COMP124

Computer Systems

11 | Priority Scheduling | Round Robin

Dr Stuart Thomason

Process Scheduling

- Recall the two types of scheduling algorithm
  - Non-preemptive – Process stays on the CPU until it terminates or blocks for I/O
  - Preemptive – Process can be interrupted and replaced with another process

- Remember…
  - The burst time is the time a process takes on the CPU before it blocks or terminates
  - The arrival time is the time a process is first put into the ready state

- Processes are being created and terminated all the time
  - These lecture examples are simplified
  - Arrival time is not always zero for all processes
  - A newly arrived process might slot into the schedule before an existing process
  - Calculations (eg. average wait time) are constantly changing as processes arrive

Priority Scheduling

- A preemptive algorithm that gives preferential treatment to important processes
  - Each process has a priority assigned to it
  - Highest priority in ready queue gets onto the CPU first
  - Processes with equal priority just use FCFS
  - If a new process arrives with higher priority than current process, interrupt it

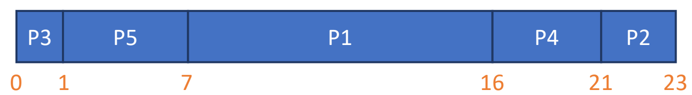
- Priorities can be set by the user or assigned by the OS depending on whatever characteristic is most important
  - Memory usage
  - I/O throughput
  - Total CPU time
  - Time already elapsed

Priority Scheduling – Example

- We have five processes with the following burst times and priorities, and they arrive in this order…

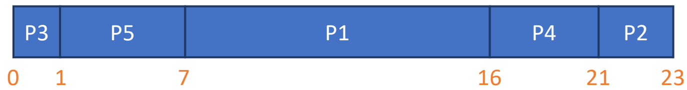
  - P1 with burst of 9 ms, priority 3
  - P2 with burst of 2 ms, priority 2
  - P3 with burst of 1 ms, priority 5
  - P4 with burst of 5 ms, priority 3
  - P5 with burst of 6 ms, priority 4

- Higher numbers mean higher priority (zero is least important)



Priority Scheduling – Average Wait Time

- We can work out the average wait time as we did before…
  - P1 waited for 7 ms
  - P2 waited for 21 ms
  - P3 waited for 0 ms
  - P4 waited for 16 ms
  - P5 waited for 1 ms
- Average wait time = (7 + 21 + 0 + 16 + 1) / 5 = 9 ms



Priority Scheduling – Advantages & Disadvantages

- Advantages…
  - Smaller wait times and latency for higher priority processes
  - Important processes are dealt with quickly
- Disadvantages…
  - Bigger wait times and latency for lower priority processes
  - Process starvation is still possible
- Starvation can be solved by process aging
  - Increase priority for processes that have been waiting a long time
  - Eventually their priority will bring them to the front of the queue
- Commonly used in real-time operating systems where processes must meet deadlines

Round Robin Scheduling

- A preemptive algorithm that gives a set amount of CPU time to each process
- Time slice (quantum) is typically between 10 ms and 100 ms depending on OS and need

- Similar to FCFS but processes can be interrupted before they terminate or block

- Ready state is treated as a circular FIFO queue

  - New processes arrive at back of queue
  - Scheduler takes next process from front of queue
  - Process runs for a fixed amount of time
  - Context switch moves current process to back of queue and despatches next in line
  - Scheduler keeps going round and round all processes in the ready queue

Round Robin Scheduling

- When a process is placed on the CPU…
  - It will either terminate before the quantum expires
  - Or it will be interrupted (preempted) before it terminates

- If the burst time is less than the quantum time…
  - The process will terminate or block for I/O
  - Scheduler will immediately take next process from queue and place on the CPU
  - See T on next slide

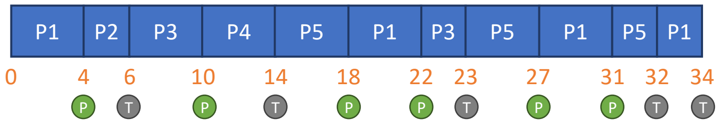
- If the burst time is more than the quantum time…
  - Process will be interrupted and a context switch will happen
  - Interrupted process will be moved to the back of the ready queue
  - See P on next slide

Round Robin Scheduling – Example

- We have five processes with the following burst times, and they arrive in this order…

  - P1 with a burst of 14 ms
  - P2 with a burst of 2 ms
  - P3 with a burst of 5 ms
  - P4 with a burst of 4 ms
  - P5 with a burst of 9 ms

- The scheduler is using a fixed time quantum of 4 ms

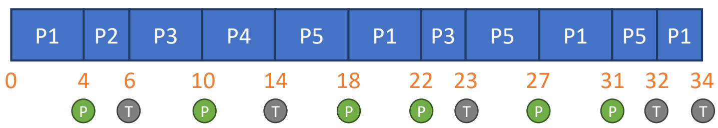


Round Robin – Preemption & Termination

- In the example…

  - P1 had 4 ms on the CPU and was then preempted
  - P2 only needed 2 ms and terminated before the quantum
  - P3 had the next 4 ms and was preempted
  - P4 needed 4 ms so used all the quantum before terminating
  - P5 had the next 4 ms and was preempted

- Then the round robin started again at the front of the queue…



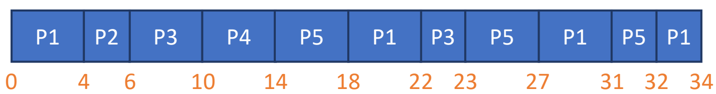
Round Robin – Turn Around & Wait Times

- For each process…
  - Its turnaround time is the time when it terminated minus the time when it arrived in the process queue
  - Its wait time is its turnaround time minus its burst time
- To simplify the example, assume that all processes arrived at time 0 (in reality processes are arriving all the time)
- Consider P2…
  - It arrived at time 0 and terminated at time 6
  - So its turnaround time was 6 ms
  - Its burst time was 2 ms
  - So its wait time was 4 ms

Round Robin – Average Time Calculations

- Average turnaround time is…
  (34 + 6 + 23 + 14 + 32) / 5 = 21.8 ms
- Average wait time is…
  (20 + 4 + 18 + 10 + 23) / 5 = 15 ms

Process	Turnaround Time	Wait Time
P1	34 – 0 = 34	34 – 14 = 20
P2	6 – 0 = 6	6 – 2 = 4
P3	23 – 0 = 23	23 – 5 = 18
P4	14 – 0 = 14	14 – 4 = 10
P5	32 – 0 = 32	32 – 9 = 23



Round Robin – Advantages & Disadvantages

- Advantages:
  - Easy to implement as a queue
  - Doesn't depend on guessing or estimating burst times
  - Response time based on number of processes instead of process burst times
  - No starvation (every process gets a fair turn on the CPU)
  - Balanced throughput (see below)

- Disadvantage:
  - Depends on selecting a good time quantum
  - Extensive overhead if the quantum is too short

- If quantum is too big, behaves like FCFS (but shorter processes executed faster)
- If quantum is too small, behaves like SJF (but longer processes executed faster)

Windows Priority Scheduling Algorithm

- Windows uses the Priority Round Robin algorithm
- Priority values range from 0 to 31
- Every new process is given one of five base priorities
  - IDLE (4)
  - BELOW_NORMAL (6)
  - NORMAL (8) – the default priority
  - HIGH (13)
  - REALTIME (24)
- Processes given a priority boost when they are in foreground (ie. they are the currently active window)
- Process can be boosted when it receives input or an event for which it has been waiting
- Boosted processes are reduced in priority by one level at the end of each time slice (quantum) until base level is reached

Linux Priority Scheduling Algorithm

- Linux also assigns a priority to each process
  - Kernel processes have a value from 1 to 99
  - User processes have a value from 100 to 139
- It also assigns a ‘nice’ value to each user
  - Ranges from −20 (highest priority) to +19 (lowest priority)
  - Default is 0 for every user
  - Users can change priorities with nice and renice commands

- Linux uses an algorithm called Completely Fair Scheduling (similar to Round Robin)
  - Ready processes are stored in a balanced tree (not a queue)
  - Gives credit for time spent in the blocked state
  - Time quantum varies depending on processor demand
  - Process priority and nice values combine to give overall priority