University of Newcastle
School of Electrical Engineering and Computing

COMP2240 - Operating Systems
Workshop 3
Topics: Scheduling

1. Consider a variant of the RR scheduling algorithm where the entries in the ready queue are pointers to the PCBs.

a) What would be the effect of putting two pointers to the same process in the ready queue?
b) What would be the major advantage/disadvantage of this scheme?
c) How could you modify the basic RR algorithm to achieve the same effect without the duplicate pointers?

2. An interactive system using round-robin scheduling and swapping tries to give guaranteed response to trivial requests using the following algorithm: After completing a round-robin cycle among all Ready processes, the system determines the time slice to allocate to each Ready process for the next cycle by dividing the maximum response time by the number of processes requiring service. Will this work in practice?

3. The following processes are being scheduled using a preemptive, round robin scheduling algorithm. Each process is assigned a numerical priority, with a higher number indicating a higher relative priority.
In addition to the processes listed below, the system also has an ***idle task*** (which consumes no CPU resources and is identified as $\mathbf{P_{idle}}$). This task has priority 0 and is scheduled whenever the system has no other available processes to run. The length of a time quantum is 10 units.

Thread	**Priority**	**Burst**	**Arrival**
P1	40	20	0
P2	30	25	25
P3	30	25	30
P4	35	15	60
P5	5	10	100
P6	10	10	105

If a process is preempted by a higher-priority process, the preempted process is placed at the end of the queue.

a) Show the scheduling order of the processes using a Gantt chart.
b) What is the turnaround time for each process?
c) What is the waiting time for each process?
d) What is the CPU utilization rate?

4. A variation of preemptive priority scheduling has dynamically changing priorities. A new process is assigned a priority of 0. While a process is in the ready queue, its priority changes at the rate of α; and while it is executing, its priority changes at the rate of β. The different values of α and β give different algorithms. If larger values imply higher priorities, state with reasons the type of algorithm that will result if:

a) $\beta > \alpha > 0$
b) $\alpha < \beta < 0$

5. Consider a variant of Round Robin Scheduling, say NRR scheduling. In NRR scheduling, each process can have its own time quantum, *q*. The value of *q* starts out at 40 ms and decreased by 10 ms each time it goes through the round robin queue, until it reaches a minimum of 10 ms. Thus, long jobs get decreasingly shorter time slices.

Analyse this scheduling algorithm for three jobs A, B and C that arrive in the system having estimated burst times of 100 ms, 120 ms, and 60 ms respectively. Also identify some advantages and disadvantages that are associated with this algorithm.

6. Suppose that a scheduling algorithm (at the level of short-term CPU scheduling) favors those processes that have used the least processor time in the recent past. Why will this algorithm favor I/O-bound programs and yet not permanently starve CPU-bound programs?

7. Android operating system uses the CFS (Completely Fair Scheduler) algorithm implemented at its Linux Kernel. The default scheduling algorithm in Linux 2.6.23 is the completely fair scheduler (CFS). As the documentation to the kernel states, 80% of CFS design can be summed up in a single sentence: CFS basically models an “ideal, precise multitasking CPU” on real hardware.

The general principle behind this scheduler is to provide maximum fairness to each task in the system in terms of the computation power it is given. More precisely, this means that when the CFS makes a scheduling decision, it always selects processes that have been waiting for the longest amount of time. In android system, with *n* running processes, each process would be having *1/n* amount of CPU-time while running constantly – in any measurable period all the tasks would have run for the same amount of wall-clock time.

A nice value in the range from -20 to +19 is assigned to each task of android process. Lower nice value indicates a higher relative priority and processes with lower nice values receive a higher proportion of CPU processing time than tasks with higher nice values. The default nice value is 0. A nice value is mapped to a weight value (W_i).

CFS identifies a targeted latency (TL), which is an interval of time during which every runnable task should run at least once.

$$\text{Time slice for a task } i = TL * W_i / (\text{Sum of all } W_i)$$

Calculate the time slice of two android processes t1 and t2 with nice values of 0 (weight value is 1024) and 5 (weight value is 335) respectively. Where the targeted latency of the system is 20 ms.

Supplementary problems:

S1 Which type of process is generally favoured by a multi-level feedback queuing scheduler: a processor-bound process or an I/O-bound process? Briefly explain why?

S2 Contrast the scheduling policies you might use when trying to optimize a time-sharing system with those you would use to optimize a *multiprogrammed* batch system.

S3. Explain how time quantum value and context switching time affect each other, in a round-robin scheduling algorithm.