CPU Scheduling

Hui Chen [a]

[a]CUNY Brooklyn College

March 5, 2020

Outline

Outline

Recall Process Queues and State Transitions

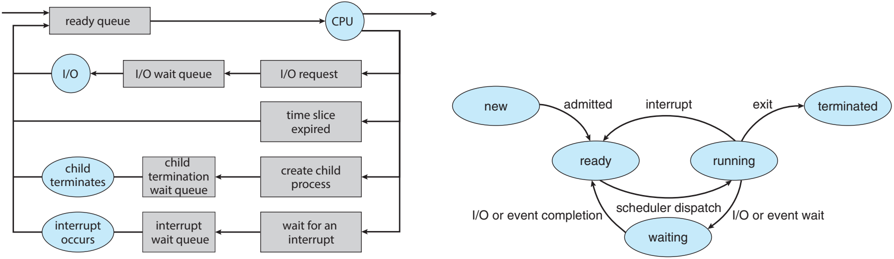

Figure: Process queues and transitions[1].

- 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

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

When does scheduling happens?

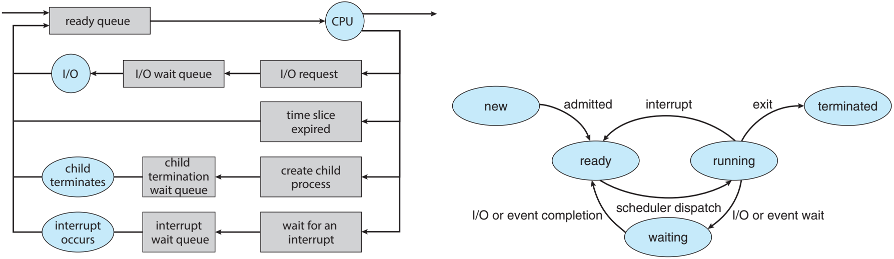

Figure: Process queues and transitions[2].

- 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?

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

Preemptive vs. Non-preemptive Scheduling

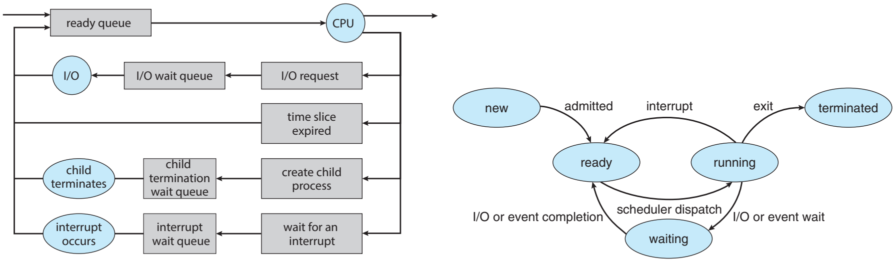

Figure: Process queues and transitions[3].

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

1. `Running` → `Waiting`
2. `Running` → `Ready`
4. `Waiting` → `Ready`
5. $*$ → `Terminated`

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

Preemptive vs. Non-preemptive Scheduling

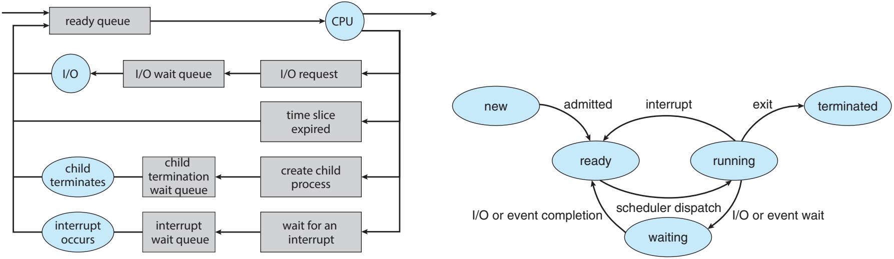

Figure: Process queues and transitions[3].

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

1. `Running` → `Waiting`
2. `Running` → `Ready`
3. `Waiting` → `Ready`
4. `*` → `Terminated`

[3] Silberschatz, 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

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

- man vmstat

  system

  cs: The number of context switches per second.

Monitoring Process Context Switch in Linux

```
~$ ~$  for p in /proc/[0-9]*; do \
> echo "Process ${p#/proc/}:"; \
> while read ln; do \
> echo -e "\t${ln}"; done <<< $(grep -E -o "^.*_ctxt_switches.*$" ${p}/status); \
> done;
Process 1:
  voluntary_ctxt_switches:        41215
  nonvoluntary_ctxt_switches:     15741
Process 10:
  voluntary_ctxt_switches:        26239510
  nonvoluntary_ctxt_switches:     10
...
Process 99:
  voluntary_ctxt_switches:        4
  nonvoluntary_ctxt_switches:     0
~$
```

- 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

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. interacctive systems vs. real-time systems vs. ...

Outline

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	1	8
P_3	2	7
P_4	3	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,

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

  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

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.