CPS 346 Lecture notes: Scheduling

Coverage: [OSCJ8] Chapter 5 (pp. 193-239)

Scheduling management

- processes are managed through the use of multiple queues (or lists) of PCB's; the word queue (in an OS context) has a loose interpretation
- the job queue contains all jobs submitted to the system, but not yet in main memory
- the ready queue contains all jobs in main memory ready to execute
- each I/O device has a queue of jobs waiting for various I/O operations
- a process is dispatched from the ready queue to the CPU; its processing may cause it to be put on a device queue
- all of these events are signaled by interrupts
- job scheduling versus process scheduling (or CPU scheduling)
- here we are primarily discussing process scheduling

CPU and I/O Bursts

- a process cycles between CPU processing and I/O activity
- a process generally has many short CPU bursts or a few long CPU bursts

- I/O bound processes have many short CPU bursts
- CPU bound processes have few long CPU bursts

- this can effect the choice of CPU scheduling algorithm used in an OS

Preemptive scheduling

- CPU scheduling decisions may take place when a process
  1. switches from the running to waiting state
  2. switches from the running to ready state
  3. switches from the waiting to ready state
  4. terminates
- scheduling under conditions 1 and 4 is called non-preemptive (context switch is caused by the running program)
- scheduling under conditions 2 and 3 is preemptive (context switch caused by external reasons)

Scheduling Criteria

Each scheduling algorithm favors particular criteria:

- CPU utilization (maximize)
- throughput: number of processes which complete execution per time unit (maximize)
- turnaround time (TA): total amount of time to execute a particular process (minimize)
- waiting time: amount of time a process has been waiting in the ready queue (minimize)
- response time: amount of time it takes from when a request is submitted to when the response is produced (minimize); does not include the time for a response to be output

Some work is being done to minimize response time variance, to promote predictability.

CPU Scheduling Algorithms

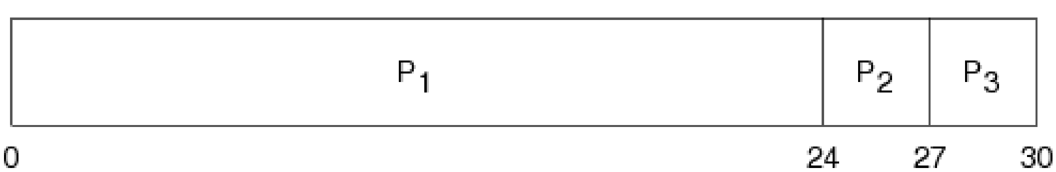
- First-Come, First Serve (FCFS or FIFO) (non-preemptive)
- Priority (e.g., Shortest Job First (SJF; non-preemptive) or Shortest Remaining Time First (SRTF; preemptive))
- Round Robin (preemptive)
- Multi-level Queue
- Multi-level Feedback Queue

First-Come, First Serve

- non-preemptive scheduling management
- ready queue is managed as a FIFO queue
- example: 3 jobs arrive at time 0 in the following order (batch processing):

Process	Burst Time	Arrival	Start	Wait	Finish	TA
1	24	0	0	0	24	24
2	3	0	24	24	27	27
3	3	0	27	27	30	30

Gantt chart:



(regenerated from [OSC9] p. 189)

average waiting time: (0+24+27)/3 = 17

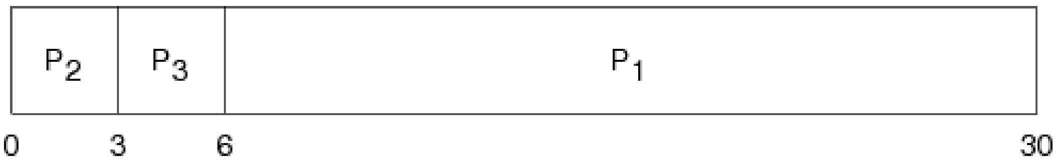
average turnaround time: (24+27+30) = 27

- consider arrival order: 2, 3, 1

Process	Burst Time	Arrival	Start	Wait	Finish	TA

2	3	0	0	0	3	3
3	3	0	3	3	6	6
1	24	0	6	6	30	30

Gantt chart:



(regenerated from [OSC] p. 189)

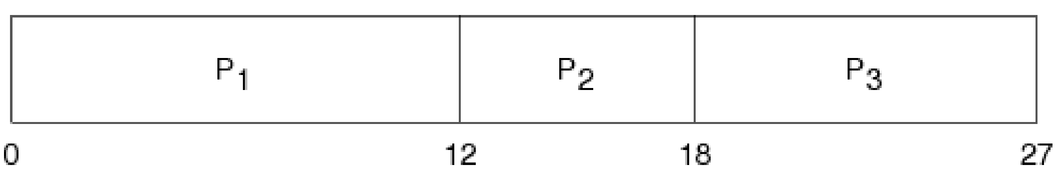
average waiting time: (0+3+6)/3 = 3

average turnaround time: (3+6+30) = 13

- another example:

Process	Burst Time	Arrival	Start	Wait	Finish	TA
1	12	0	0	0	12	12
2	6	1	12	11	18	17
3	9	4	18	14	27	23

Gantt chart:



average waiting time: (0+11+14)/3 = 8.33

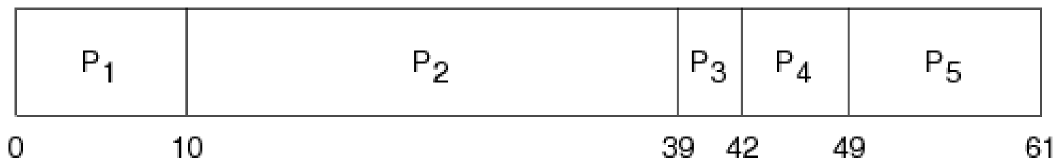
average turnaround time: (12+17+23) = 52/3 = 17.33

- another example:

Process	Burst Time	Arrival	Start	Wait	Finish	TA
1	10	0	0	0	10	10

2	29	0	10	10	39	39
3	3	0	39	39	42	42
4	7	0	42	42	49	49
5	12	0	49	49	61	61

Gantt chart:



(regenerated from [OSC] p. 214)

average waiting time: (0+10+39+42+49)/5 = 28

average turnaround time: (10+39+42+49+61)/5 = 40.2

Priority Scheduling

- associate a priority with each process, allocate the CPU to the process with the highest priority
- any 2 processes with the same priority are handled FCFS
- SJF is a version of priority scheduling where the priority is defined using the predicted CPU burst length
- priorities are usually numeric over a range
- high numbers may indicate low priority (system dependent)
- internal (process-based) priorities: time limits, memory requirements, resources needed, burst ratio
- external (often political) priorities: importance, source (e.g., faculty, student)
- priority scheduling can be non-preemptive or preemptive
- problem: starvation --- low priority processes may never execute because they are waiting indefinitely for the CPU
- a solution: aging --- increase the priority of a process as time progresses
- `nice` in UNIX executes a utility with an altered scheduling priority

- `renice` in UNIX alters the priority of running processes

Shortest Job First (SJF)

- associate with each process the length of its next CPU burst
- schedule the process with the shortest time
- two schemes
  - non-preemptive: once scheduled, a process continues until the end of its CPU burst
  - preemptive: preempt if a new process arrives with a CPU burst of less length than the remaining time of the currently executing process; known as the Shortest Remaining Time First (SRTF) algorithm
- SJF is provably optimal; it yields a minimum average waiting time for any set of processes
- however, we cannot always predict the future (i.e., we do not know the next burst length)
- we can only estimate its length
- an estimate can be formed by using the length of its previous CPU bursts:

  T_n = actual length of the nth CPU burst

  ψ_n = predicted value of nth CPU burst

  $0 <= w <= 1$

  $\psi_{n+1} = w * T_n + (1-w) * \psi_n$

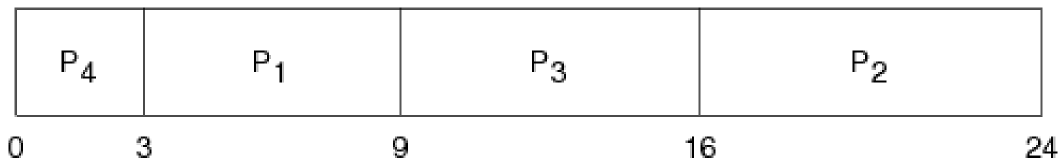
SJF (non-preemptive) examples

- example 1:

Process	Burst Time	Arrival	Start	Wait	Finish	TA

1	6	0	3	3	9	9
2	8	0	16	16	24	24
3	7	0	9	9	16	16
4	3	0	0	0	3	3

Gantt chart:



(regenerated from [OSC] p. 190)

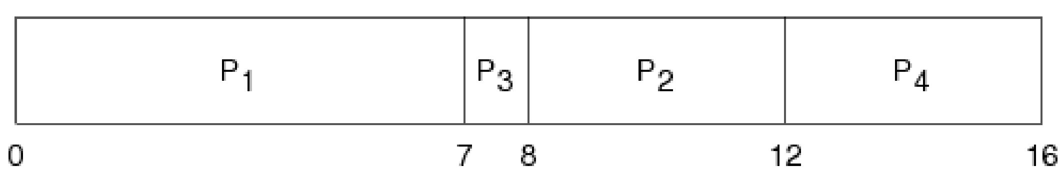
average waiting time: (3+16+9+0)/4 = 7

average turnaround time: (9+24+16+3)/4 = 13

- example 2:

Process	Burst Time	Arrival	Start	Wait	Finish	TA
1	7	0	0	0	7	7
2	4	2	8	6	12	10
3	1	4	7	3	8	4
4	4	5	12	7	16	11

Gantt chart:



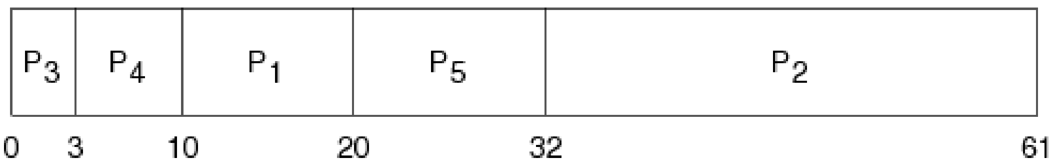
average waiting time: (0+6+3+7)/4 = 4

average turnaround time: (7+4+10+11)/4 = 8

- example 3:

Process	Burst Time	Arrival	Start	Wait	Finish	TA
1	10	0	10	10	20	20
2	29	0	32	32	61	61
3	3	0	0	0	3	3
4	7	0	3	3	10	10
5	12	0	20	20	32	32

Gantt chart:



(regenerated from [OSC] p. 214)

average waiting time: (10+32+0+3+20)/5 = 13

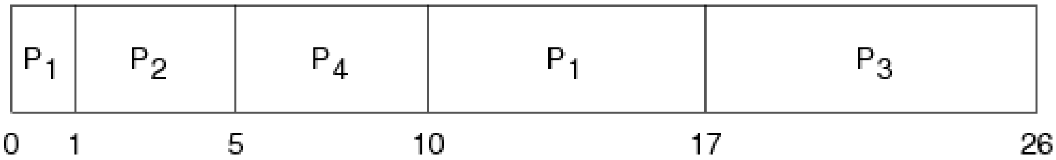
average turnaround time: (10+39+42+49+61)/5 = 25.2

SRTF (preemptive) examples

- example 1:

Process	Burst Time	Arrival	Start	Wait	Finish	TA
1	8	0	0	9	17	17
2	4	1	1	0	5	4
3	9	2	17	15	26	24
4	5	3	5	2	10	7

Gantt chart:



(regenerated from [OSC] p. 192)

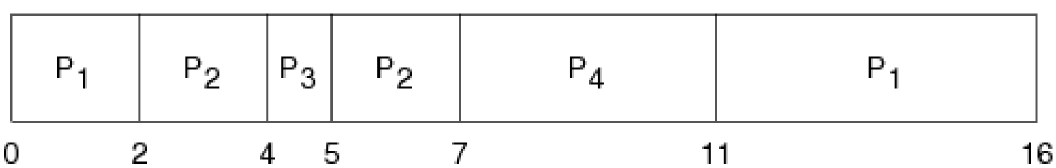
average waiting time: (9+0+15+2)/4 = 6.5

average turnaround time: (17+4+24+7)/4 = 13

- example 2:

Process	Burst Time	Arrival	Start	Wait	Finish	TA
1	7	0	0	9	16	16
2	4	2	2	1	7	5
3	1	4	4	0	5	1
4	4	5	7	2	11	6

Gantt chart:



average waiting time: (9+1+0+2)/4 = 3

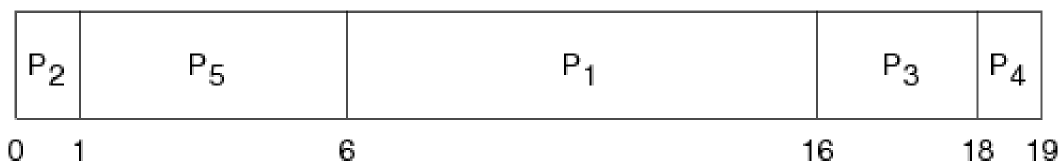
average turnaround time: (16+5+1+6)/4 = 7

Priority Scheduling example

Process	Burst Time	Priority	Arrival	Start	Wait	Finish	TA
1	10	3	0	6	6	16	16
2	1	1	0	0	0	1	1
3	2	4	0	16	16	18	18

4	1	5	0	18	18	19	19
5	5	2	0	1	1	6	6

Gantt chart:



(regenerated from [OSC] p. 193)

average waiting time: (6+0+16+18+1)/5 = 8.2

average turnaround time: (1+6+16+18+19)/5 = 12

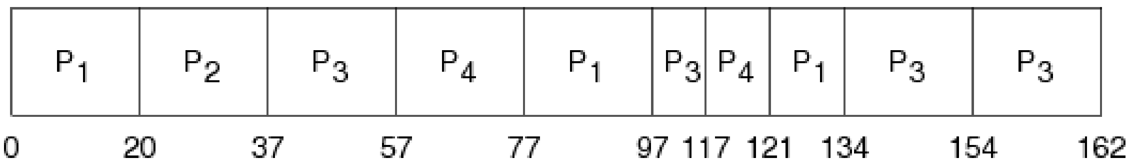
Round Robin

- time sharing (preemptive) scheduler where each process is given access to the CPU for 1 time quantum (slice) (e.g., 20 milliseconds)
- a process may block itself before its time slice expires
- if it uses its entire time slice, it is then preempted and put at the end of the ready queue
- the ready queue is managed as a FIFO queue and treated as a circular
- if there are n processes on the ready queue and the time quantum is q, then each process gets 1/n time on the CPU in chunks of at most q time units
- no process waits for more than (n-1)q time units
- the choice of how big to make the time slice (q) is extremely important
  - if q is very large, Round Robin degenerates into FCFS
  - if q is very small, the context switch overhead defeats the benefits

- example 1 (q = 20):

Process	Burst Time	Arrival	Start	Wait	Finish	TA
1	53	0	0	?	134	134
2	17	0	20	?	37	37
3	68	0	37	?	162	162
4	24	0	57	?	121	121

Gantt chart:



waiting times:

p_1: (77-20) + (121-97) = 81

p_2: (20-0) = 20

p_3: (37-0) + (97-57) + (134-117) = 94

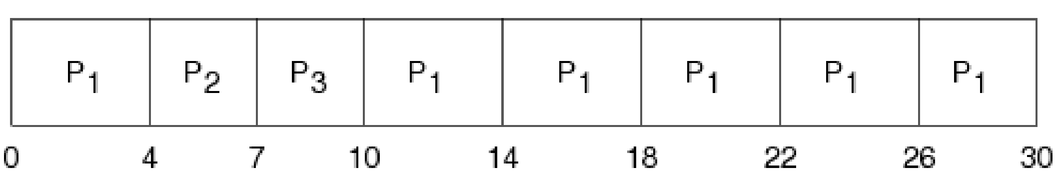
p_4: (57-0) + (117-77) = 97

average waiting time: (81+20+94+97)/4 = 73

- example 2 (q = 4):

Process	Burst Time	Arrival	Start	Wait	Finish	TA
1	24	0	0	6	30	30
2	3	0	4	4	7	7
3	3	0	7	7	10	10

Gantt chart:



(regenerated from [OSC] p. 194)

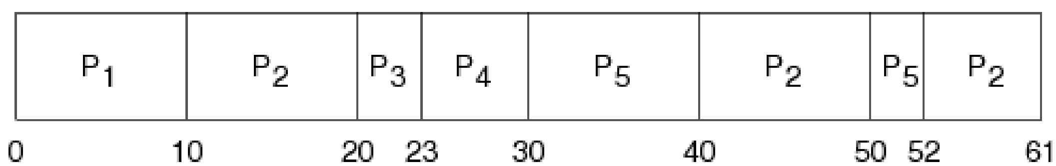
average waiting time: (6+4+7)/3 = 5.67

average turnaround time: (30+7+10)/3 = 15.67

- example 3 (q = 10):

Process	Burst Time	Arrival	Start	Wait	Finish	TA
1	10	0	0	0	10	10
2	29	0	10	32	61	61
3	3	0	20	20	23	23
4	7	0	23	23	30	30
5	12	0	30	40	52	52

Gantt chart:



(regenerated from [OSC] p. 214)

average waiting time: (0+32+20+23+40)/5 = 23

average turnaround time: (10+39+42+49+61)/5 = 35.2

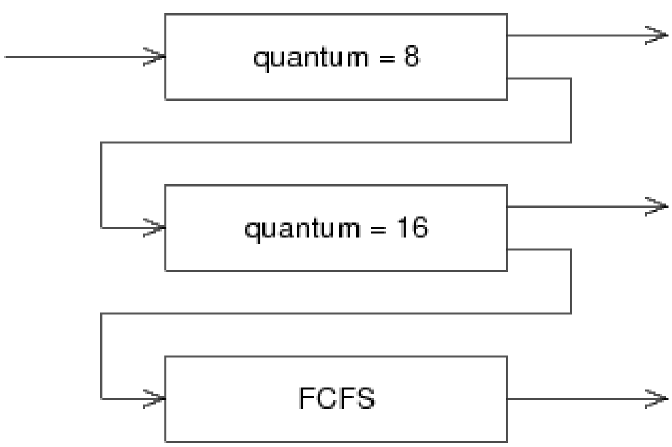
Multilevel Queue

- the ready queue is managed as multiple queues based on various characteristics. For instance,
  - foreground (interactive)
  - background (batch)
- each queue uses a particular scheduling algorithm. For instance,
  - foreground (round robin)

- background (FCFS)
- scheduling must be done between queues:
  - fixed priority (may lead to starvation) (e.g., foreground jobs have absolute priority over background jobs)
  - time slice per queue

Multilevel Feedback Queue

- processes move between the various queues
- a multilevel feedback queue is characterized by
  - number of queues
  - scheduling algorithm for each queue
  - method used to determine when to upgrade a process
  - method used to determine when to demote a process
  - method used to determine on which queue a process begins (each time it returns to the ready state)
- example:
  - 3 queues
  - fixed priority based on length of CPU burst
  - RR for 1st queue, FCFS for last queue
  - each process begins on top queue (quantum = 8)



(regenerated from [OSC] Fig. 5.7 on p. 198)

Algorithm Evaluation

- which algorithm should be used in a particular system?
- how should the parameters (e.g., q, number of levels) be defined?
- on which criteria do we base our decisions?

Four approaches to evaluation

- deterministic modeling
- queue models
- simulation
- implementation

Deterministic modeling

- define a workload and compare it across algorithms
- simple to execute and results in distinct values to compare
- however, the results apply only to that case and cannot be generalized
- a set of workload scenarios with varying characteristics can be defined and analyzed
- must be careful about any conclusion drawn

Queuing models

- n = average queue length
- W = average waiting time in the queue
- λ = average arrival rate
- Little's Formula: $n = \lambda * W$
- Little's formula can be applied to the CPU and ready queue, or the wait queue for any device
- values can be obtained by measuring a real system over time and mathematically estimating
- the estimates are not always accurate due to:
    - complicated algorithms

  - assumptions
- therefore, the queuing model may not reflect reality to the level needed

---

References

[OSC9] A. Silberschatz, P.B. Galvin, and G. Gagne. Operating Systems Concepts. John Wiley and Sons, Inc., Ninth edition, 2009.

[OSCJ8] A. Silberschatz, P.B. Galvin, and G. Gagne. Operating Systems Concepts with Java. John Wiley and Sons, Inc., Eighth edition, 2010.