

CPU Scheduling

Hui Chen ^a

^aCUNY Brooklyn College

April 15, 2021

Outline

- 1 CPU Scheduling
- 2 Scheduling Criteria
- 3 Scheduling Algorithms
- 4 Thread Scheduling and Multiprocessor Scheduling

Outline

- 1 CPU Scheduling
- 2 Scheduling Criteria
- 3 Scheduling Algorithms
- 4 Thread Scheduling and Multiprocessor Scheduling

Recall Process Queues and State Transitions

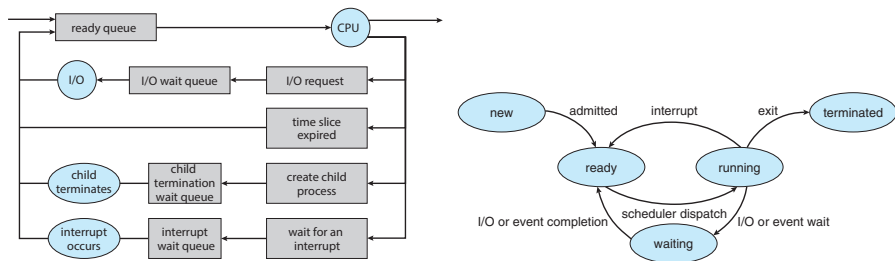


Figure: Process queues and transitions¹.

- ▶ CPU scheduling is the basis of multiprogrammed operating systems, it is about selecting a task from the *Ready* queue to execute it on CPU.
 - ▶ Process scheduling vs. thread scheduling

¹Silberschatz, Galvin, and Gagne, *Operating system concepts*.

When does scheduling happens?

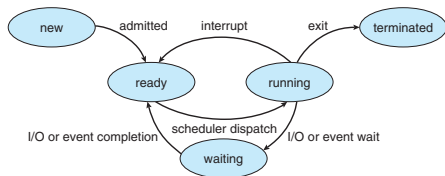
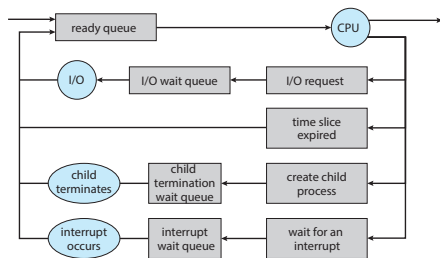


Figure: Process queues and transitions².

- ▶ CPU scheduling is about selecting a task from the *Ready* queue to execute it on CPU, but when does the OS make such an action?

²Silberschatz, Galvin, and Gagne, *Operating system concepts*.

Preemptive vs. Non-preemptive Scheduling

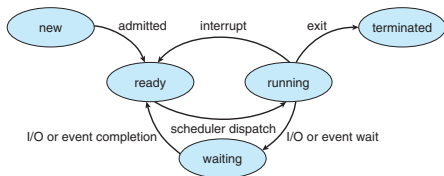
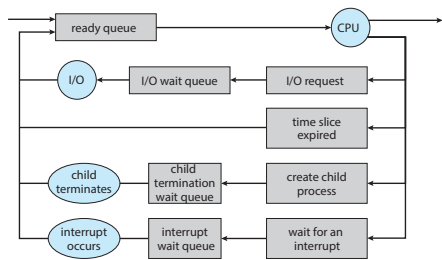


Figure: Process queues and transitions³.

Consider when a task goes into the *Ready* queue, or a task is off CPU,

1. Running \rightarrow Waiting
2. Running \rightarrow Ready
3. Waiting \rightarrow Ready
4. * \rightarrow Terminated

³Silberschatz, Galvin, and Gagne, *Operating system concepts*.

Preemptive vs. Non-preemptive Scheduling

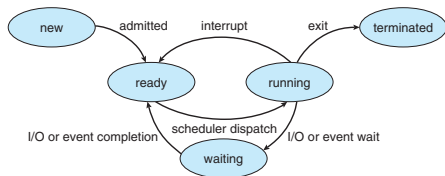
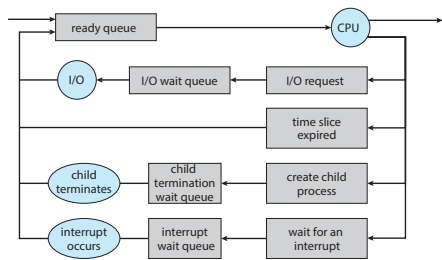


Figure: Process queues and transitions³.

Under 1 and 4, nonpreemptive or cooperative; otherwise, preemptive.

1. Running \rightarrow Waiting

3. Waiting \rightarrow Ready

2. Running \rightarrow Ready

4. * \rightarrow Terminated

³Silberschatz, Galvin, and Gagne, *Operating system concepts*.

Scheduling and Context Switch

1. CPU scheduler makes the decision and select a task from the *Ready* queue.
2. Dispatcher gives the control of the CPU to the selected task.
 - 2.1 Switching context from the active task to the selected CPU.
 - 2.2 Switching to user mode
 - 2.3 Jumping to the proper location in the selected task to resume that task

Monitoring System Context Switching in Linux

```

1 -$ vmstat 1 3 # show vm statistics every 1 second for 3 times
2 procs -----memory----- ---swap-- -----io----- -system-- -----cpu-----
3  r  b   swpd   free   buff  cache   si   so    bi    bo    in   cs  us  sy  id  wa  st
4  0  0   1340 123188 112552 716608   0   0    0    2    4   10  0  0 100  0  0
5  0  0   1340 123188 112552 716640   0   0    0    0  576   74  0  0 100  0  0
6  0  0   1340 123188 112552 716640   0   0    0    0  576   68  0  0 100  0  0
7 -$

```

▶ `man vmstat`

`system`

`cs`: The number of context switches per second.

Monitoring Process Context Switch in Linux

```

1  ~$ for p in /proc/[0-9]*; do \
2  > echo "Process ${p#/proc/}:"; \
3  > while read ln; do \
4  > echo -e "\t${ln}"; done <<< $(grep -E -o "\.*_ctxt_switches.*$" ${p}/status); \
5  > done;
6  Process 1:
7     voluntary_ctxt_switches:          41215
8     nonvoluntary_ctxt_switches:      15741
9  Process 10:
10    voluntary_ctxt_switches:         26239510
11    nonvoluntary_ctxt_switches:      10
12    ...
13  Process 99:
14    voluntary_ctxt_switches:          4
15    nonvoluntary_ctxt_switches:      0
16  ~$

```

- ▶ A voluntary context switch occurs when a task has given up control of the CPU because it requires a resource that is currently unavailable (such as blocking for I/O.)
- ▶ A nonvoluntary context switch occurs when the CPU has been taken away from a task, such as when its time slice has expired or it has been preempted by an another task.

Outline

- 1 CPU Scheduling
- 2 Scheduling Criteria**
- 3 Scheduling Algorithms
- 4 Thread Scheduling and Multiprocessor Scheduling

Not all processes are created equal

The design or selection of CPU scheduling algorithm depends on an observed property of processes:

- ▶ CPU burst and I/O burst cycle
- ▶ Distribution of CPU and I/O bursts
- ▶ CPU-bound processes
- ▶ I/O-bound processes

Scheduling Criteria

Criteria from design or selection of CPU scheduling algorithm

- ▶ CPU utilization. % of time CPU being busy
- ▶ Throughput. # of tasks completed per time unit.
- ▶ Turnaround time. Interval from task submission to task completion.
- ▶ Waiting time. Total time a task spends (i.e., waits) in the ready queue.
- ▶ Response time. Interval from the submission of a request until the first response is produced.

Optimizing for Scheduling Criteria

Maximize CPU utilization and throughput and to minimize turnaround time, waiting time, and response time.

- ▶ Consider min, max, average, variance ...
- ▶ Criteria may conflict with each other
- ▶ Batch systems vs. interactive systems vs. real-time systems vs. ...

Outline

- 1 CPU Scheduling
- 2 Scheduling Criteria
- 3 Scheduling Algorithms**
- 4 Thread Scheduling and Multiprocessor Scheduling

Scheduling Algorithms

- ▶ First-Come, First-Served Scheduling
- ▶ Shortest-Job-First Scheduling
 - ▶ Shortest-Remaining-Time-First Scheduling
- ▶ Round-Robin Scheduling
- ▶ Priority Scheduling
 - ▶ Multilevel Queue Scheduling
 - ▶ Multilevel Feedback Queue Scheduling

First-Come, First-Served Scheduling (FCFS)

Consider the *ready* queue with the following tasks,

Task	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

Shortest-Job-First Scheduling (SJF)

Consider the *ready* queue with the following tasks,

Task	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

Shortest-Remaining-Time-First Scheduling (SRTF)

Consider the following tasks that arrive in the *ready* queue,

Task	Arrival Time	Burst Time
P_1	0	6
P_2	2	8
P_3	3	7
P_4	6	3

Estimating CPU Burst Time

- ▶ Do we know the CPU burst times at the time when we invoke the CPU scheduling algorithm?

Estimating CPU Burst Time

- ▶ Do we know the CPU burst times at the time when we invoke the CPU scheduling algorithm?
- ▶ How do we predict CPU burst times?

Estimating CPU Burst Time

- ▶ Do we know the CPU burst times at the time when we invoke the CPU scheduling algorithm?
- ▶ How do we predict CPU burst times?
The next CPU burst is generally predicted as an *exponential average* of the measured lengths of previous CPU bursts, commonly,

$$\tau_{n+1} = \alpha t_n + (1 - \alpha)\tau_n \quad (1)$$

where

t_n : the n -th CPU burst that the OS records

τ_{n+1} : the next (i.e., $n+1$) predicted value of the CPU burst

α : $\alpha \in [0, 1]$, an aging exponent that determines the effect of history of CPU bursts.

Round-Robin Scheduling (RR)

Assume time quantum = 2 and consider the *ready* queue with the following tasks,

Task	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

Priority Scheduling

Consider the *ready* queue with the following tasks where 1 means the highest priority, and 3 lowest,

Task	Burst Time	Priority
P_1	6	3
P_2	8	1
P_3	7	2
P_4	3	1

Outline

- 1 CPU Scheduling
- 2 Scheduling Criteria
- 3 Scheduling Algorithms
- 4 Thread Scheduling and Multiprocessor Scheduling

Thread Scheduling

- ▶ Kernel and user threads
- ▶ Contention scope

Multiprocessor Scheduling

- ▶ Multiprocess architecture, multicore CPUs vs. multithreaded cores vs. NUMA systems vs. Heterogeneous multiprocessing
- ▶ Common ready queue vs. per-core ready queue
- ▶ Load balancing
- ▶ Processor affinity and cache

References I

Silberschatz, Abraham, Peter B. Galvin, and Greg Gagne. *Operating system concepts*. 10th edition. John Wiley & Sons, 2018.