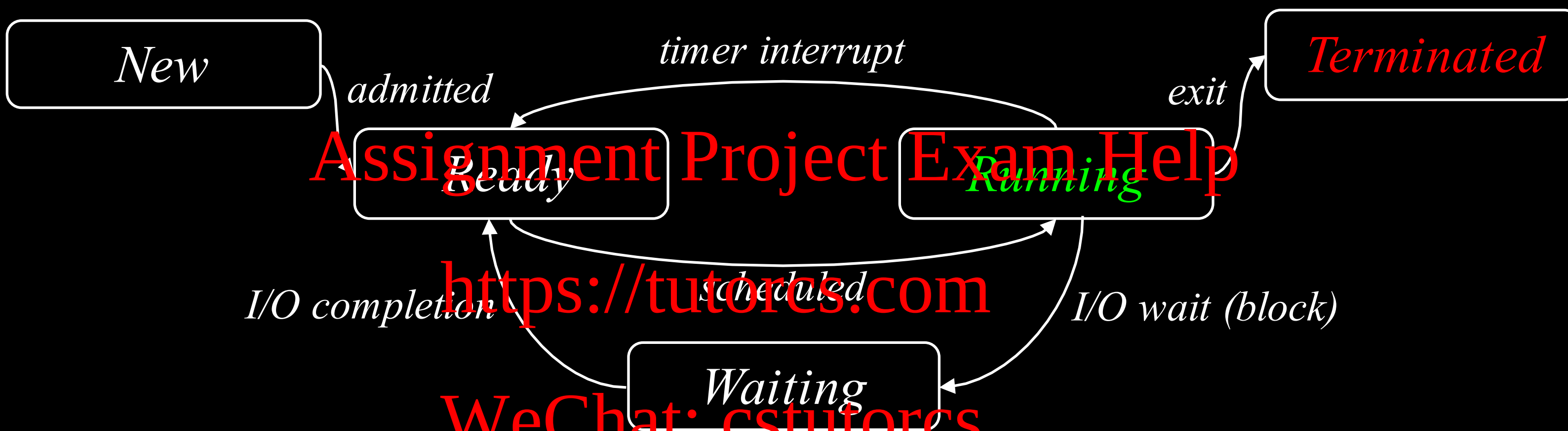
COSI 131a: Operating Systems

CPU Scheduling (1)

Chapter 6

Review: Process States

Processes Not Always Running



New: *Process is being created*

Running: *Instructions are being executed*

Waiting: *Process is waiting for some event to occur*

Ready: *Process is waiting to be assigned the CPU*

Terminated: *Process has finished execution*

Agenda

1. Scheduling Overview
   1. *First Example of Resource (CPU) Management (Sharing)*
   2. *Non-Preemptive (N) vs. Preemptive Scheduling (P)*
   3. *Metrics: Ways to Assess Effectiveness of Scheduling Policies*

2. Scheduling Policies
   1. *First-Come-First-Served (FCFS) (N)*
   2. *Shortest-Job-First (SJF) (N, P)*
   3. *Priority (N, P)*
   4. *Round-Robin (RR) (P)*
   5. *Multilevel Queues (MLQ) (P) , Lottery*
   6. *Real-time*

3. Examples

Basic Concepts

- One processor -> one process running at a time
  - *All other processers wait for CPU to be free*
  - *Often running process does need CPU (e.g., I/O requests)*

- Multiprogramming goal: avoid CPU idle times-> some process is using the CPU all the time

  - *Keep multiple processes in memory at one time*

  - *When one process has to wait the CPU is allocated to another process*
  - *Scheduling algorithms: decide which process goes into the CPU*

CPU-I/O Burst Cycle

A Typical Process Execution Flow

```
...
LOAD R1, …
LOAD R2, …
ADD R1, R2
read from file
```

CPU Burst

WAIT FOR I/O

I/O Burst

```
LOAD …
STORE …
INCREMENT …
write to file
```

CPU Burst

WAIT FOR I/O

…

I/O Burst

alternates between CPU and I/O bursts

∴ *scheduler invoked very frequently*

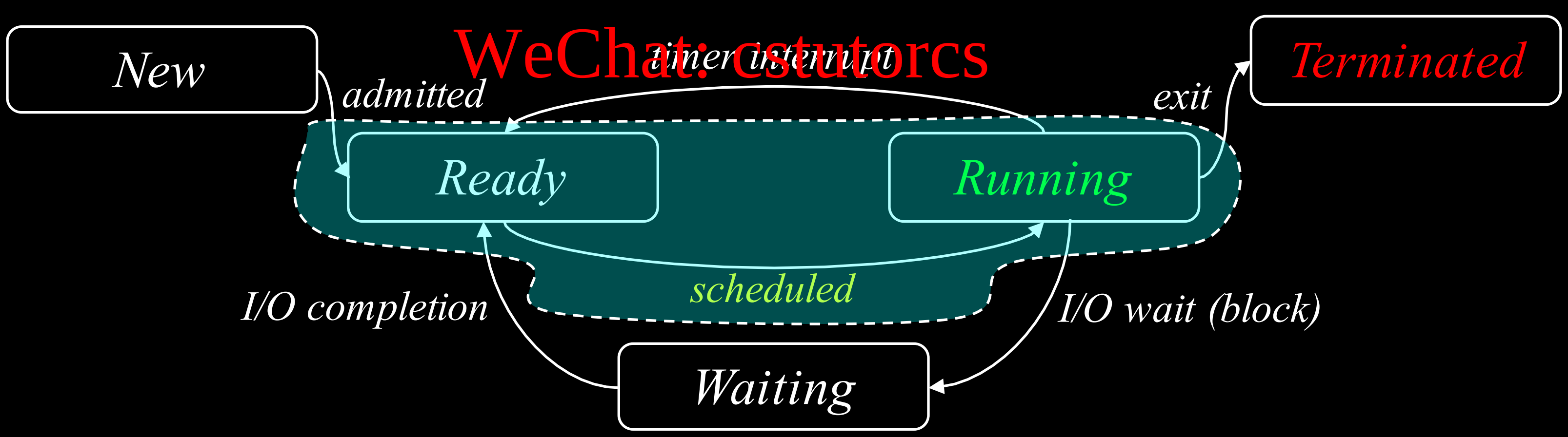
∴ *scheduler must be very fast*

Distribution of CPU vs I/O bursts are important in selecting the right CPU scheduling approach

CPU Scheduler

CPU (Short-Term) Scheduler

- *first example of resource <u>manager</u> (manages sharing of CPU)*
- *applicable to either multiprocessing or multithreading (we'll assume former)*
- *What? A kernel process that is always active (daemon)*
- *responsible for choosing process to run from ready queue*



Agenda

1. Scheduling Overview
   1. *First Example of Resource (CPU) Management (Sharing)*
   2. *Non-Preemptive (N) vs. Preemptive Scheduling (P)*
   3. *Metrics: Ways to Assess Effectiveness of Scheduling Policies*

2. Scheduling Policies

   1. *First-Come-First-Served (FCFS) (N)*
   2. *Shortest-Job-First (SJF) (N, P)*

   3. *Priority (N, P)*
   4. *Round-Robin (RR) (P)*
   5. *Multilevel Queues (MLQ) (P) , Lottery*
   6. *Real-time*

3. Examples

CPU Scheduling

When Does the Scheduler Get Invoked?

1. *when there's an interrupt*
- *from timer: moves process from running to ready state*

- *upon I/O interrupt: moves process from wait to ready state*

1. *when running process terminates*

2. *when running process issues an I/O request (block) and moves to wait state*

CPU Scheduling

Two Types of Scheduling:

Non-Preemptive: processes only give up CPU voluntarily

- *simpler to implement (no timer interrupts)*
- *greedy or buggy process can starve others*

Preemptive: processes also may be preempted by an interrupt

- *adds complexity*
- *adds protection, better at ensuring fairness, as well as doing better in other scheduling metrics*

CPU Scheduling Policies

1. First Come First Served
   - *Non-preemptive*
2. Shortest Job First
   - *a) Non-preemptive and b) Preemptive*
3. Priority Scheduling
   - *a) Non-preemptive and b) Preemptive*
4. Round-Robin
   - *Preemptive*
5. Multilevel Queues, Lottery
   - *Preemptive*

How to compare these?

- *requires appropriate scheduling metrics*

Agenda

1. Scheduling Overview
   1. *First Example of Resource (CPU) Management (Sharing)*
   2. *Non-Preemptive (N) vs. Preemptive Scheduling (P)*
   3. *Metrics: Ways to Assess Effectiveness of Scheduling Policies*

2. Scheduling Policies

   1. *First-Come-First-Served (FCFS) (N)*
   2. *Shortest-Job-First (SJF) (N, P)*

   3. *Priority (N, P)*
   4. *Round-Robin (RR) (P)*
   5. *Multilevel Queues (MLQ) (P) , Lottery*
   6. *Real-time*

3. Examples

Scheduling Metrics

Typical Metrics for Comparing Scheduling Policies:

1. *Throughput*: *measured in # processes completed / time unit*
2. *Turnaround Time*: *average (time of completion – time of creation)*
   - *related metric: Waiting Time (average time on ready queue)*
   - *Waiting vs Turnaround: no penalty for processes w/ long processing times (Turnaround Time = Processing /Blocked Time + Waiting Time)*

3. *Response Time*: *time of 1st response is produced– time of creation*

4. *Overhead*: *time spent related to scheduling (e.g., context switch time)*
5. *Fairness*: *how much variation in waiting time*
   a) *minimal*
   b) *proportional (to processing time)*
   c) *never infinite (aka: no starvation)*

Can't Meet All Performance Goals With Any 1 Policy

Scheduling Performance Metrics

How Do We Measure Effectiveness of Scheduling Policies?

1. CPU Utilization

   What percentage of time is the CPU doing useful work? (Good = High)

2. Throughput:

   Number of processes that complete execution per time unit (Good = High)

3. Turnaround time:

   Amount of time to execute a particular process. Includes time in ready and waiting queues (Good = Low)

4. Waiting time:

   Average time process spends in Ready Queue (Good = Low)

5. Response time:

   Average time before process produce first response after request (Good = Low)

Scheduling Metrics

Can't Meet All Performance Goals With 1 Policy

$\therefore$ *Pick 1 or 2 that matter most*

Typically Important Metrics:

1. *Throughput / Turnaround time*:

   Throughput = n processes / sec $\Rightarrow$ *Turnaround = 1/n sec per process*

2. *Waiting time*

3. *Response time*: *(especially for highly interactive systems)*

Also Care About:

1. *Dispatch time*:

   Time it takes to choose next running process including schedule time + context switch time. If lengthy, effects effectiveness of scheduler

2. *Fairness*: *Want to avoid process starvation*

Agenda

1. Scheduling Overview
    1. *First Example of Resource (CPU) Management (Sharing)*
    2. *Non-Preemptive (N) vs. Preemptive Scheduling (P)*
    3. *Metrics: Ways to Assess Effectiveness of Scheduling Policies*

2. Scheduling Policies

    1. *First-Come-First-Served (FCFS) (N)*
    2. *Shortest-Job-First (SJF) (N, P)*

    3. *Priority (N, P)*
    4. *Round-Robin (RR) (P)*
    5. *Multilevel Queues (MLQ) (P)*
    6. *Multiprocessor, energy aware, real time, overload*
3. Examples

Schedule Policy #1:

FCFS: First Come First Served

1. First-Come First-Serve (FCFS)

Non-Preemptive Policy

Example

Process	CPU Burst Time
P_1	24
P_2	3
P_3	3

Suppose processes arrive in order: P_1; P_2; P_3

(*Gantt Chart*)

Waiting Time

P_1 : 0
P_2 : 24
P_3 : 27

Average Wait Time = 17

1. First-Come First-Serve (FCFS)

Non-Preemptive Policy

Example

Process	CPU Burst Time
P_1	24
P_2	3
P_3	3

Suppose processes arrive in order: P_2; P_3; P_1

Waiting Time

P_1 : 6
P_2 : 0
P_3 : 3

Average Wait Time = 3

Waiting time determined by order of arrival

1. First-Come First-Serve (FCFS)

Non-Preemptive Policy

+: *Easy to implement (ready queue = scheduling queue)*

No starvation

–: *Convoy Effect: Order of arrival determines performance*

e.g., many processes might end up waiting for a big one to get off the CPU

(Food for thought: maybe shortest process should go first instead?)

CPU Scheduling Policies Summary

Policy	***Throughput***	***Waiting***	***Response***	***Fairness***	***Overhead***	***Comments***
FCFS	✗	✗	✗	✓	✓	

Schedule Policy #2: SJF: Shortest Job First

2a. Shortest-Job-First (SJF)

Non-Preemptive Policy

Idea: *Rank processes by CPU time requests.*
Optimizes (minimizes) average waiting time

Example

Process	CPU Burst
P_1	24
P_2	3
P_3	3
P_4	7

P_2	P_3	P_4	P_1
0–3	3–6	6–13	13–37

Average wait = (0 + 3 + 6 + 13) / 4 = 5.5

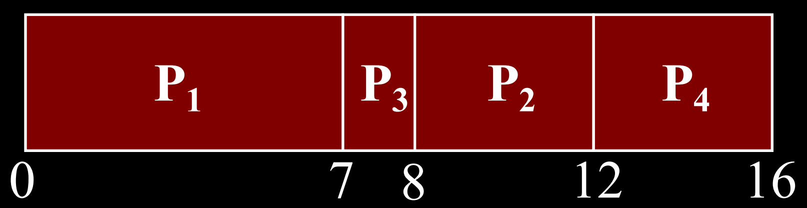
2a. Shortest-Job-First (SJF)

Q: What if processes don't arrive at same time?

A: *Whenever process finishes, choose next process with shortest CPU burst*

Example

Process	Arrival Time	CPU Burst
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4



Average wait time: ?

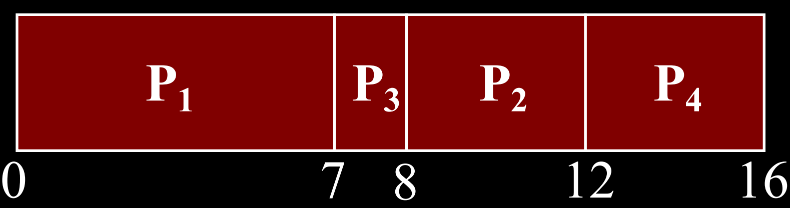
2a. Shortest-Job-First (SJF)

Q: What if processes don't arrive at same time?

A: Whenever process finishes, choose next process with shortest CPU burst

Example

Process	Arrival Time	CPU Burst
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4



Average wait time: (0+6+3+7) / 4 = 4

Preemptive version of SJF does even better …

2b. Preemptive SJF

Preemptive Version of SJF

Idea: *Currently running process can be preempted if new process arrives with shorter remaining CPU burst*

Example

Process	Arrival Time	CPU Burst
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

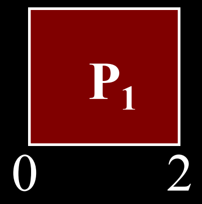
2b. Preemptive SJF

Preemptive Version of SJF

Idea: *Currently running process can be preempted if new process arrives with shorter remaining CPU burst*

Example

Process	Arrival Time	CPU Burst
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4



2b. Preemptive SJF

Preemptive Version of SJF

Idea: *Currently running process can be preempted if new process arrives with shorter remaining CPU burst*

Example

Process	Arrival Time	CPU Burst
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

P_1

0 2

$P_1 = 5, \; P_2 = 4$

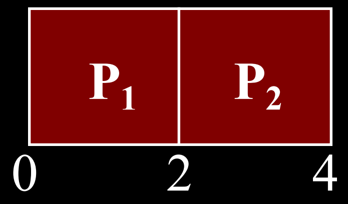
2b. Preemptive SJF

Preemptive Version of SJF

Idea: *Currently running process can be preempted if new process arrives with shorter remaining CPU burst*

Example

Process	Arrival Time	CPU Burst
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4



$P_1 = 5, \; P_2 = 4$

$P_1 = 5, \; P_2 = 2, \; P_3 = 1$

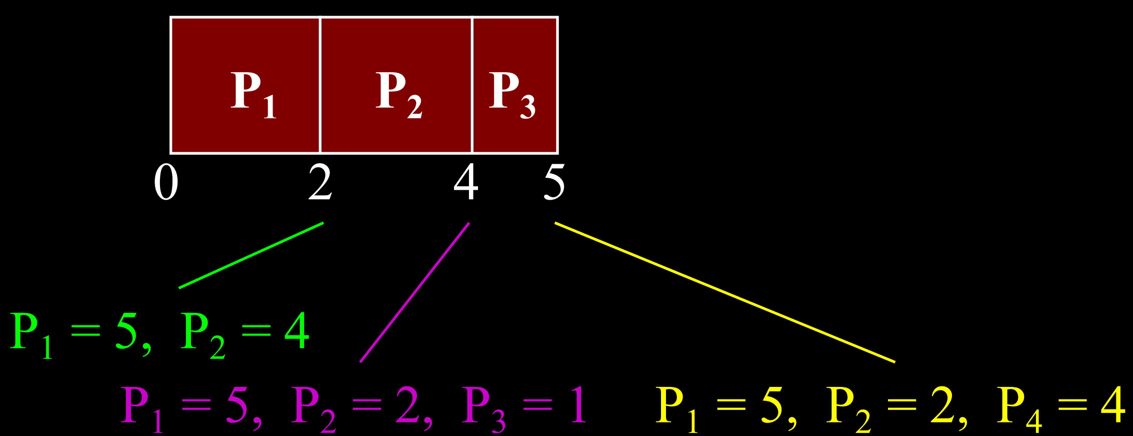
2b. Preemptive SJF

Preemptive Version of SJF

Idea: *Currently running process can be preempted if new process arrives with shorter remaining CPU burst*

Example

Process	Arrival Time	CPU Burst
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4



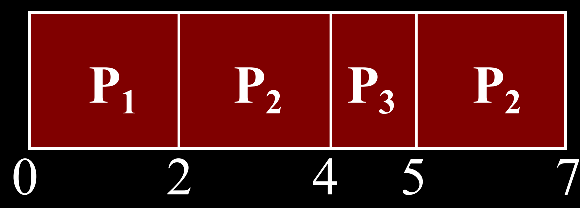
2b. Preemptive SJF

Preemptive Version of SJF

Idea: *Currently running process can be preempted if new process arrives with shorter remaining CPU burst*

Example

Process	Arrival Time	CPU Burst
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4



$P_1 = 5,\ P_2 = 4$

$P_1 = 5,\ P_2 = 2,\ P_3 = 1$ $P_1 = 5,\ P_2 = 2,\ P_4 = 4$ $P_1 = 5,\ P_4 = 4$

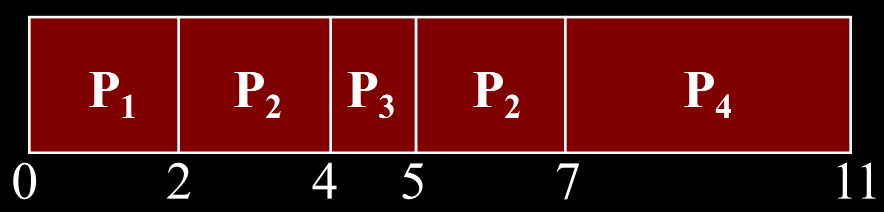
2b. Preemptive SJF

Preemptive Version of SJF

Idea: *Currently running process can be preempted if new process arrives with shorter remaining CPU burst*

Example

Process	Arrival Time	CPU Burst
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4



$P_1 = 5,\ P_2 = 4$

$P_1 = 5,\ P_2 = 2,\ P_3 = 1$ $P_1 = 5,\ P_2 = 2,\ P_4 = 4$ $P_1 = 5,\ P_4 = 4$

$P_1 = 5$

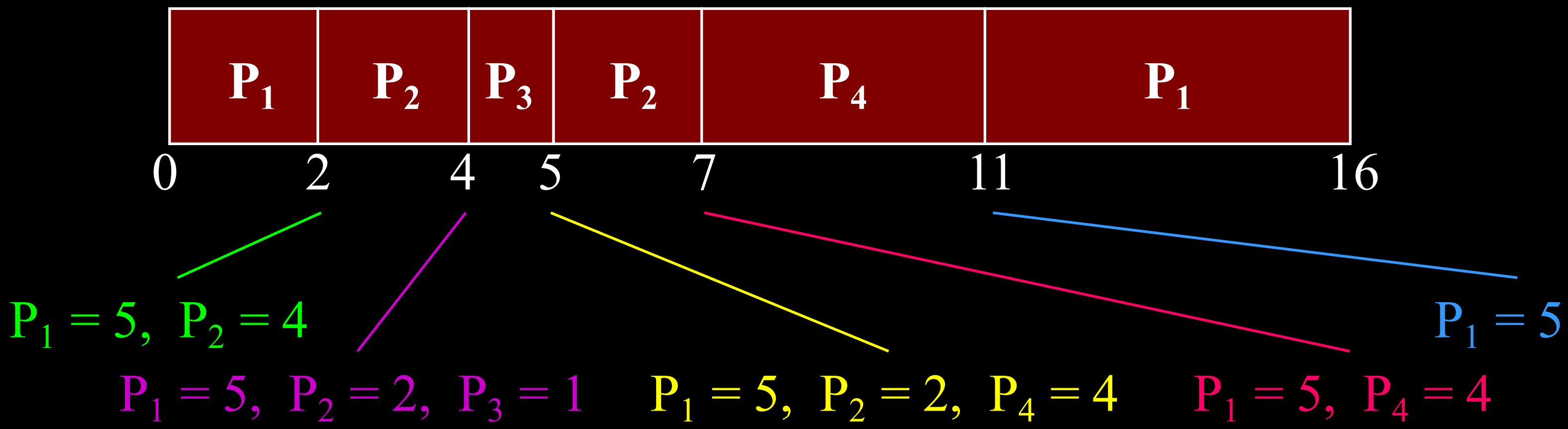
2b. Preemptive SJF

Preemptive Version of SJF

Idea: *Currently running process can be preempted if new process arrives with shorter remaining CPU burst*

Example

Process	Arrival Time	CPU Burst
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4



2b. Preemptive SJF

Example:

Process	Arrival Time	CPU Burst
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

Average wait time:

P_1 : 9
P_2 : 1
P_3 : 0
P_4 : 2

Average = 3

(Compare to non-preemptive SJF average = 4)