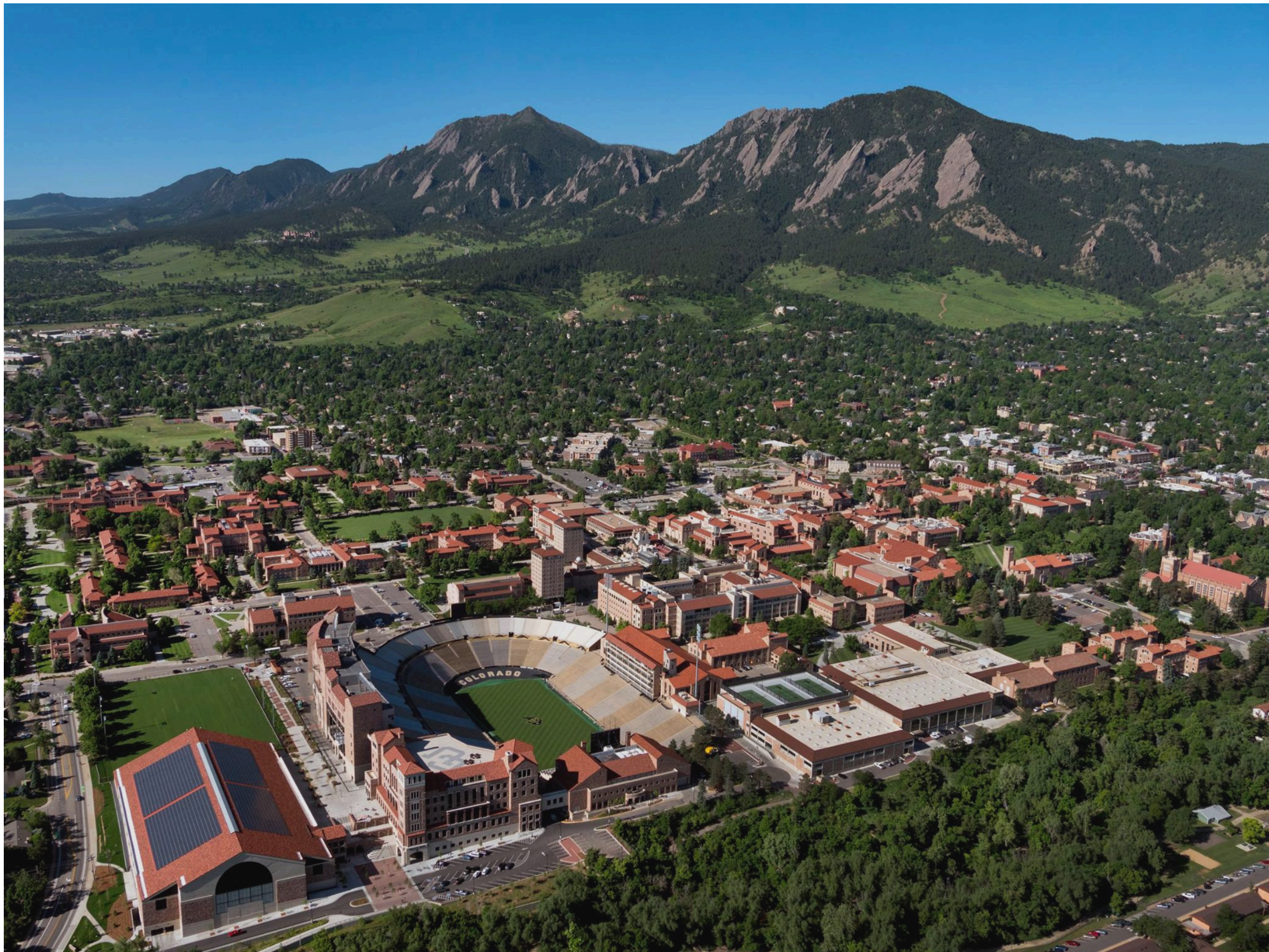
Department of Computer Science
UNIVERSITY OF COLORADO BOULDER

Design and Analysis of Operating Systems CSCI 3753

Dr. David Knox
University of Colorado Boulder

These slides adapted from materials provided by the textbook authors.


COLORADO

Process Scheduling

Diagram of Process State

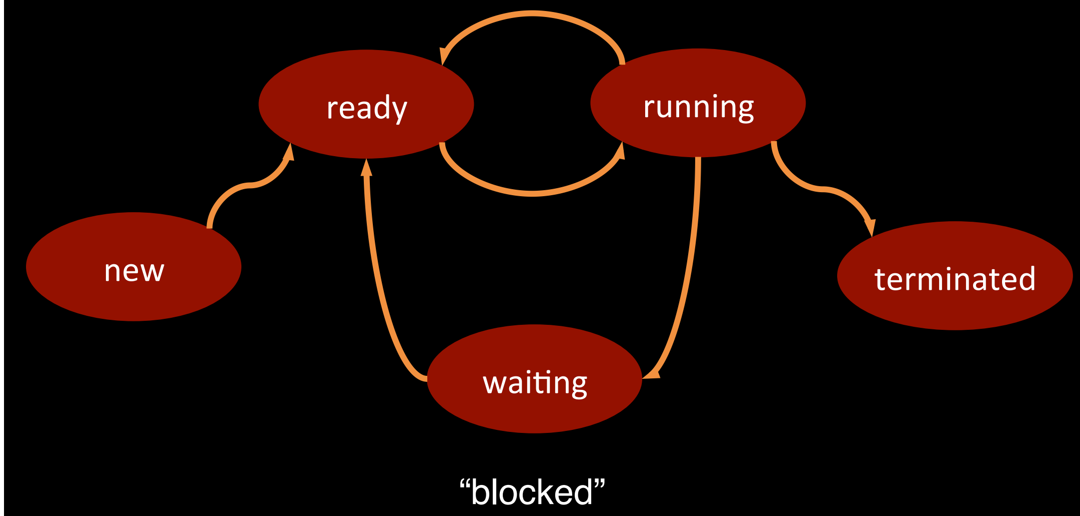
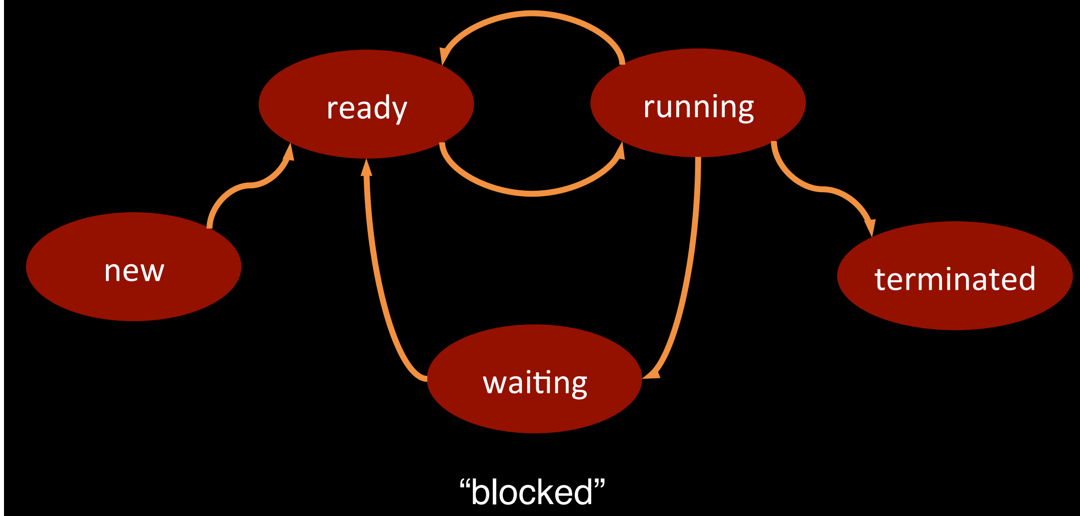


Diagram of Process State

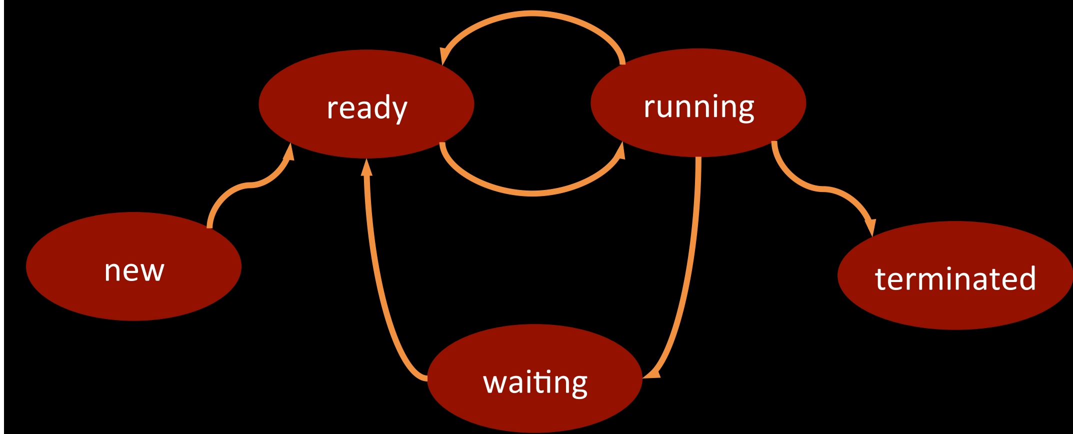


Switching Between Processes

When can a process be switched out :

- blocking on I/O
- voluntarily yielding the CPU, e.g. via other system calls
- being preemptively time sliced, i.e. interrupted
- termination

Diagram of Process State



"blocked"

Switching Between Processes

- The dispatcher gives control of CPU to the process selected by the scheduler, causing context switch:

Policy

Mechanism

  - save old state
  - select next process
  - load new state
  - switch to user mode, jumping to the proper location in the user process to restart that process

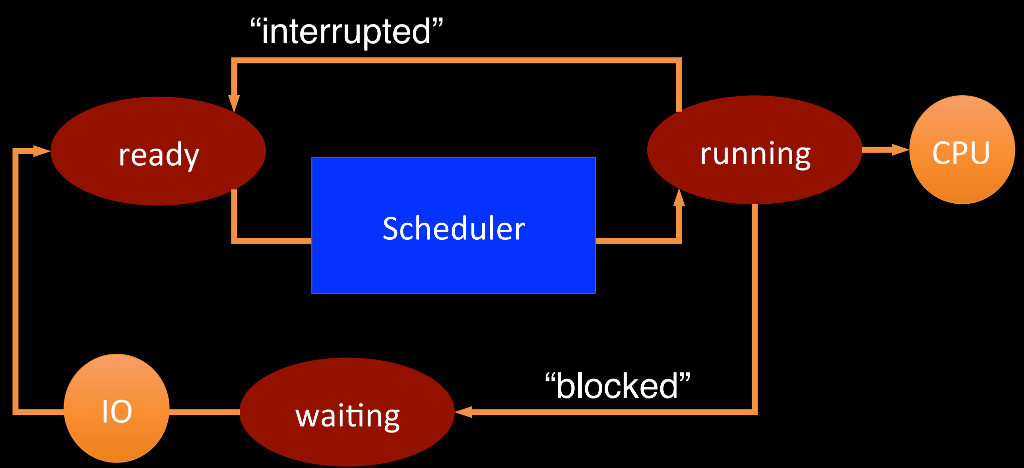
- Separate
  - *mechanism* of scheduling
  - from *policy* of scheduling

Diagram of Process State

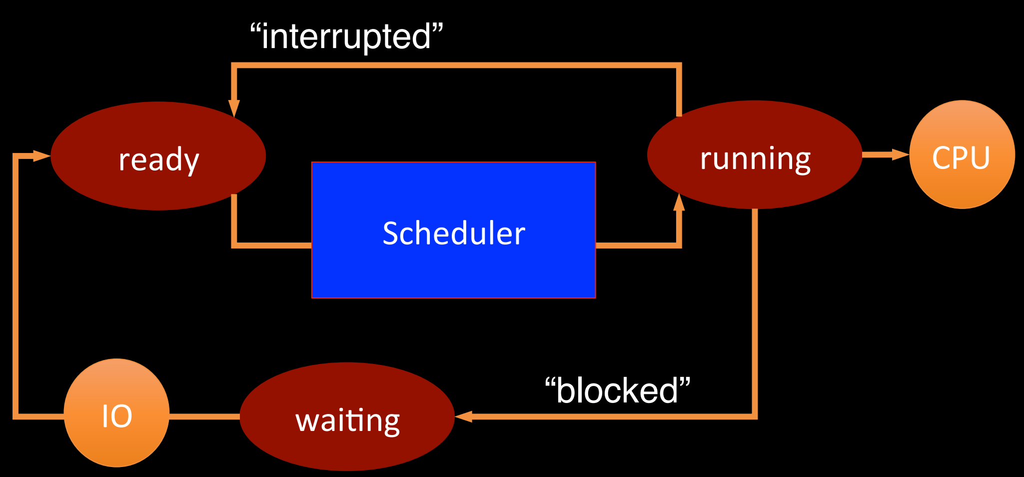
Context Switch Overhead

- Typically take 10 microseconds to copy register state to/from memory
  - on a 1 GHz CPU, that's 10000 wasted cycles per context switch!
- If the time slice is on the order of a context switch, then CPU spends most of its time context switching
  - Typically choose time slice to be large enough so that only 10% of CPU time is spent context switching
  - Most modern systems choose time slices of 10-100 ms

Process Scheduling



Process Scheduling

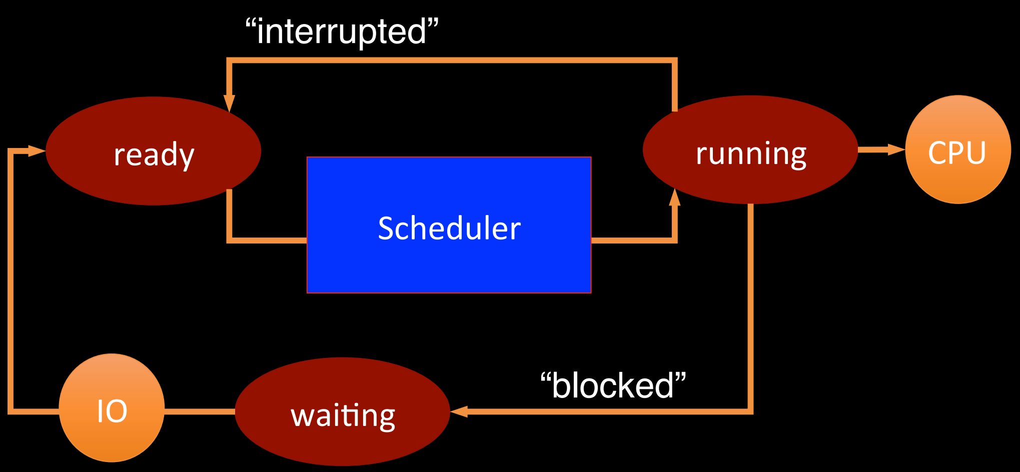


Scheduling

Scheduler's job is to decide the next process (or kernel thread) to run

- From among the set of processes/ kernel threads in the ready queue

Process Scheduling

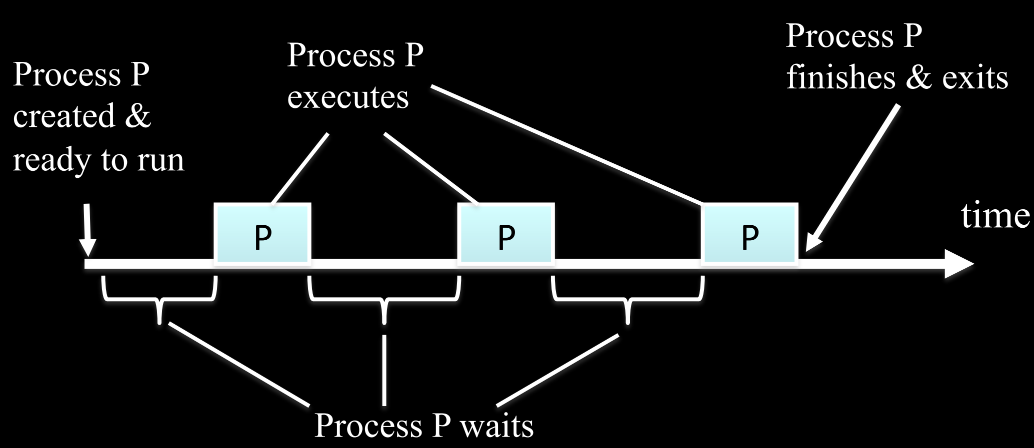


Scheduling Policy

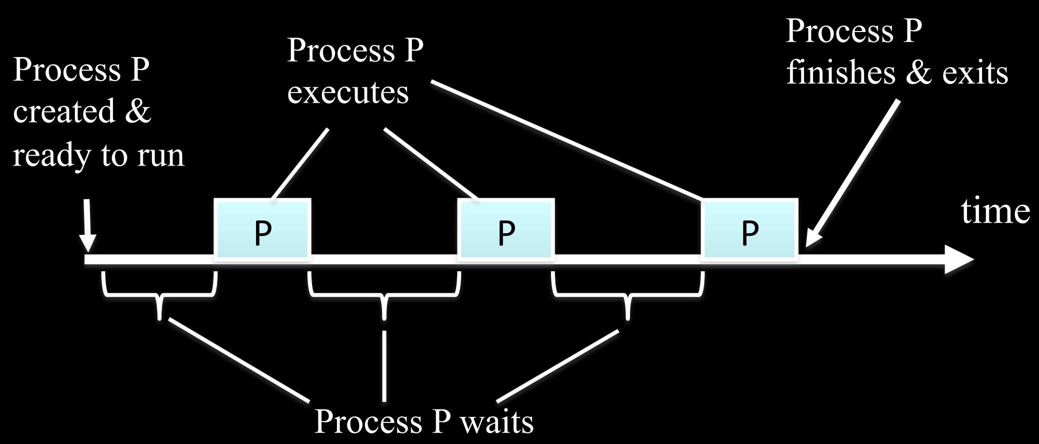
Scheduler implements a *scheduling policy* based on some of the following goals:

- maximize CPU utilization: 40% to 90%
- maximize throughput: # processes completed/second
- maximize fairness
- meet deadlines or delay guarantees
- ensure adherence to priorities
- minimize average or peak turnaround time: from 1st entry to temination
- min avg or peak waiting time: sum of time in ready queue
- min avg or peak response time: time until first response

Scheduling Definitions



Scheduling Definitions



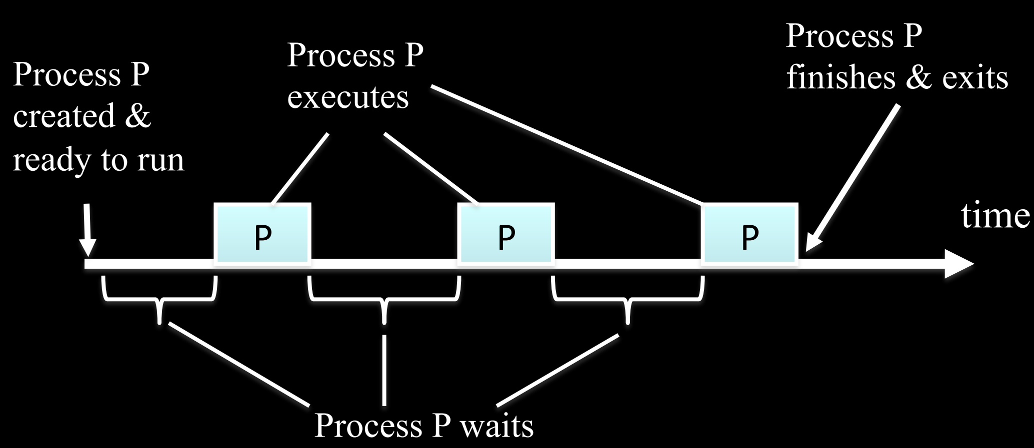
Scheduling Criteria

execution time

$E(P_i)$ = the time on the CPU required to fully execute process i

- Sum up the time slices given to process i
- Also called the "burst time" by textbook

Scheduling Definitions



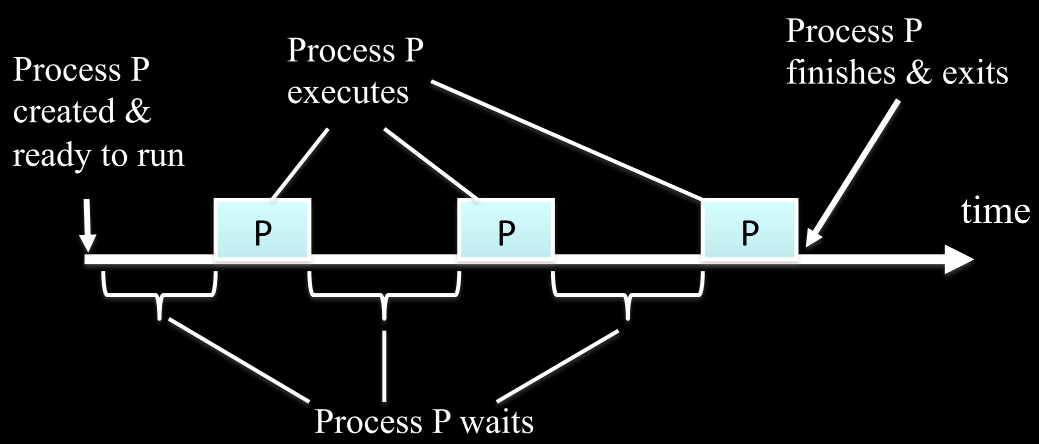
Scheduling Criteria

wait time

$W(P_i)$ = the time process i is in the ready state/ queue waiting but not running

- does NOT include time waiting for IO to complete

Scheduling Definitions



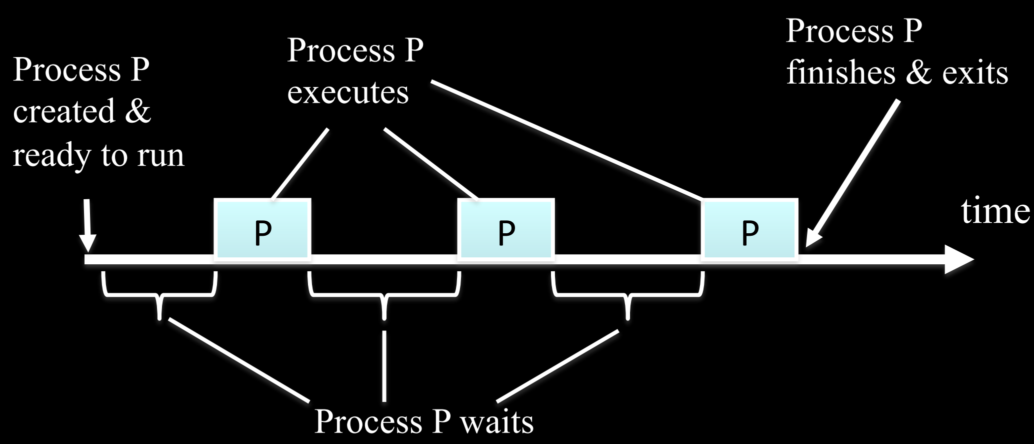
Scheduling Criteria

turnaround time

$T(P_i)$ = the time from 1st entry of process i into the ready queue to its final exit from the system

- does include time waiting and time for IO to complete

Scheduling Definitions



Scheduling Criteria

response time

$R(P_i)$ = the time from 1st entry of process i into the ready queue to its 1st scheduling on the CPU (1st occurrence in running state)

- useful for interactive tasks

Scheduling Analysis

- We analyze various scheduling policies to see how efficiently they perform with respect to metrics
  - wait time, turnaround time, response time, etc.
- Some algorithms will be optimal in certain metrics
- To simplify our analysis, assume:
  - No blocking I/O. Focus only on scheduling processes/tasks that have provided their execution times
  - Processes execute until completion, unless otherwise noted

Scheduling Policy

- How does Scheduler pick the next process to be run?
  - depends on which policy is implemented

- What is the simplest policy you can think of for picking from a group of processes?

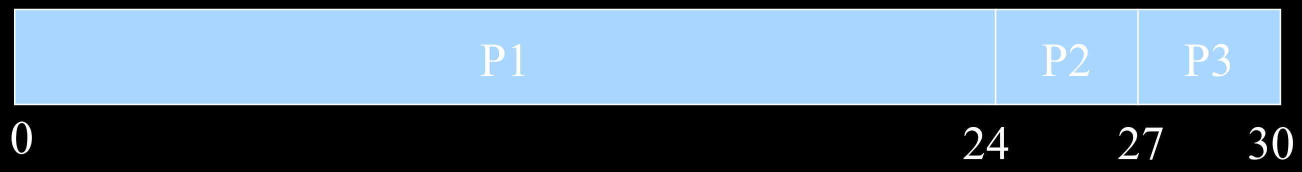
- How about something more complicated?

First Come First Serve (FCFS) Scheduling Policy

- order of arrival dictates order of scheduling
  - Non-preemptive, processes execute until completion

- If processes arrived in order P1, P2, P3 before time 0, then CPU service time is:

Process	CPU Execution Time
P1	24
P2	3
P3	3



FCFS Scheduling

- If processes arrive in reverse order P3, P2, P1 before time 0, then CPU service time is:

Process	CPU Execution Time
P1	24
P2	3
P3	3

Gantt Chart

- The chart we have been using to visualize the sequence of events and time allocated to processes is call a Gantt Chart
- We used this same concept in our first descriptions of scheduling using multi-process and multi-tasking

Process	CPU Execution Time
P1	24
P2	3
P3	3

Gantt Chart

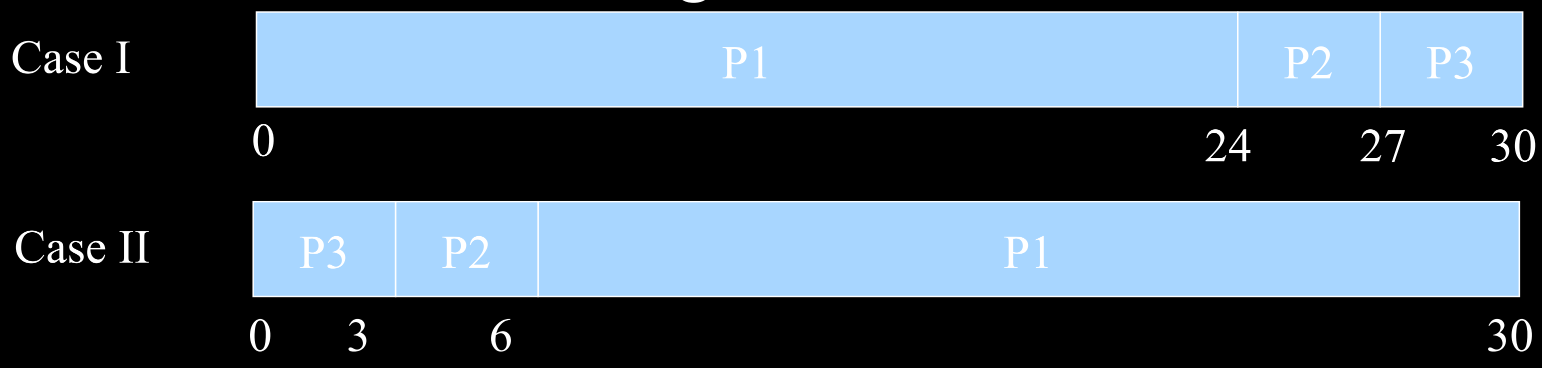
- We used a different format to describe the process times of multi-tasking and multi-programming

- These formats are equivalent

Process	CPU Execution Time
P1	24
P2	3
P3	3

FCFS Scheduling

Process	CPU Execution Time
P1	24
P2	3
P3	3



Lets calculate the ***average wait time*** for each of the cases

All processes arrived just before time 0

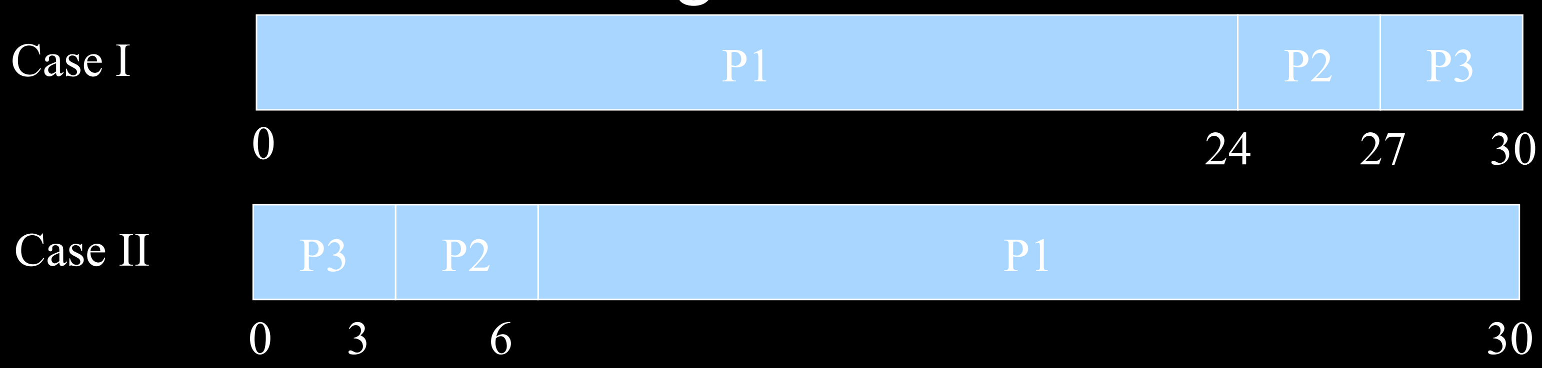
P1 does not wait at all

P2 waits for P1 to complete before being scheduled

P3 waits until P2 has completed

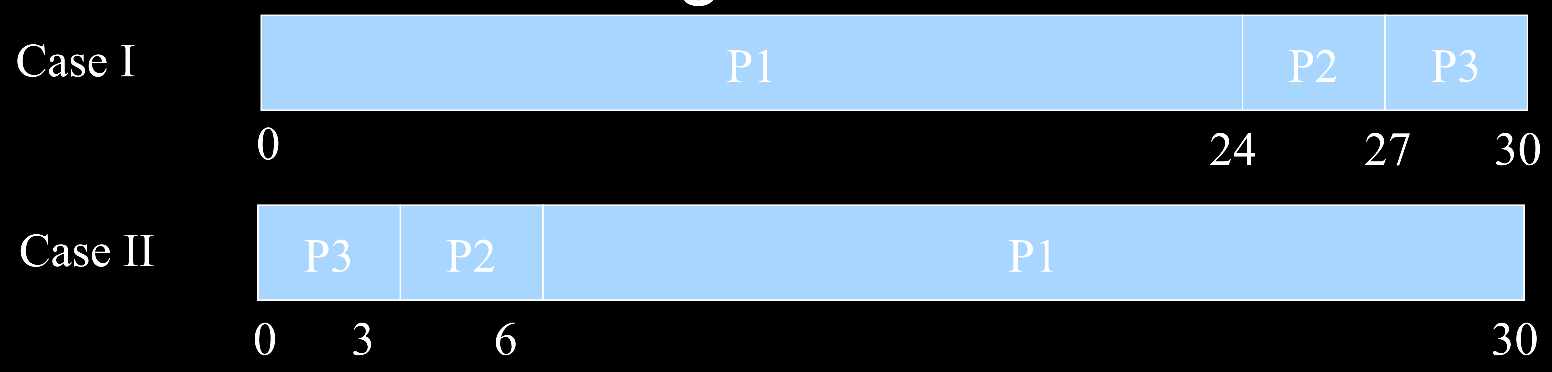
$(0 + 24 + 27)/3 = 17$

FCFS Scheduling



- Case I: average wait time is (0+24+27)/3 = 17 seconds
- Case II: average wait time is (0+3+6)/3 = 3 seconds
- FCFS wait times are generally not minimal - vary a lot if order of arrival changed, which is especially true if the process service times vary a lot (are spread out)

FCFS Scheduling



Lets calculate the ***average turnaround time*** for each of the cases

- Case I: average turnaround time is (24+27+30)/3 = 27 seconds
- Case II: average turnaround time is (3+6+30)/3 = 13 seconds

- A lot of variation in turnaround time too.

FCFS Scheduling

- Just pick the next process in the queue to be run?
  - No other information about the process is required

- Does it meet our goals?
  - maximize CPU utilization: 40% to 90%?
  - maximize throughput: # processes completed/second?
  - minimize average or peak wait, turnaround, or response time?
  - meet deadlines or delay guarantees?
  - maximize fairness?

Department of Computer Science
UNIVERSITY OF COLORADO BOULDER

Design and Analysis of Operating Systems CSCI 3753

Dr. David Knox
University of Colorado Boulder

These slides adapted from materials provided by the textbook authors.