Comp 204: Computer Systems and Their Implementation

Lecture 11: Scheduling cont'd

Today

- Scheduling algorithms continued
  - Shortest remaining time first (SRTF)
  - Priority scheduling
  - Round robin (RR)
  - Multilevel queues

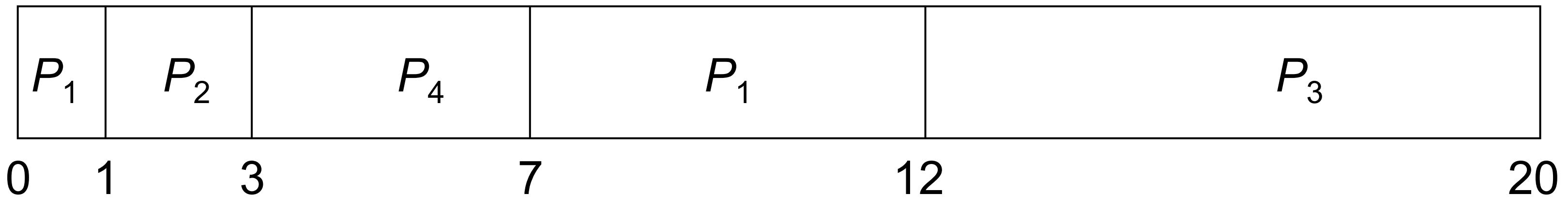
Shortest Remaining Time First

- Preemptive version of the SJF algorithm
  - CPU is allocated to the job that is closest to being completed
  - Can be preempted if there is a newer job in the ready queue that has a shorter completion time

- Suppose we have four processes arriving one CPU time cycle apart and in the following order:
  - P_1 with CPU burst of 6 milliseconds (arrives at time 0)
  - P_2 with CPU burst of 2 milliseconds (arrives at time 1)
  - P_3 with CPU burst of 8 milliseconds (arrives at time 2)
  - P_4 with CPU burst of 4 milliseconds (arrives at time 3)

- Using the SRTF algorithm we can schedule the processes as viewed in the following Gantt chart…..

Example



- P_1 starts at time 0 then P_2 arrives at time 1
- As P_2 requires less time (2 milliseconds) to complete than P_1 (5 milliseconds), then P_1 is preempted and P_2 is scheduled

- The next processes are then treated in this same manner

- Thus, the average wait time is:

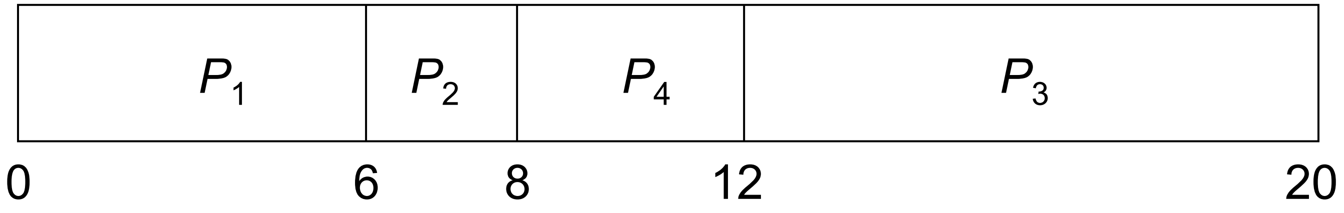
  ((7 - 1) + (1 - 1) + (12 - 2) + (3 - 3))/4 = 4 milliseconds

Note: the above calculation accounts for the arrival time of each processes in that it is subtracted

Exercise

- Using the same processes, arrival times and CPU burst times as in the previous example, what would the average wait time be if we were using the Shortest Job First algorithm?

Answer



- Thus, the average wait time is:

  (0 + (6 - 1) + (12 - 2) + (8 - 3))/4 = 20/4 = 5 milliseconds

Shortest Remaining Time First

- This preemptive version of the SJF algorithm is faster than the non-preemptive version
- However, within our calculation for SRTF, we have not included any time for context switching
  - Context switching is required for all preemptive algorithms
  - Time taken to perform context switches will depend upon the particular system, but it must be accounted for

Shortest Remaining Time First

- Advantages:
  - Allows for preemption, which reduces wait time over non-premptive version
  - Short jobs completed quickly
- Disadvantages:
  - Time gain diminished by the need for context switching
    - Can be kept to a minimum if system implements efficient context switching

Priority Scheduling

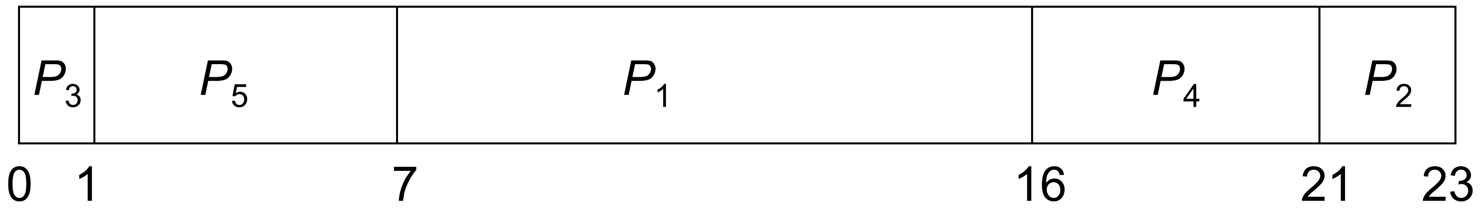
- Algorithm that gives preferential treatment to important jobs
  - Each process is associated with a priority and the one with the highest priority is granted the CPU
  - Equal priority processes are scheduled in FCFS order
    - SJF is a special case of the general priority scheduling algorithm

- Priorities can be assigned to processes by a system administrator (e.g. staff processes have higher priority than student ones) or determined by the Processor Manager on characteristics such as:
  - Memory requirements
  - Peripheral devices required
  - Total CPU time
  - Amount of time already spent processing

Example

- Suppose we have five processes all arriving at time 0 in the following order and having the following CPU burst times:
  - P_1 with CPU burst of 9 milliseconds, priority 3
  - P_2 with CPU burst of 2 milliseconds, priority 4
  - P_3 with CPU burst of 1 millisecond, priority 1
  - P_4 with CPU burst of 5 milliseconds, priority 3
  - P_5 with CPU burst of 6 milliseconds, priority 2

- Assuming that 0 represents the highest priority, using the priority algorithm we can view the result as the following Gantt chart:



Priority Scheduling

- For the previous example, the average waiting time is: (7 + 21 + 0 + 16 + 1)/5 = 9 milliseconds

- Advantages:
  - Simple algorithm
  - Important jobs are dealt with quickly
  - Can have a preemptive version

- Disadvantages:
  - Process starvation can be a problem
    - Can be alleviated through the aging technique: gradually increasing the priority of processes that have been waiting a long time in the system

Round Robin

- Preemptive algorithm that gives a set CPU time to all active processes
  - Similar to FCFS, but allows for preemption by switching between processes
  - Ready queue is treated as a circular queue where CPU goes round the queue, allocating each process a pre-determined amount of time
- Time is defined by a time quantum: a small unit of time, varying anywhere between 10 and 100 milliseconds
- Ready queue treated as a First-In-First-Out (FIFO) queue
  - new processes joining the queue are added to the back of it
- CPU scheduler selects the process at the front of the queue, sets the timer to the time quantum and grants the CPU to this process

Round Robin

- Two potential outcomes ensue:

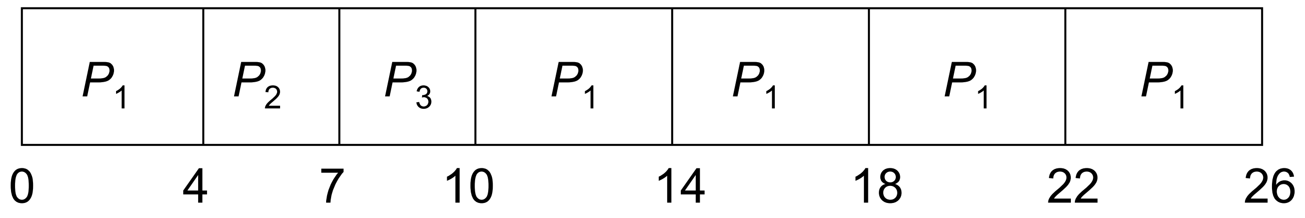
  1) If the process’ CPU burst time is less than the specified time quantum it will released the CPU upon completion
  - Scheduler will then proceed to the next process at the front of the ready queue

  2) If the process’ CPU burst time is more than the specified time quantum, the timer will expire and cause an interrupt (i.e. the process is preempted) and execute a context switch
  - The interrupted process is added to the end of the ready queue
  - Scheduler will then proceed to the next process at the front of the ready queue

Example

- Suppose we have three processes all arriving at time 0 and having CPU burst times as follows:
  - P_1 with CPU burst of 20 milliseconds
  - P_2 with CPU burst of 3 milliseconds
  - P_3 with CPU burst of 3 milliseconds

- Supposing that we use a time quantum of 4 milliseconds, using the round robin algorithm we can view the result as the following Gantt chart:



Round Robin

- In the previous example P_1 executed for the first four milliseconds and is then interrupted after the first time quantum has lapsed, but it requires another 16 milliseconds to complete
- P_2 is then granted the CPU, but as it only needs 3 milliseconds to complete, it quits before its time quantum expires
- The scheduler then moves to the next process in the queue, P_3 , which is then granted the CPU, but that also quits before its time quantum expires

Round Robin

- Now each process has received one time quantum, so the CPU is returned to process P_1 for an additional time quantum

- As there are no other processes in the queue, P_1 is given further additional time quantum until it completes
  - No process is allocated the CPU for more than one time quantum in a row, unless it is the only runnable process

- The average wait time is ((10 - 4) + 4 + 7)/3 = 5.66 milliseconds

Round Robin

- The performance of the round robin algorithm depends heavily on the size of the time quantum
  - If time quantum is too large, RR reduces to the FCFS algorithm
  - If time quantum is too small, overhead increases due to amount of context switching needed

- Advantages:
  - Easy to implement as it is not based on characteristics of processes
    - Commonly used in interactive/time-sharing systems due to its preemptive abilities
  - Allocates CPU fairly

- Disadvantages:
  - Performance depends on selection of a good time quantum
    - Context switching overheads increase if a good time quantum is not used

Multilevel Queues

- A class of scheduling algorithms that categorise processes by some characteristic and can be used in conjunction with other policies
  - Processes can be categorised by: memory size, process type, their response time, their externally assigned priorities, etc.
- A multilevel queue-scheduling algorithm divides the ready queue into several separate queues to which processes are assigned
  - Each queue is associated with one particular scheduling algorithm
- Also requires scheduling *between* queues
  - Commonly implemented as fixed-priority preemptive scheduling