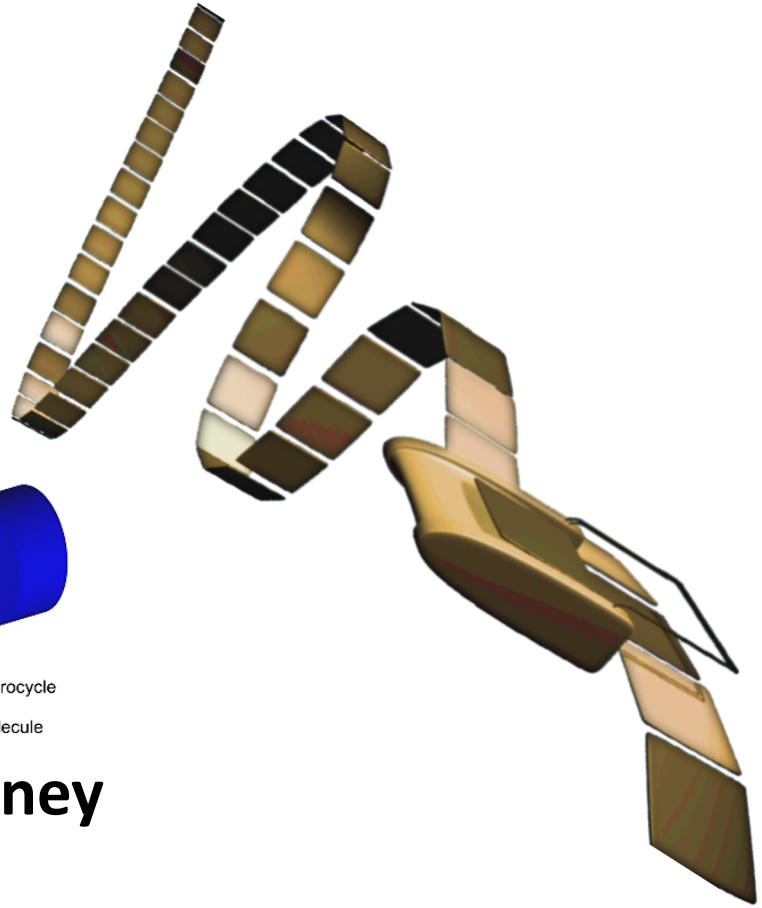
Introduction to Computer Science

CSCI 109

China – Tianhe-2





Andrew Goodney

Fall 2019

Lecture 8: Operating Systems *October 21th, 2019*

Operating Systems

Working Together

Schedule

Date	Topic		Assigned	Due	Quizzes/Midterm/Final	Slide Deck
26-Aug	Introduction	What is computing, how did computers come to be?				1
2-Sep	Labor day					
9-Sep	Computer architecture	How is a modern computer built? Basic architecture and assembly	HW1			2
16-Sep	Data structures	Why organize data? Basic structures for organizing data			**Quiz 1** on material taught in class 8/26 and 9/9	3
23-Sep	Data structures	Trees, Graphs and Traversals	HW2	HW1		4
30-Sep	More Algorithms/Data Structures	Recursion and run-time				5
7-Oct	Complexity and combinatorics	How "long" does it take to run an algorithm. P vs NP			**Quiz 2** on material taught in class 9/16 and 9/23	5
14-Oct	Algorithms and programming	Programming, languages and compilers		HW2	**Quiz 3** on material taught in class 9/30	7
21-Oct	Operating systems	What is an OS? Why do you need one?	HW3		**Quiz 4** on material taught in class 10/7	8
28-Oct	Midterm	Midterm			**Midterm** on all material taught so far.	
4-Nov	Computer networks	How are networks organized? How is the Internet organized?		HW3		9
11-Nov	Artificial intelligence	What is AI? Search, plannning and a quick introduction to machine learning			**Quiz 5** on material taught in class 9/4	10
18-Nov	The limits of computation	What can (and can't) be computed?	HW4		**Quiz 6** on material taught in class 11/11	11
25-Nov	Robotics	Robotics: background and modern systems (e.g., self-driving cars)			**Quiz 7** on material taught in class 11/18	12
2-Dec	Summary, recap, review	Summary, recap, review for final		HW4	**Quiz 8** on material taught in class 11/25	13
13-Dec	Final exam 11 am - 1 pm in SGM 123				**Final** on all material covered in the semester	

Agenda

- Talk about operating systems
- Midterm Style Questions
- Quiz #4

Operating Systems

- What is an OS?
- The kernel, processes and resources
- Protection/Isolation/Security
- Competing for time

Reading:
St. Amant Ch. 6

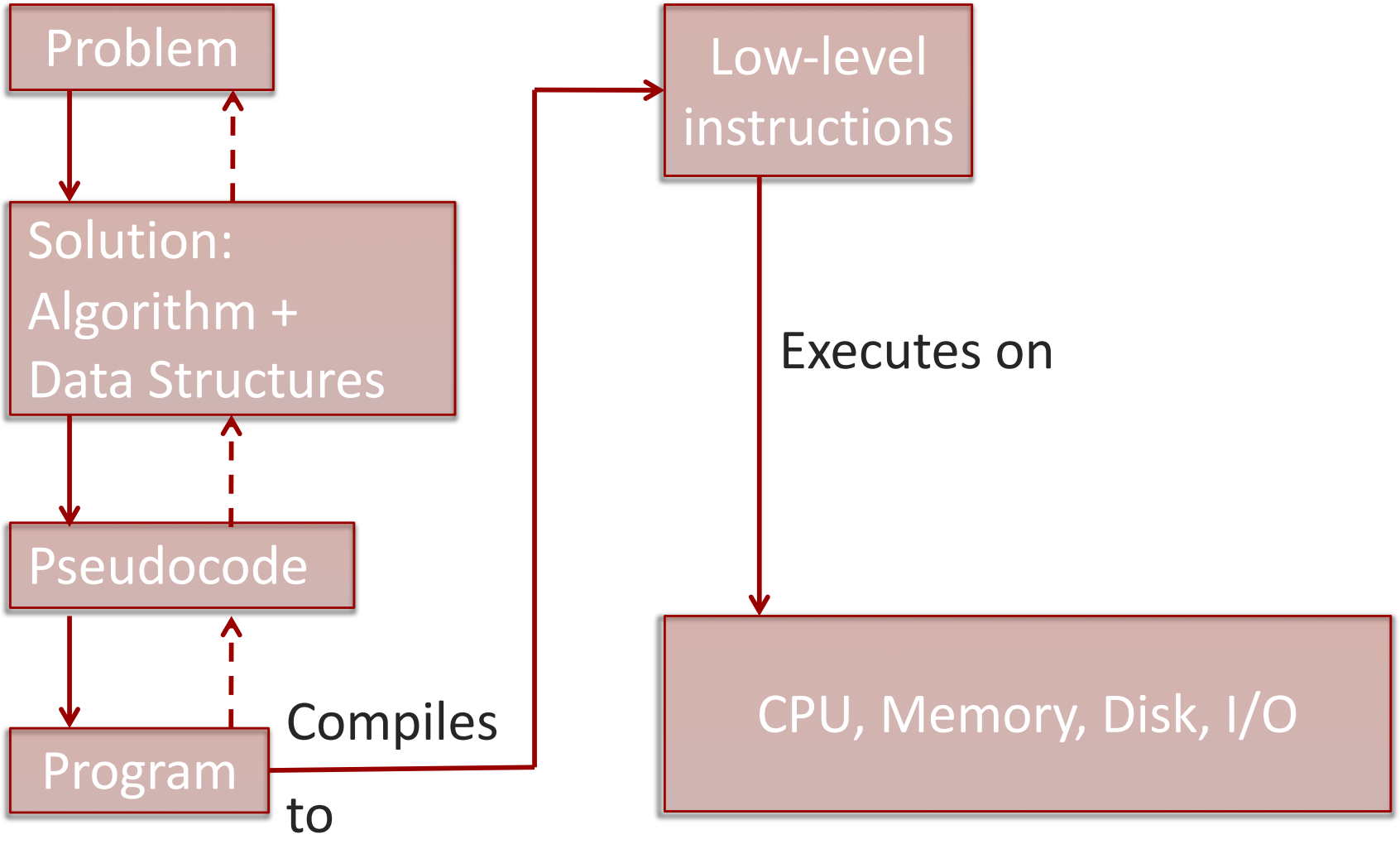
Before Operating Systems

- One computer ⇔ one program
- Program runs start to finish (or crashes)
  - Once done, load the next program
- Thought experiment?
  - Program waits 10ms to load a data item from tape (every once in a while)
  - Over the course of the program execution loads 1,000,000 data items.
  - What happens to the 10,000s while the program was waiting?

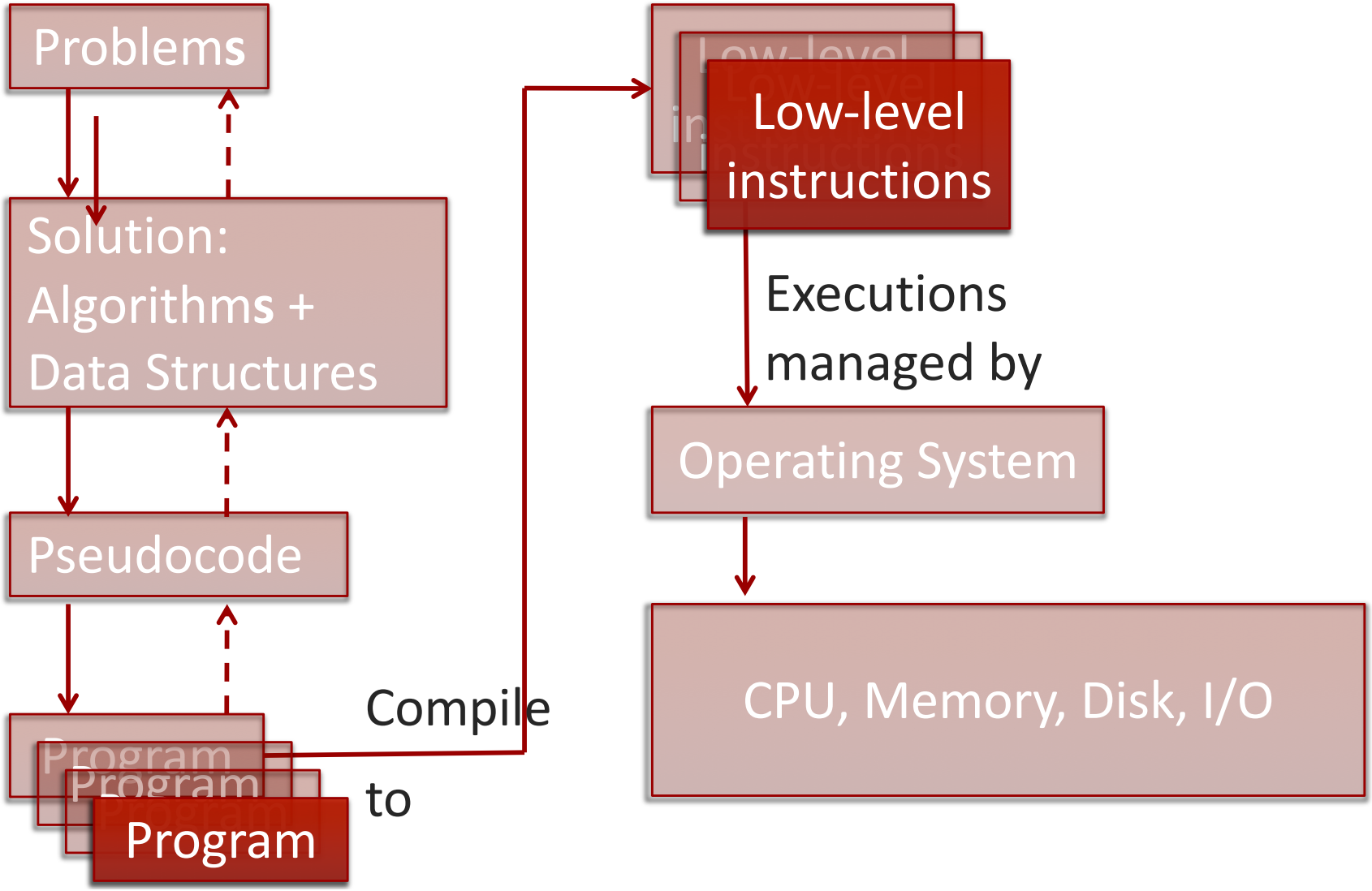
Observation

- Most programs perform some I/O
  - I/O is slow (tape, disk, network, human user, etc.)
- CPU literally does nothing while waiting for I/O
  - This is inefficient
- What if we could share the CPU so when one program is waiting we can run another program?
- Operating systems came out of this need to “time-share” the CPU

The need for an OS



The OS as a executive manager



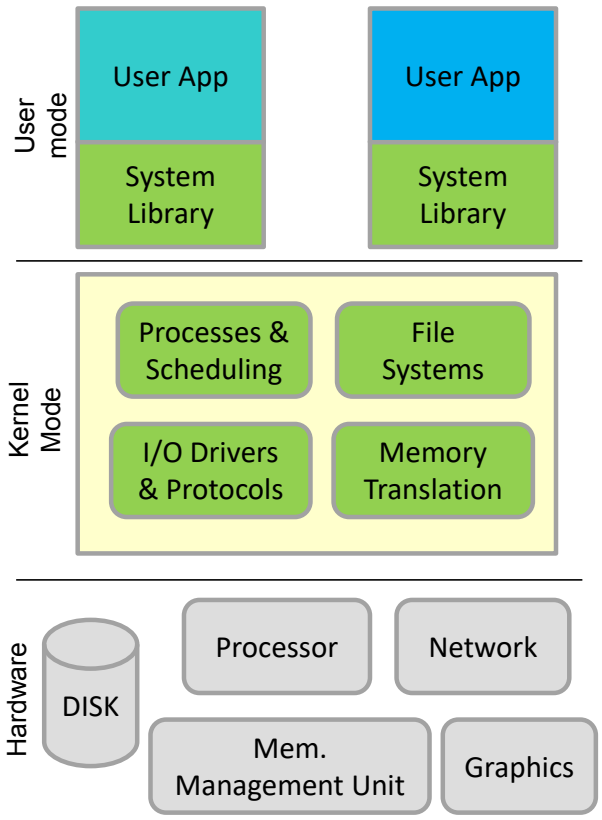
What is an Operating System?

- An executive manager for the computer
- Manages resources
  - Space (i.e. memory)
  - Time (i.e. CPU compute time)
  - Peripherals (i.e. input and output)
- OS is a program that starts, runs, pauses, restarts, and ends other programs

- (some content from the following slides is courtesy of Mark Redekopp and CS350)

Definition

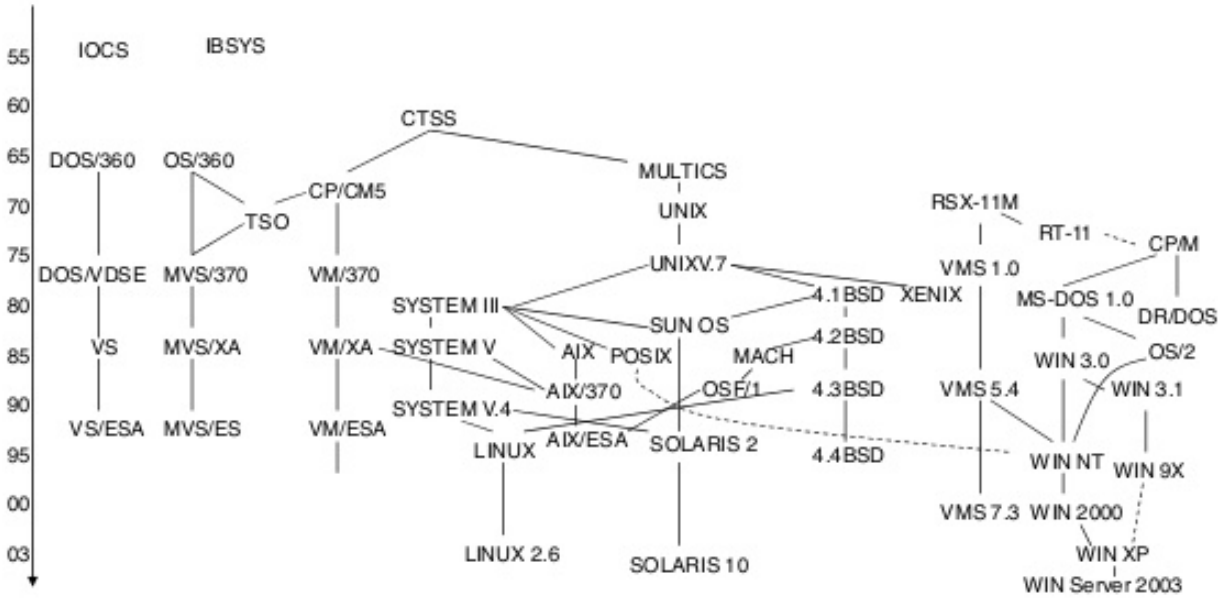
- A piece of software that manages a computer's resources
- What resources need managing?
  - CPU (threads and processes)
  - Memory (Virtual memory, protection)
  - I/O (Abstraction, interrupts, protection)

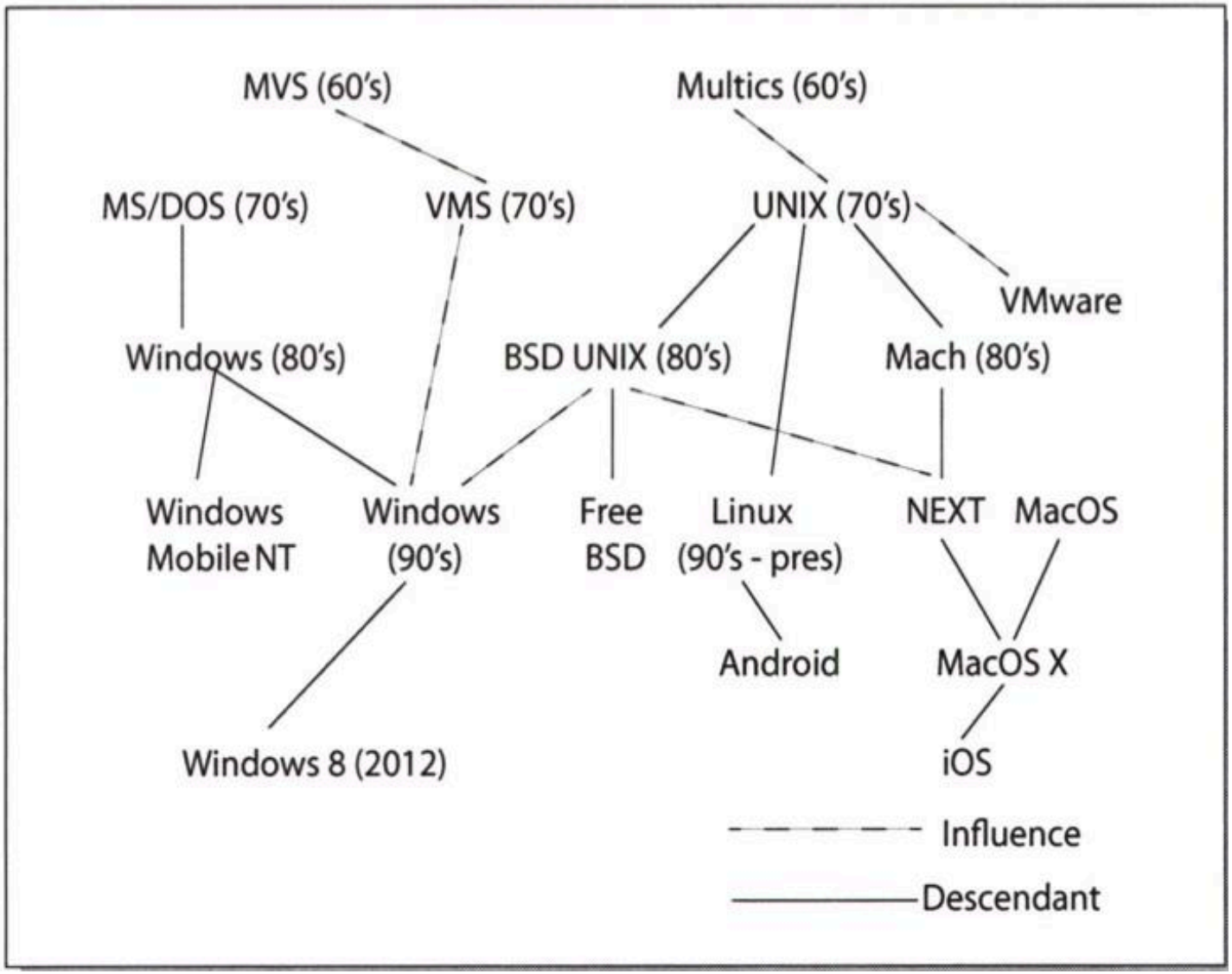


Examples of Operating Systems

- Microsoft Family
  - MSDOS, Windows 3.1 – 98, WindowsNT -> Windows 10
  - Predominately x86 (Intel) hardware, some PowerPC, some ARM
  - FreeDOS
- POSIX (UNIX/like)
  - macOS, FreeBSD, openBSD, netBSD, Solaris, AIX, and others
    - Run on most processor architectures
  - iOS
  - Linux
    - Little side project of university student
    - ”UNIX clone” that won the war
    - 20+ popular distributions
    - Android: heavily customized Linux and Java on phone/tablet
- Others
  - PlaystationOS, VxWorks

History of Modern Operating Systems




MVS (60's)
Multics (60's)
MS/DOS (70's)
VMS (70's)
UNIX (70's)
VMware
Windows (80's)
BSD UNIX (80's)
Mach (80's)
Windows Mobile NT
Windows (90's)
Free BSD
Linux (90's - pres)
NEXT
MacOS
Android
MacOS X
Windows 8 (2012)
iOS
Influence
Descendant

Important Vocabulary

- Resource
  - Some part of the computer that programs use:
    - Memory, CPU, Input/Output devices
- Policy
  - Rules enforced by algorithms that share access to resources
- OS Developers (humans) write policies that achieve some set of goals for the operating systems

What does an Operating System do?

- A bare computer is just hardware
- Programs are written to use that hardware, but exclusive use is inefficient
- In simple terms, the OS:
  - Enables more than one program at a time to use the computer hardware
  - Present computer resources (CPU, disk, I/O) through abstract interfaces to allow sharing
  - Enforce policies to manage/regulate the sharing of resources

Roles

- Referee
  - Protection against other applications
  - Enforce fair resource sharing
    - Why doesn't an infinite loop require a reboot?
- Illusionist (Virtualization)
  - Each program thinks it is running separately
  - Each program thinks it has full access to computer's resources (or unlimited resources)
- Glue
  - Common services (such as copy/paste)
  - Files can be read by any application
  - UI routines for look & feel
  - Separate applications from hardware
    - so you don’t need to know which keyboard, disk drive, etc

OS Design Criteria

- Reliability (and availability)
- Security & Privacy
- Performance
- Portability

Reliability and Availably

- Reliable systems work properly
  - Correct (or expected) outputs are generated for a set of inputs
  - If this is not the case, the system has failed
    - Examples?
- Available systems are available to do work
- Available does not imply reliable
  - System can be available but not reliable (system has bugs, generates wrong results)
  - System can be reliable but not available
    - Crash every 5 minutes, but saves results and restarts 5 minutes later

Privacy, Security, Isolation

- For an OS security means the OS does not run unintended code or get into a compromised state
  - No virus/malware
- OS privacy means programs should not get access to data they should not have
  - Password keychains, files in other users directories
- Security and Privacy require some tradeoffs with performance, which is why OS's are not 100% secure
  - Some are better than others!

Portability

- Many machine types exist: x86, x86_64, PPC, ARM, MIPS
- Many different motherboards or hardware platforms exist: server with 8 CPUs 12 PCIe slots to RaspberryPi, to AppleTV, etc.
- OS with good portability abstracts these differences into a stable API so programmers don't notice
- Also, can the OS itself be ported to new hardware easily?
- Good portability leads to wide adoption
  - Linux, Windows

Performance

- What does performance mean?
  - Lots of computation?
  - Fluid GUI for game?
  - Low latency disk for database?
- OS balances these with policies
  - Major axis is throughput vs. response time
  - Different OS's are tuned based on use case
  - DB server has different policies than Windows gaming rig

Examples of Policies

- Tasks are given priorities; higher priority tasks are handled first
- Some kind of tasks are never interrupted
- All tasks are equal priority; round-robin
- Some tasks can only use part of a disk
- Some tasks can use network

The kernel

- The kernel is the core of an OS
- Kernel coordinates other programs
- When the computer starts up the kernel is copied from the disk to the memory
- Kernel runs until some other program needs to use the CPU
- Kernel pauses itself to run other program

Multitasking

- One program uses the CPU at a time
- OS switches CPU usage (rapidly)
- Creates an illusion that all the programs are running at the same time
- Changeover from one program to another is called a *context switch*
- Examples of context switching?
- Can context switching be good for a program?
- Can context switching be good for a CPU?

Abstractions: Processes and Resources

- Resources
  - Space (memory)
  - Time (CPU)
  - Peripherals (printers etc.)
- Process: an executing program
  - Program counter
  - Contents of registers
  - Allocated memory & contents
- OS doesn't worry about what each program does
- Instead OS cares about
  - What resources does a process need?
  - How long will it run?
  - How important is it?

Protection/Isolation

- Other processes have to be prevented from writing to the memory used by the kernel
- Crash in one program shouldn't crash OS or other programs
- OS has access to all resources: privileged mode
- User programs have restricted access: user mode
- When a user program needs access to protected resources it makes a *system call* (e.g., managing files, accessing a printer)
- Principle of *least privilege* (kernel has highest privilege)

Keep the CPU busy!

- Keeping CPU busy is THE MOST IMPORTANT THING EVER!
- Lets look at some policies that can help us do that.
- We assume we have lots of work (i.e. different programs)

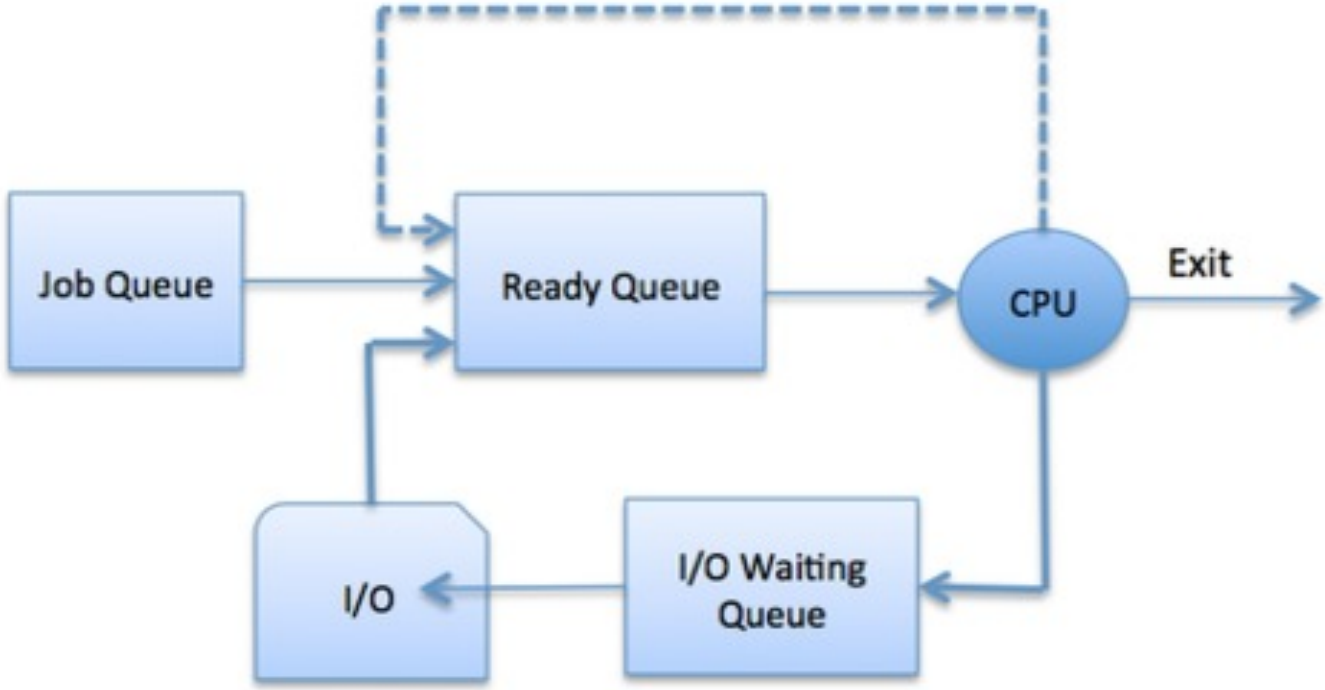
Competing for time

- Think of the time the CPU spends in chunks or blocks
- How can blocks of time be allocated to different processes so that work can be done efficiently?
- Policy: rules to enforce process prioritization

Process Scheduling Policies

- The process *queue*
- Round-robin
- First-come, first-served
- Priority-based
  - Preset priority for each process
  - Shortest-remaining-time
- All these policies keep the CPU busy
- Are there other ways to judge a policy?

Keeping CPU busy



How to evaluate a policy?

- Utilization: how much work the CPU does
- Throughput: # of processes that use the CPU in a certain time
- Latency: average amount of time that processes have to wait before running
- Fairness: every process gets a chance to use the CPU

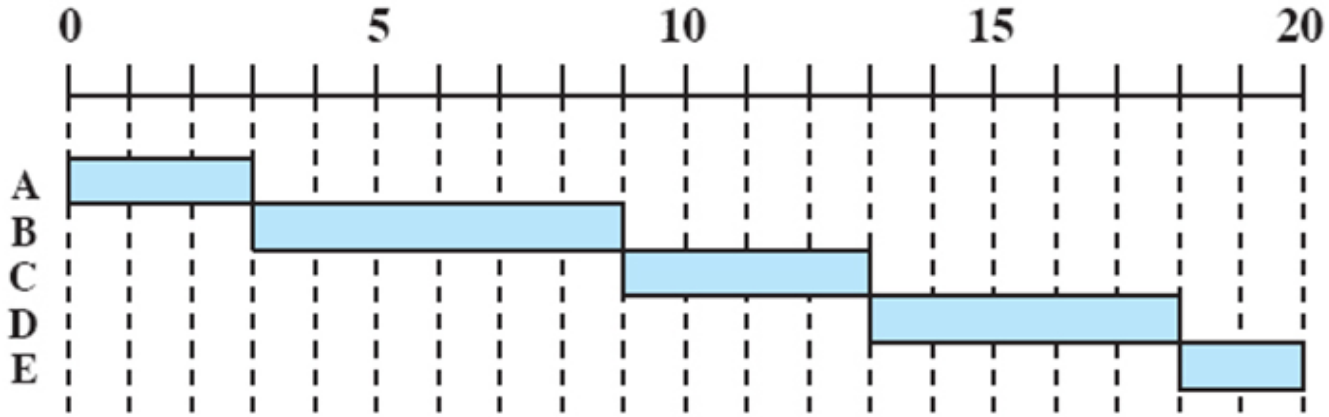
	CPU utilization	Throughput	Latency	Fairness
Round-robin	Good	Variable	Potentially high	Yes No starvation
First-come first-served	Good	Variable		Yes
Shortest remaining time	Good	High	Potentially high	No Could have starvation
Fixed priority	Good			

Everyday policies

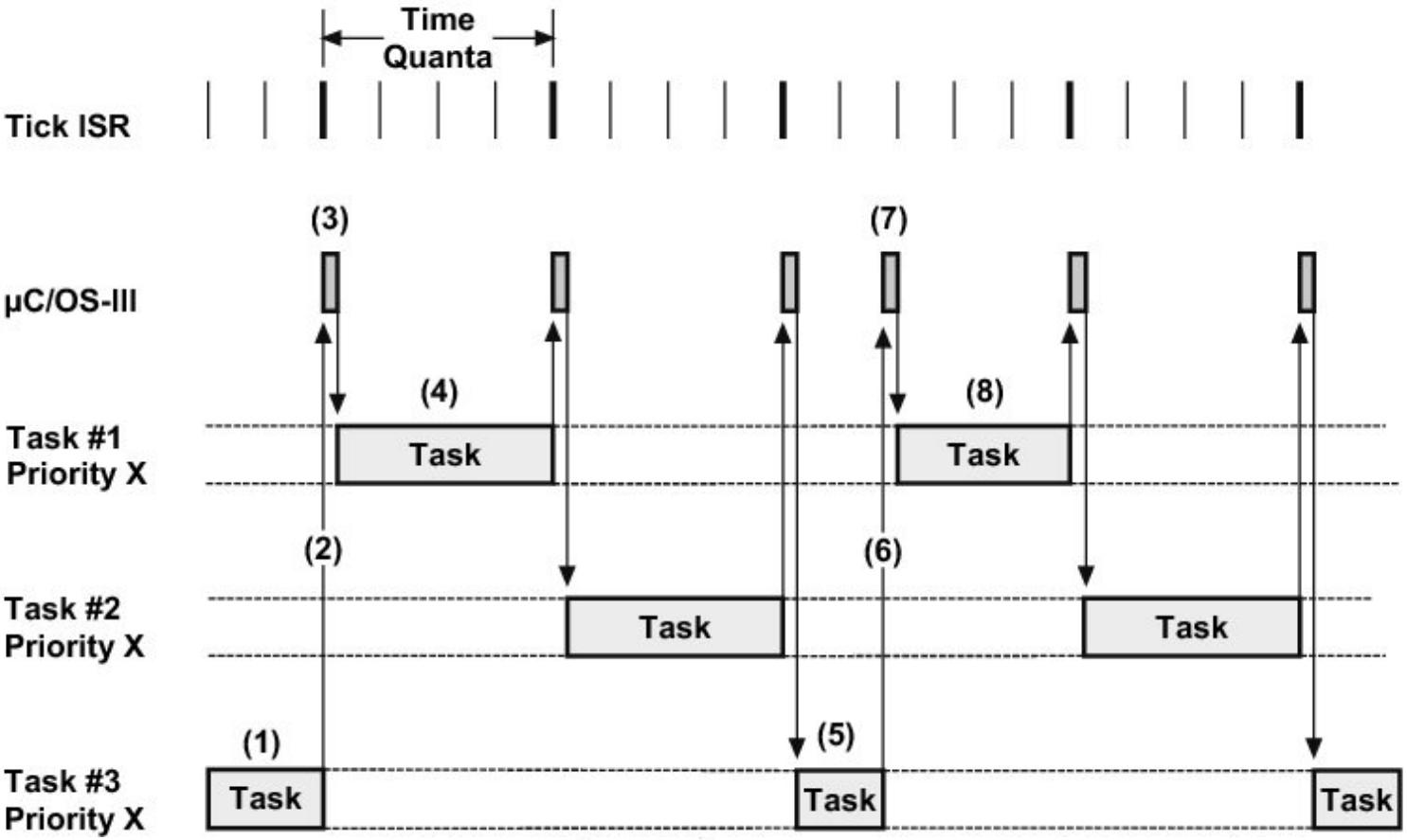
- Planes taking off: first come first served
  - High efficiency for the runway
  - If several smaller planes in line before a large one, not efficient for the average passenger
- Traffic at an intersection w/light out: round robin
  - First traffic in one direction, then another
  - If a police car arrives, then switch to priority-based
  - Unlikely to ever be shortest remaining time

First-come, first serve (non-pre-emptive)

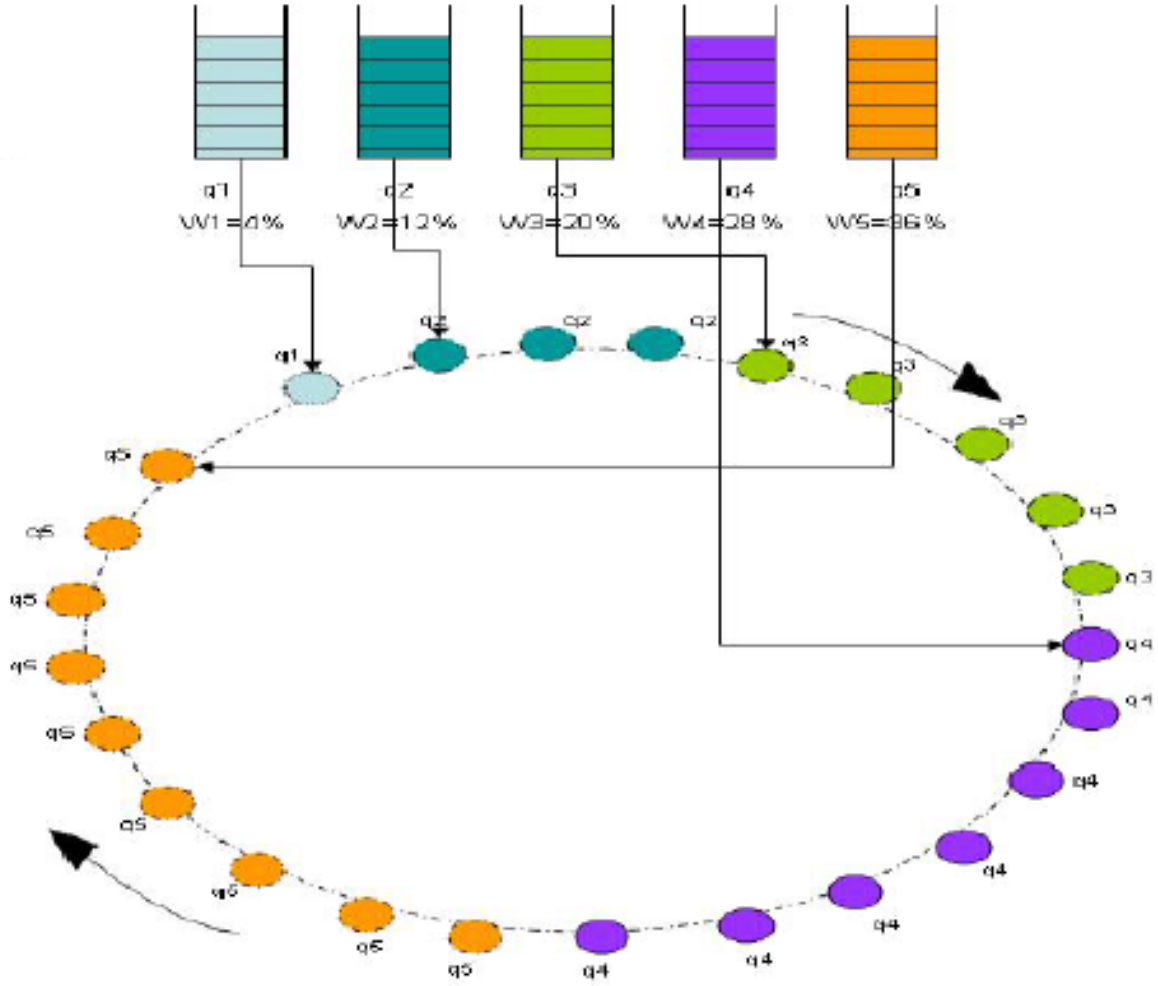
First-Come-First Served (FCFS)



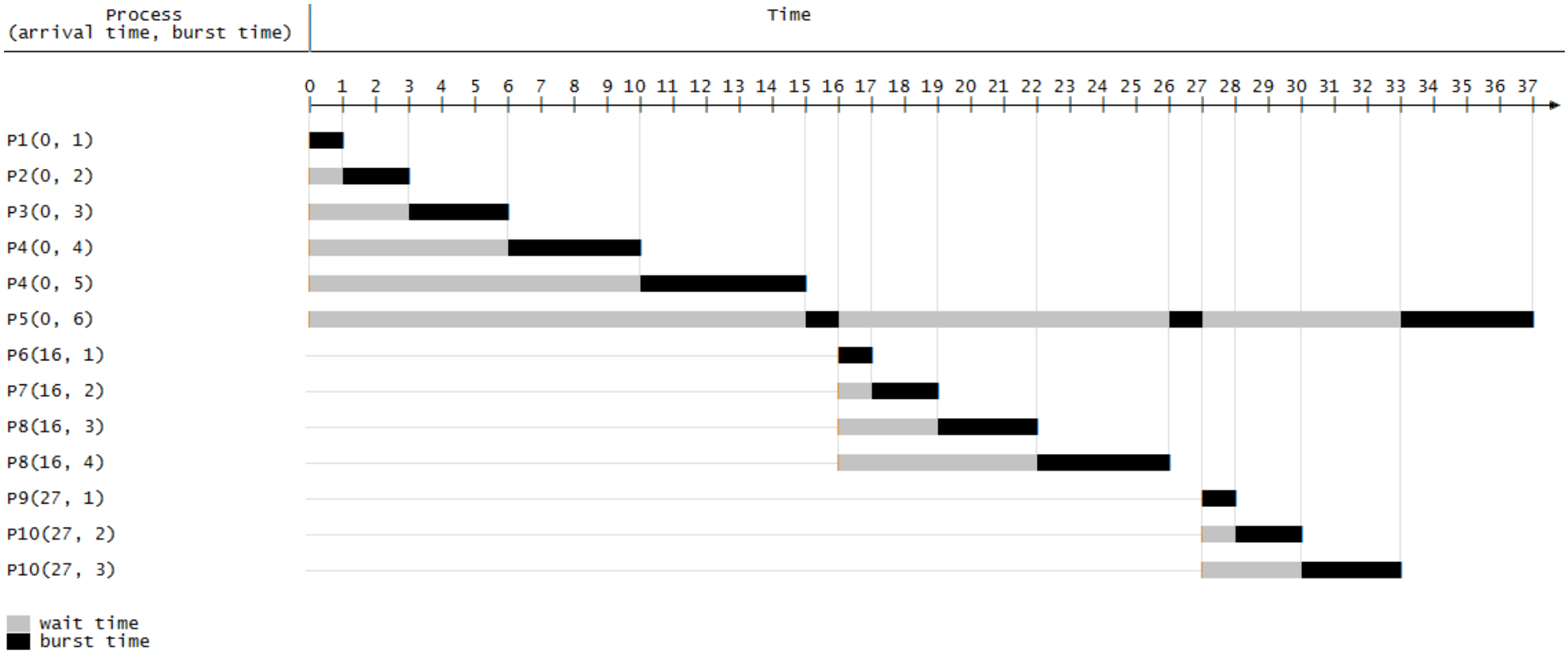
Round Robin (pre-emptive)



Weighted Round Robin



Shortest-time Remaining



Midterm Style Questions

Which of the problems described CANNOT be solved optimally with an MST (minimum spanning tree)?

A. Build the shortest-length bridge network between a set of islands.
B. Eliminate loops in a computer network.
C. Given a list of cities and the distances between each pair, find the shortest possible route that visits each city and returns to the starting city.
D. Eliminate multiple paths between any two vertices in a graph.
E. All of the above CAN be solved optimally with a MST.

Which of the following is **TRUE** about binary search?

A. Considering the input data, binary search will ALWAYS have a smaller runtime vs. sequential search on the same data.
B. Binary search can be applied to any list
C. Binary search has runtime complexity of $O(2^N)$ for an unsorted list
D. Binary search can be implemented recursively
E. None of the above is true

Midterm Style Questions

Which choice for pivot always allows optimal runtime of the quicksort algorithm?

A. Maximum element
B. Minimum element
C. Average among all elements
D. Average between maximum and minimum elements
E. None of the above

You are in a maze and a friend suggests that you put your right hand on the wall and follow the wall until you find the exit. This "right hand rule" represents an algorithm for solving the maze. Which algorithm discussed in class does the approach correspond to?

A. Breadth First Search
B. Depth First Search
C. Kruskal's Algorithm
D. Binary Search

Midterm Style Questions

The Jacquard Loom (and similar machines) are considered information transformers, but not computers. Which answer best describes why:

A. Programming these machines doesn't scale

B. Programming these machines requires punch-cards

C. Machines like these do not have memory or control flow

D. Machines like these are too old to be considered computers

The subset-sum problem has time complexity $O(N*2^N)$. Where does the factor N come from?

A: That is how many subsets a set of size N has.

B: O(N) is the time complexity required to check each possible subset sum.

C: That is the time complexity of the algorithm that generates the subsets.

D: None of the above.

Midterm Style Questions

28. You are working with a list of test scores that was supposedly entered in numerical order. However, about 1% of the scores are instead out of place by one spot. Which sorting algorithm will complete the job faster?
    A. Selection Sort
    B. Insertion Sort
    C. Doesn't matter, we will get the same runtime.
    D. Not enough information has been given to answer this question.

20. When an instruction is loaded from memory, it is desirable to load the contents of a few succeeding memory addresses into the cache. Why is that?
    A. The CPU is unable to only load one instruction at a time
    B. Those contents are likely to be useful in the immediate future according to the spatial locality principle
    C. Those contents are likely to be useful in the immediate future according to the temporal locality principle
    D. The contents stored after the instruction are the values used in the computation of the instruction and therefore must be loaded with the instruction.