Operating Systems!

Scheduling:
Uniprocessor Scheduling
(Part 1)

Prof. Travis Peters

Montana State University

CS 460 - Operating Systems

Fall 2020

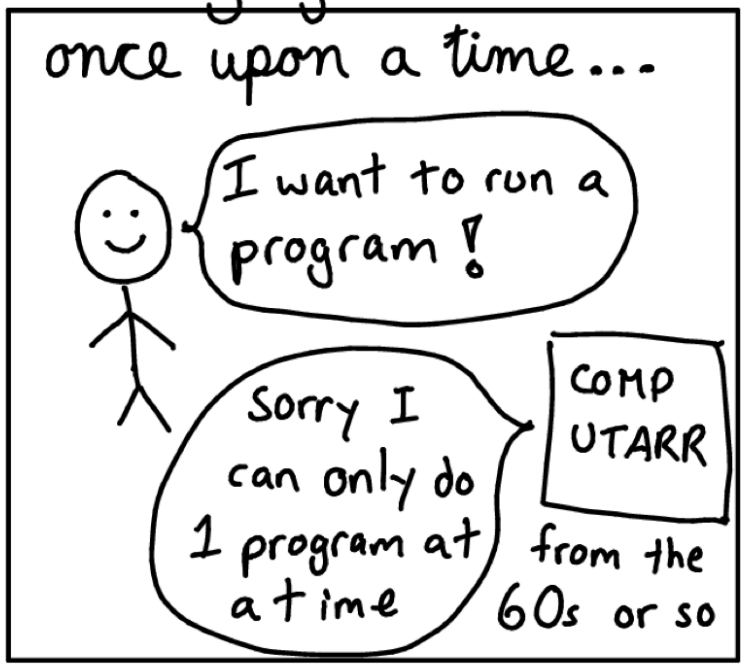
https://www.cs.montana.edu/cs460

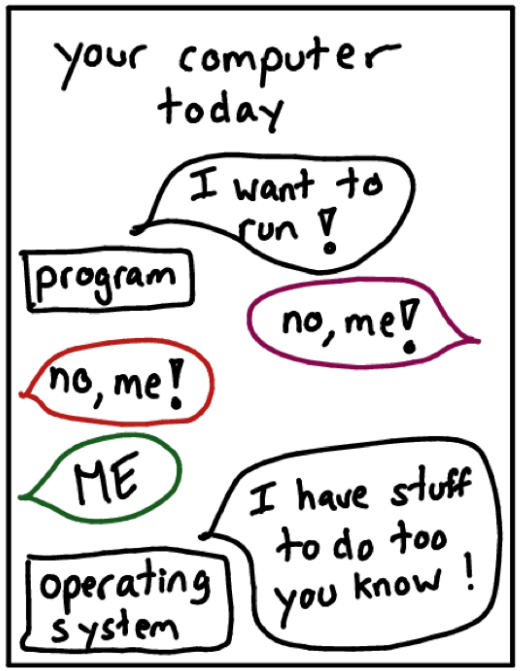
Some diagrams and notes used in this slide deck have been adapted from Sean Smith's OS courses @ Dartmouth. Thanks, Sean!

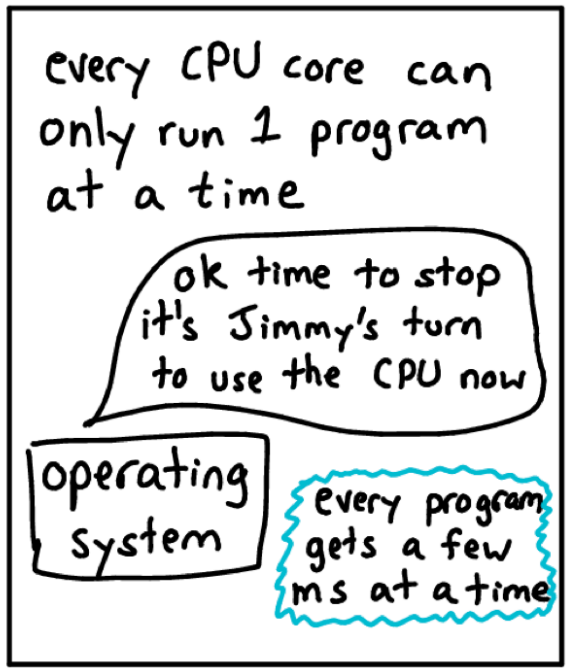
drawings.jvns.ca

CPU scheduling

Julia Evans
@b0rk







steps when we switch the running process

"context switch"

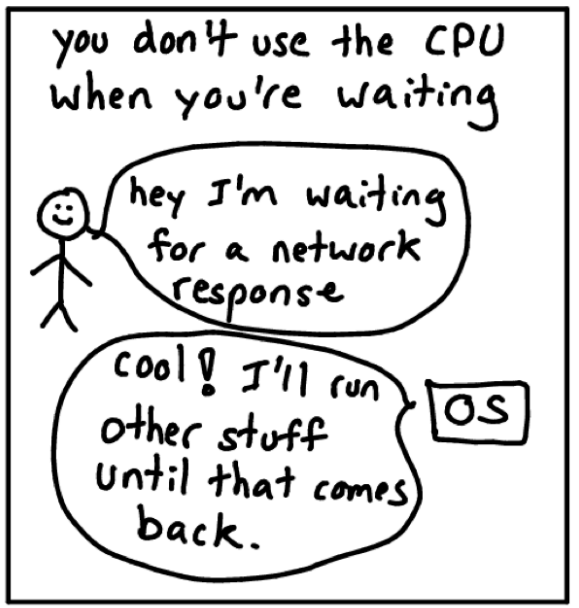
→ save:
- registers
- stack pointer
- which CPU instruction to start at next time

→ set up memory for new process

→ load new registers and stuff

all this takes time (2 microseconds?).

It's ok to do but you don't want to be switching processes constantly



https://drawings.jvns.ca/scheduling/

Today

- Announcements
  - ~~Yalnix~~ How are projects/proposals going?! ☺
  - *Congrats on submitting PA3!*

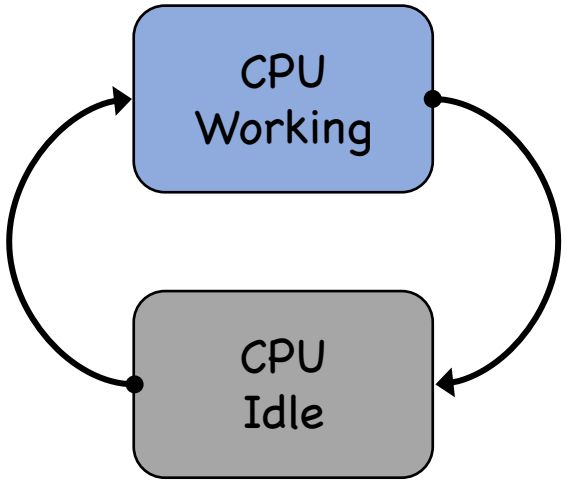
- Upcoming Deadlines
  - **Project Proposal (HARD DEADLINE) Sunday [10/25/2020] @ 11:59 PM (MST)**

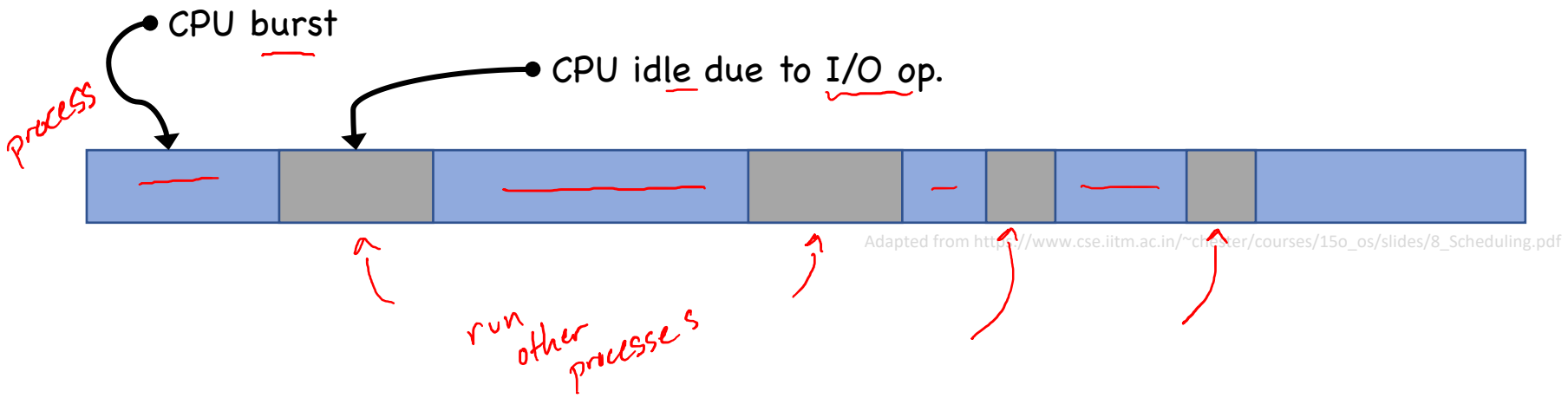
Yalnix Teams:
You should plan to be near completing checkpoint 3 at this point!

Non-Yalnix Teams:
Let me know if you want to discuss projects before submitting your proposal!

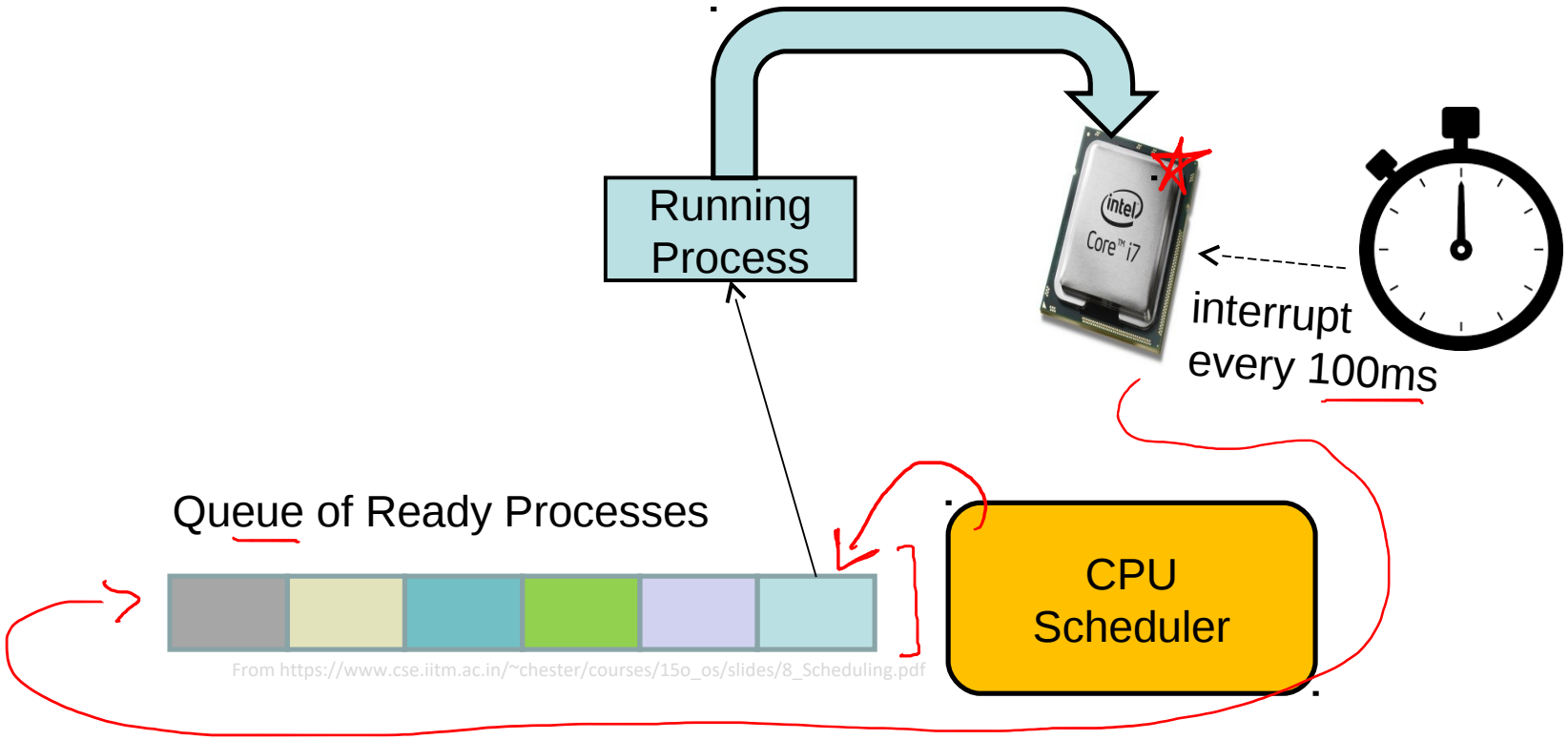
Today *(cont.)*

- Agenda
  - Types of Processes
  - Long-, medium-, and ***short-term*** scheduling
  - Performance of different scheduling policies





CPU Scheduler

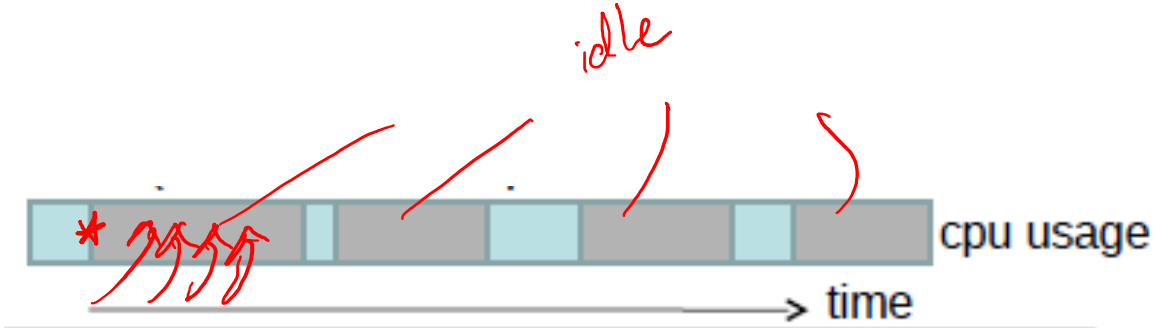


- Scheduler triggered to run when timer-based interrupt occurs, or when process is blocked on I/O
- Scheduler picks another process from the Ready Queue (RQ)
- Switch to new process via context switch

Types of Processes

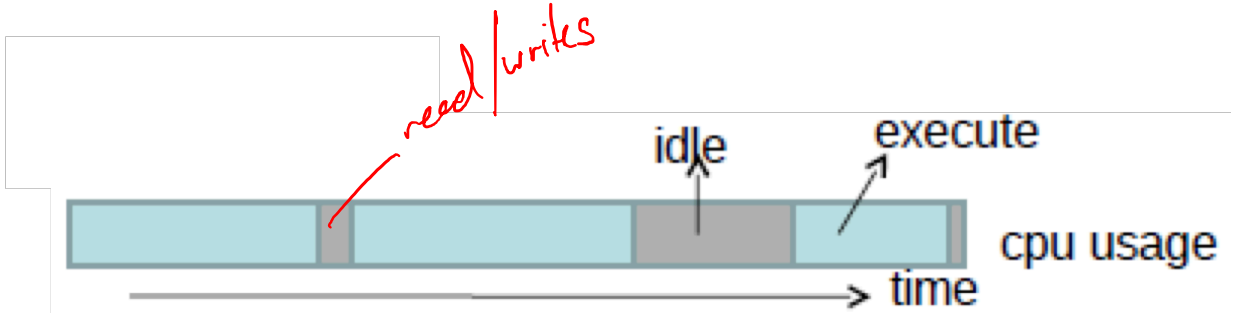
I/O Bound

- most work is I/O-related
- small bursts of CPU activity, then waits for I/O
- a process that primarily depends (and wait on) I/O operations
- affects user interaction should run with high-priority

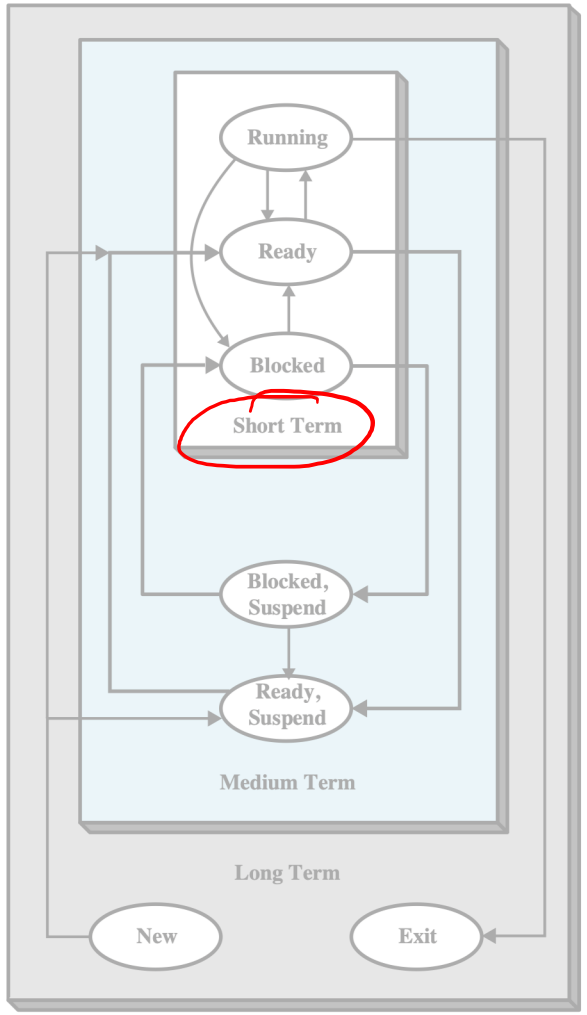


Processor Bound

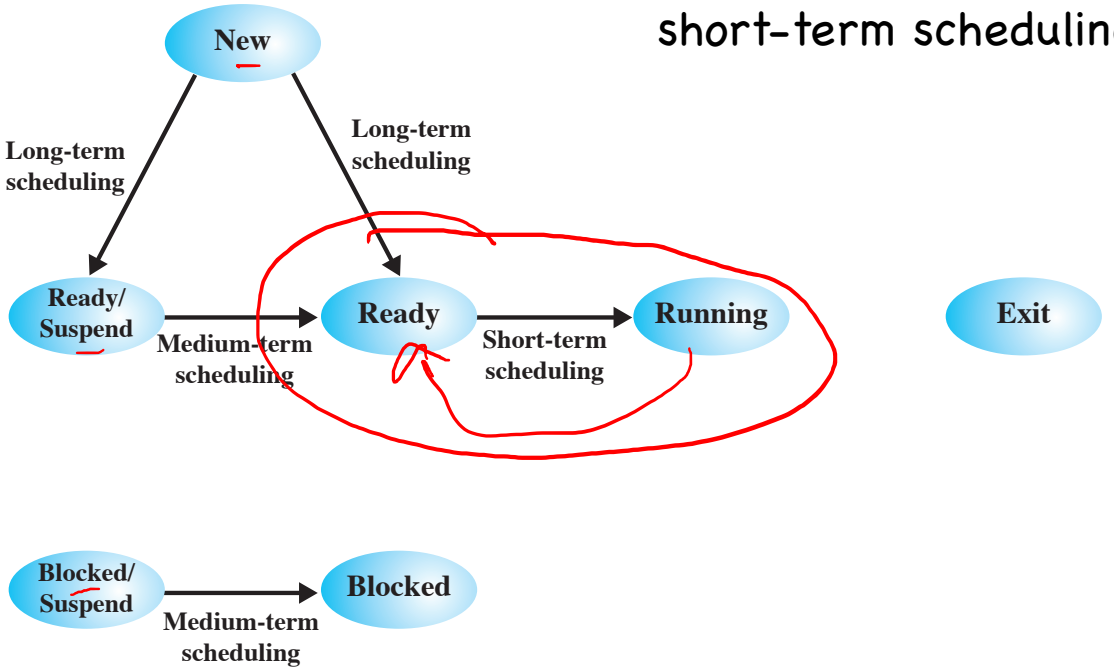
- most work is CPU-related
- hardly any I/O
- a process that primarily performs computational work
- e.g., gcc, scientific modeling, 3D rendering
- suitable for running with lower-priority



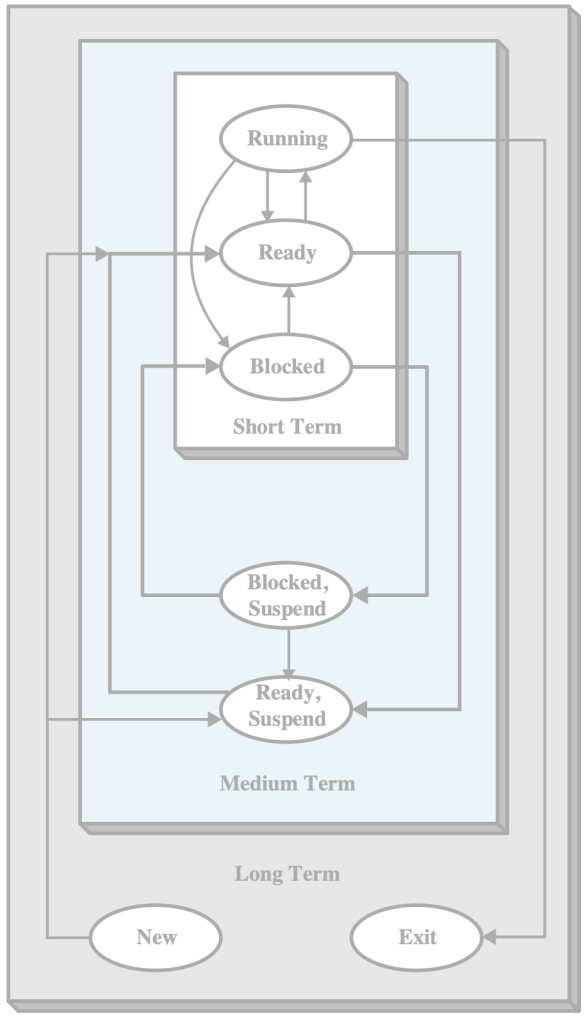
Levels of Scheduling



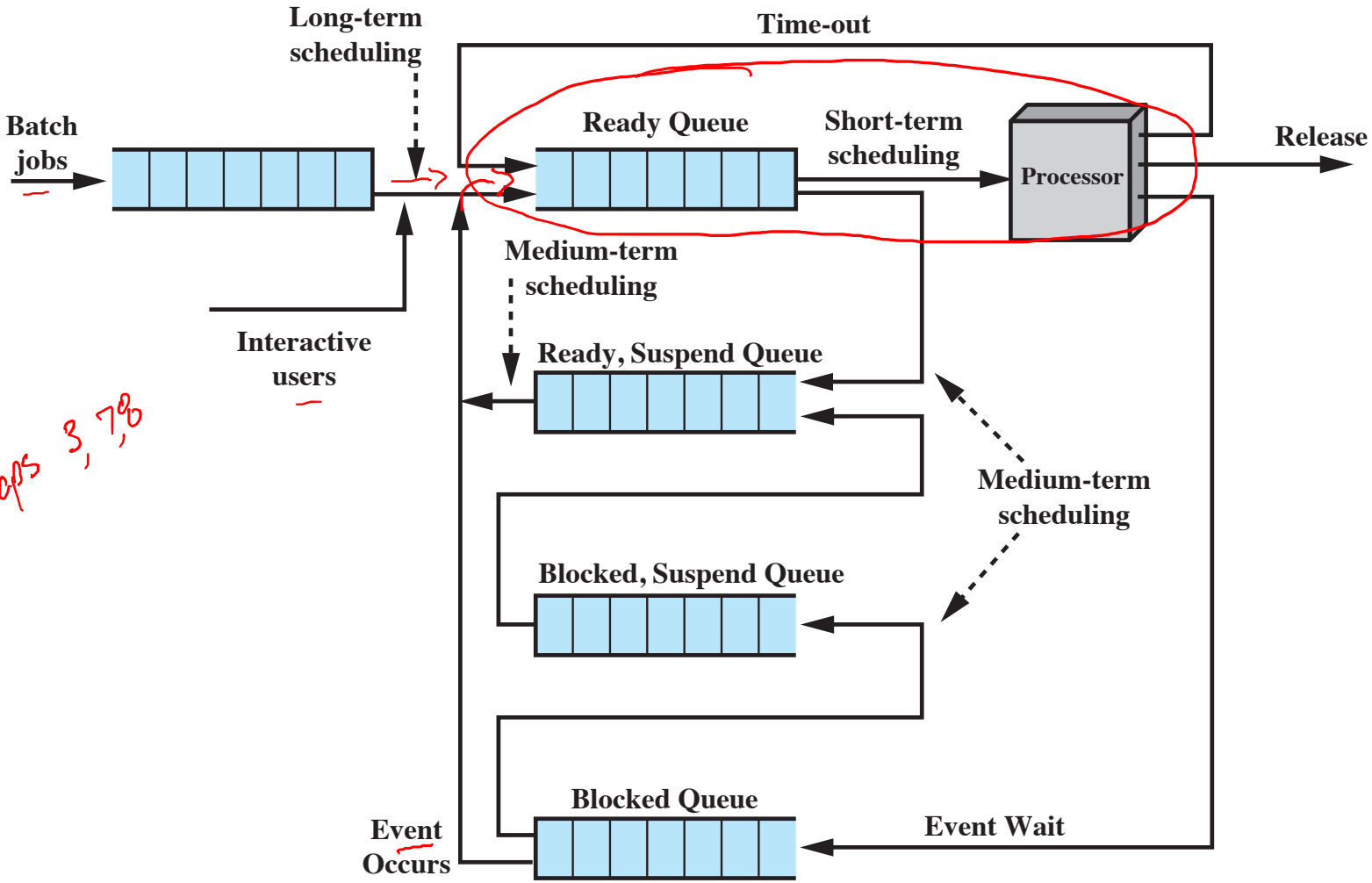
Recall process state transition diagram, now presented to illustrate long-, medium-, and short-term scheduling decisions



Levels of Scheduling



A revised queuing diagram for scheduling



Chaps 3, 7%

Scheduling Algorithms & Criteria

- Criteria for scheduling policies
  - We'll concentrate on short-term scheduling
  - **Goal:** allocate processor time to optimize 1+ aspects of system behavior

- *User*-oriented criteria vs. *System*-oriented criteria

responsiveness

avg. waiting time

proc. utilization → throughput

- *Performance*-oriented criteria vs. *Non-performance*-oriented criteria

quant.
response time
throughput

qual.
predictability

Scheduling Algorithms & Criteria *(cont.)*

Summary of Scheduling Metrics

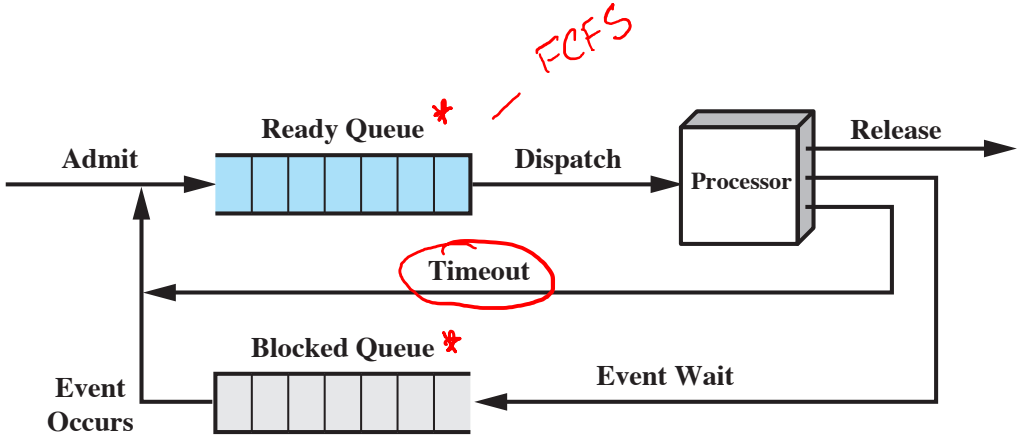
CPU Utilization Amount of time the CPU spends doing productive work *(utilization = 1 – idle_time)*
Throughput # of processes that are completed per time unit *(e.g., 54 procs per second)*
Turnaround Time Amount of time from process arrival to process completion *(TT = completion - arrival)*
Response Time How quickly a process starts running after submission *(response_time = start - arrival)*

Summary of Key Scheduling Criteria

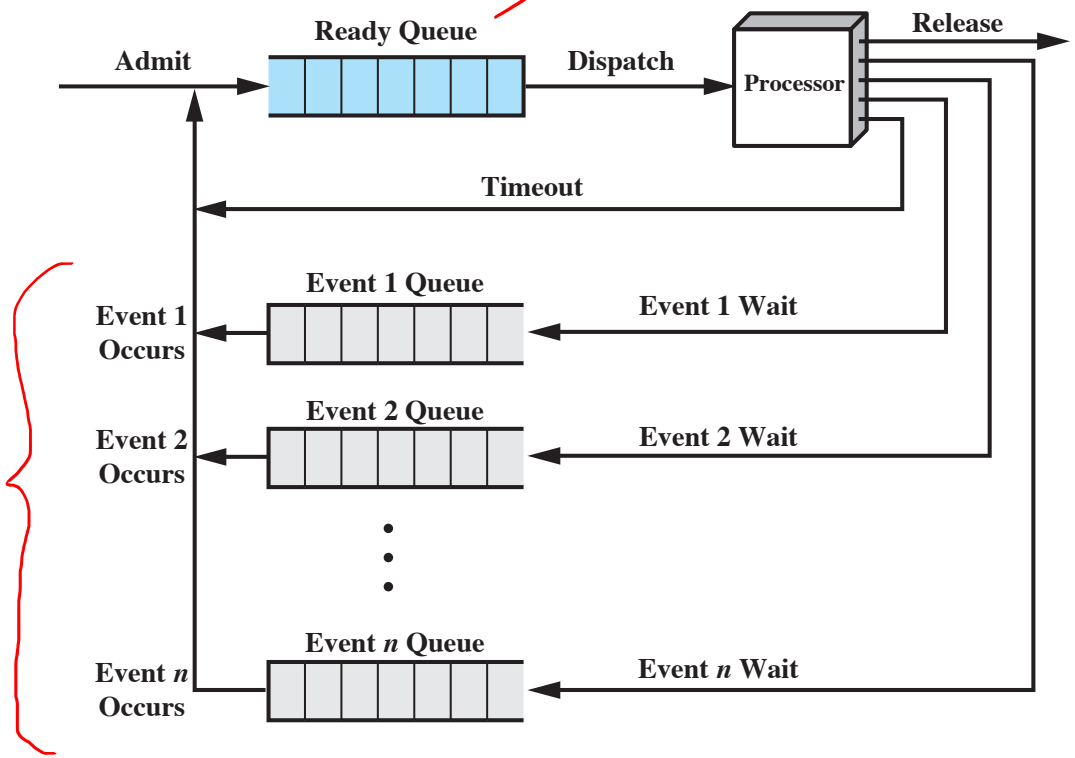
Turnaround time This is the interval of time between the submission of a process and its completion. Includes actual execution time plus time spent waiting for resources, including the processor. This is an appropriate measure for a batch job.
Response time For an interactive process, this is the time from the submission of a request until the response begins to be received. Often a process can begin producing some output to the user while continuing to process the request. Thus, this is a better measure than turnaround time from the user's point of view. The scheduling discipline should attempt to achieve low response time and to maximize the number of interactive users receiving acceptable response time.
Deadlines When process completion deadlines can be specified, the scheduling discipline should subordinate other goals to that of maximizing the percentage of deadlines met.
Predictability A given job should run in about the same amount of time and at about the same cost regardless of the load on the system. A wide variation in response time or turnaround time is distracting to users. It may signal a wide swing in system workloads or the need for system tuning to cure instabilities.
Throughput The scheduling policy should attempt to maximize the number of processes completed per unit of time. This is a measure of how much work is being performed. This clearly depends on the average length of a process but is also influenced by the scheduling policy, which may affect utilization.
Processor utilization This is the percentage of time that the processor is busy. For an expensive shared system, this is a significant criterion. In single-user systems and in some other systems, such as real-time systems, this criterion is less important than some of the others.
Fairness In the absence of guidance from the user or other system-supplied guidance, processes should be treated the same, and no process should suffer starvation.
Enforcing priorities When processes are assigned priorities, the scheduling policy should favor higher-priority processes.
Balancing resources The scheduling policy should keep the resources of the system busy. Processes that will underutilize stressed resources should be favored. This criterion also involves medium-term and long-term scheduling.

How should the scheduler select a process to run?

Ready. Or Ready Not.



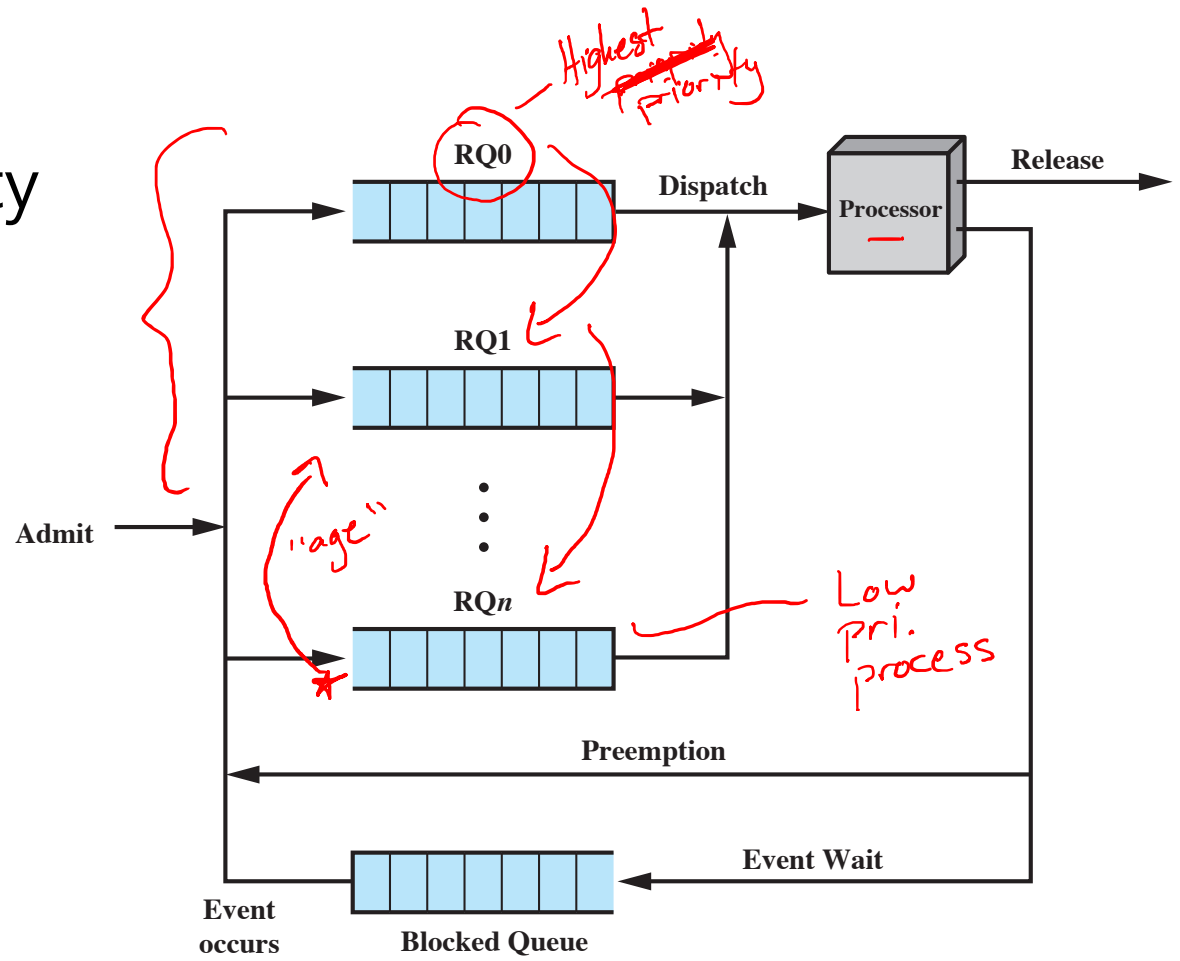
(a) Single blocked queue



(b) Multiple blocked queues

Using Priorities for Scheduling

- Each process is assigned a priority
  Smaller # = higher priority, Bigger # = lower priority (typically...)
- Scheduler selects process from higher priority queues
  Check lower priority queues successively
- Q: Any Potential Issues?



NOTE: multiple blocked queues and suspended states are not shown here to simplify the diagram... but you'd want those too! ☺

Comparing Scheduling Algorithms

Characteristics of Various Scheduling Policies

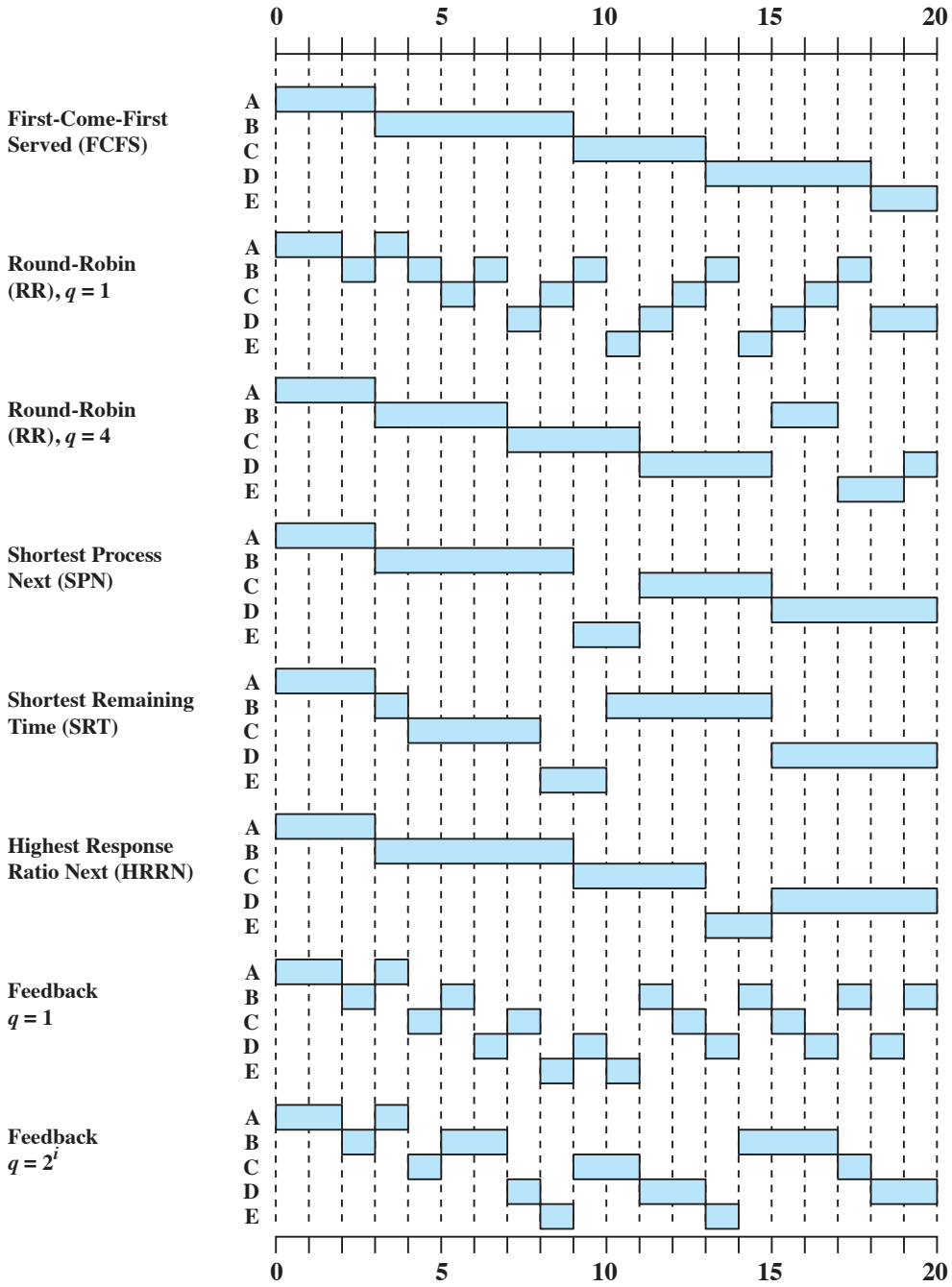
	FCFS	Round robin	SPN	SRT	HRRN	Feedback
Selection Function	max[w]	constant	min[s]	min[$s - e$]	max$\left(\frac{w+s}{s}\right)$	(see text)
Decision Mode	Non-preemptive	Preemptive (at time quantum)	Non-preemptive	Preemptive (at arrival)	Non-preemptive	Preemptive (at time quantum)
Throughput	Not emphasized	May be low if quantum is too small	High	High	High	Not emphasized
Response Time	May be high, especially if there is a large variance in process execution times	Provides good response time for short processes	Provides good response time for short processes	Provides good response time	Provides good response time	Not emphasized
Overhead	Minimum	Minimum	Can be high	Can be high	Can be high	Can be high
Effect on Processes	Penalizes short processes; penalizes I/O bound processes	Fair treatment	Penalizes long processes	Penalizes long processes	Good balance	May favor I/O bound processes
Starvation	No	No	Possible	Possible	No	Possible

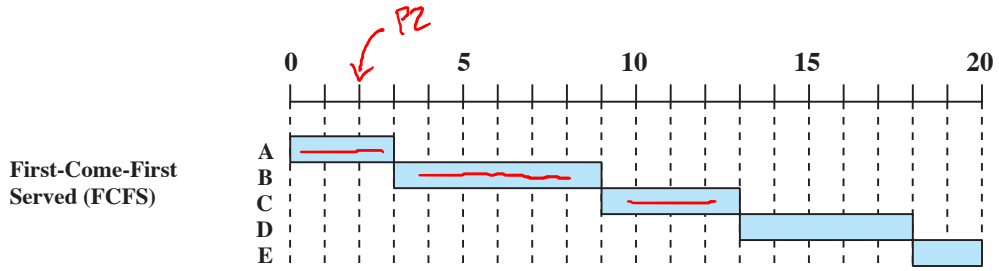
w = time spent WAITING, e = time spent EXECUTING, s = TOTAL service time

Comparing Scheduling Algorithms

An Example

Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2





Process	A	B	C	D	E	
Arrival Time	0	2	4	6	8	
Service Time (*Ts*)	3	6	4	5	2	Mean
FCFS						
Finish Time	3	9	13	18	20	
Turnaround Time (*Tr*)	3	7	9	12	12	8.60
Tr/*Ts*	1.00	1.17	2.25	2.40	6.00	2.56

First-Come-First-Served (FCFS)

- Strict queueing scheme (a.k.a. FIFO)
- The process that has been in the ready queue the longest is selected next

- **Q:** What happens when a short process arrives just after a long process?

- Not a great solution on its own for a uniprocessor system.
  Q: Why?