**Spring 2019 COMP 3511 Homework Assignment #2 Solution**

**Handout Date: March 8, 2019 Due Date: March 22, 2019**

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_ ID: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_E-Mail: \_\_\_\_\_\_\_\_\_\_\_\_\_

**Please read the following instructions carefully before answering the questions:**

* You should finish the homework assignment **individually**.
* There are total of **4** questions.
* When you write your answers, please try to be precise and concise.
* Fill in your name, student ID, email at the top of each page.
* Please fill in your answers in the space provided, or you can type your answers in the MS Word file.
* **Homework Submission**: the homework is submitted to **assignment #2** on **CASS**

1. [20 points] Multiple choices
   1. Which of the following components of program state are shared across threads in a multithreaded process?

①Register values ②Heap memory ③Global variables ④Stack memory

A) ①②

B) ②③

C) ①④

D) ③④

**Answer: B**

* 1. Which one of the following is not shared by threads?

A) program counter

B) stack

C) both program counter and stack

D) none of the mentioned

**Answer: C**

* 1. Which module gives control of the CPU to the process selected by the short-term scheduler?

A) dispatcher

B) interrupt

C) scheduler

D) none of the mentioned

**Answer: A**

* 1. Cancelling a thread asynchronously \_\_\_\_\_\_\_\_\_\_\_.

A) spoils the process execution

B) may not free each resource

C) frees all the resources properly

D) allows the target thread to periodically check if it should be cancelled

**Answer: B**

* 1. According to Amdahl's Law, what is the speedup gain for an application that is 40% parallel and we run it on a machine with 4 processing cores?

A) 0.7

B) 1.82

C) 1.43

D) 0.55

**Answer: C**

* 1. Which of the following scheduling algorithms could result in starvation?

①First-come, first served ②Shortest job first (non-preemptive)

③Round Robin ④Priority

A) ①②

B) ②④

C) ①③

D) ③④

**Answer: B**

* 1. CPU scheduling is the basis of \_\_\_\_\_\_\_\_\_\_\_

A) multiprocessor systems

B) multiprogramming operating systems

C) larger memory sized systems

D) none of the mentioned.

**Answer: B**

* 1. One of the disadvantages of the priority scheduling algorithm is that:

A) it schedules in a very complex manner

B) its scheduling takes up a lot of time

C) it can lead to some low priority process waiting indefinitely for the CPU

D) none of the mentioned

**Answer: C**

* 1. LWP is \_\_\_\_\_\_\_\_\_\_\_\_.

A) short for lightweight processor

B) placed between user and kernel threads

C) placed between system and kernel threads

D) common in systems implementing one-to-one multithreading models

**Answer: B**

* 1. With round robin scheduling algorithm in a time shared system,

A) using very large time slices degenerates it to First Come First Served scheduling algorithm

B) using very small time slices degenerates it to First Come First Served scheduling algorithm

C) using extremely small time slices increases performance

D) using very small time slices degenerates it to Shortest Job First algorithm

**Answer: A**

1. [20 points] Multithread Process
   1. What set of resources are needed or used when a thread within a process is created? How do they differ from those used when a process is created? (4 points)

**Answer**: Thread creation typically uses fewer resources than process creation, which usually involves in allocating a small data structure to hold a register set, stack, and priority (3 points). Creating a process requires allocating a process control block (PCB), a rather large data structure. The PCB includes a memory map, a list of open files, and environment variables (3 points). Allocating and managing the memory map is typically the most time-consuming activity.

* 1. Linux does not distinguish between processes and threads. Instead, it treats them in the same way by allowing a task to be more akin to a process or a thread depending on the set of flags passed to the clone() system call. Please compare and contrast clone() and fork() in Linux. (4 points)

**Answer**: Linux uses both fork() and clone() to create a task. When clone() is used, it can use a set of parameters or flags that specifies how much sharing between the parent and child tasks, further allow the child thread to execute a function specified. fork() system call requires no parameters, which enforce the child thread to execute identical codes or programs.

* 1. Consider a multicore system and a multithreaded program using *many-to-many threading model*. Let the number of user-level threads in the program be much greater than the number of processing cores in the system. How many kernel threads should the OS allocate in order to increase the utilization of the multiprocessor system, and why ? (4 points) Hint: Consider the number of kernel threads allocated to the program is less than, equal to or is greater than the number of processing cores.

**Answer**: When the number of kernel threads is less than the number of processors, then some of the processors will remain idle, since the scheduler maps only kernel threads to processors and not user-level threads to processors. When the number of kernel threads is exactly equal to the number of processors, then it is possible that all of the processors will be utilized simultaneously. However, when a kernel thread blocks inside the kernel (due to a page fault or while invoking system calls), the corresponding processor will remain idle. When there are more kernel threads than processors, a blocked kernel thread can be swapped out in favor of another kernel thread that is ready to execute, thereby increasing the utilization of the multiprocessor system.

* 1. Using Amdahl’s Law, list the speedup gain of an application in a table or figure that has a 60 percent parallel component for systems with one, two, three, … and eight processing cores. (8 points)

**Answer**: The speedup is calculated as

|  |  |  |
| --- | --- | --- |
| N | S | Speedup |
| 1 | 0.4 | 1 |
| 2 | 0.4 | 1.429 |
| 3 | 0.4 | 1.667 |
| 4 | 0.4 | 1.818 |
| 5 | 0.4 | 1.923 |
| 6 | 0.4 | 2 |
| 7 | 0.4 | 2.059 |
| 8 | 0.4 | 2.105 |

1. [20 points] CPU scheduling
   1. Distinguish between PCS and SCS scheduling. Under what threading model, PCS and SCS are the same? (5 pints)

**Answer**: PCS scheduling is local to the process. It is how the thread library schedules threads onto available LWPs. SCS scheduling is used when the operating system schedules kernel threads. On systems using either the many-to-one or the many-to-many model, the two scheduling models are fundamentally different. (4 points) On systems using the one-to-one model, PCS and SCS are the same. (1 point)

* 1. Can FCFS be considered as one type of priority scheduling? Please be careful in justifying your answer. (4 points)

**Answer**: Yes (1 point). FCFS gives the highest priority to the process that has been in existence the longest (3 points)

* 1. What is the difference between response time and turnaround time? How do they conflict with each other? (5 points)

**Answer:** Turnaround time measures the time that .. first arrival to the ready queue and the completion time of all its CPU bursts. Response time measures the time that elapses between a request and the first response produced, which is the time between arrival to the ready queue and the completion time of its current CPU burst. The turnaround time is minimized by executing the shortest tasks first. Such a scheduling policy could, however, starve other long-running tasks and thereby adversely affect the response time.

* 1. CPU scheduling often needs to balance CPU-bound and I/O-bound programs. Suppose we design a simple CPU scheduling algorithm that favors those processes that have used the least CPU time in the recent past. Why will this algorithm favor I/O-bound programs and yet not permanently starve CPU-bound programs? Can this also improve I/O device utilization? (6 points)

**Answer:** In the non-preemptive scheduling, a process will not be forced to give up CPU, while preemptive scheduling will. It will favor the I/O-bound programs because of the relatively short CPU bursts requested by them; however, the CPU-bound programs will not starve, because the I/O-bound programs will relinquish the CPU relatively often to do their I/O. Since I/O-bound programs get CPU service fasters, so they can perform I/O operations more often.

1. [40 points] Scheduling algorithms
   1. Consider two processes, process A is CPU-bound with CPU burst time one hour, and process B is I/O-bound repeating CPU burst time 1 ms and disk I/O 9 ms. Please illustrate using Gnatt charts the round-robin scheduling with a time quantum respectively setting to 100 ms and 1 ms, and SRTF scheduling. Please discuss which one performs the best (10 points)? Hint: consider disk utilization and number of context switch.

**Answer**:

B 1ms

A 100ms

B’s I/O

RR 100ms time slice

BAAAAAAAAAB

B’s I/O: 9ms

RR 1ms time slice

AB 1ms

B 1ms

B’s I/O

SRTF

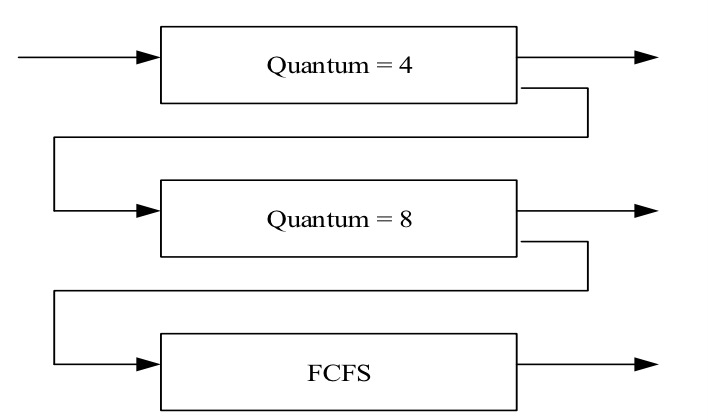
Disk utilization in each scheduling are roughly 8.9%, 90% and 90% respectively, but SRTF results in significantly fewer number of context switch.

* 1. Given the arrival time and CPU-burst times of 6 processes shown in the following diagram:

|  |  |  |
| --- | --- | --- |
| Process | Arrival Time (ms) | Burst Time (ms) |
| P1 | 0 | 8 |
| P2 | 3 | 16 |
| P3 | 7 | 3 |
| P4 | 15 | 12 |
| P5 | 20 | 6 |
| P6 | 5 | 2 |

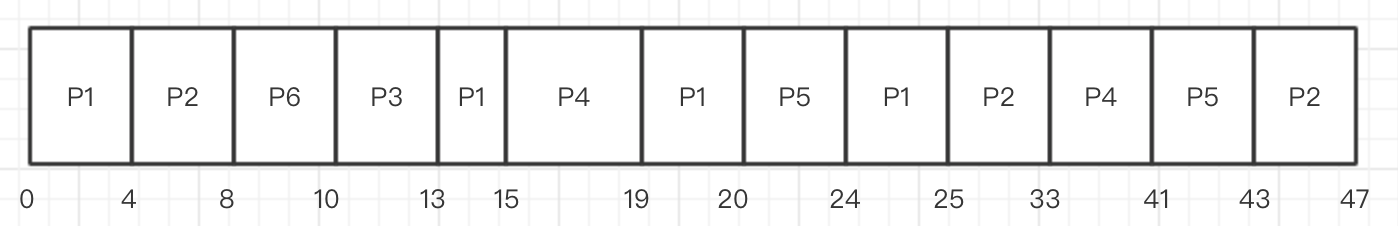
Suppose the OS uses a 3-level feedback queue to schedule the above 6 processes. Round-Robin scheduling strategy is used for the queue with the highest priority and the queue with the second highest priority, but the time quantum used in these two queues is different. First-come-first-serve scheduling strategy is used for the queue with the lowest priority. The scheduling is **preemptive**.

(Note: In this scenario, the scheduling is preemptive, which means that the execution of the current job may be preempted by another job with **higher** priority. a newly arriving job can preempt a job currently running only if its priority is **higher** than this job)



Construct a Gantt chart depicting the scheduling for the set of processes specified in the above diagram using this 3-level feedback queue, and compute the average waiting time for all processesTT (10 points)

**Answer:**



The average waiting time =

|  |  |
| --- | --- |
| P1 | 17 |
| P2 | 28 |
| P3 | 3 |
| P4 | 14 |
| P5 | 17 |
| P6 | 3 |
| Average | 13.667 |

* 1. (20 points) Consider the following set of processes, with the length of the CPU burst time given in milliseconds:

|  |  |  |
| --- | --- | --- |
| Process | Arrival Time(ms) | Burst Time(ms) |
| P1 | 0 | 4 |
| P2 | 2 | 3 |
| P3 | 3 | 1 |
| P4 | 6 | 2 |
| P5  P6 | 7  11 | 7  4 |

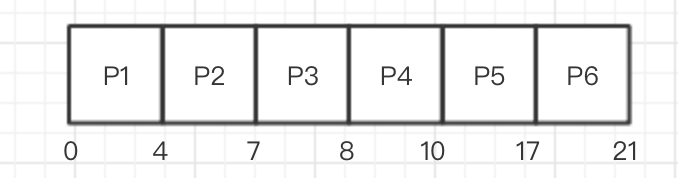
1. (10 points) Draw four Gantt charts that illustrate the execution of these processes using the scheduling algorithms listed below:
2. FCFS
3. SJF
4. Non-Preemptive priority (a smaller priority number implies a higher priority), with the priorities listed here:

|  |  |
| --- | --- |
| Process | Priority |
| P1 | 1 |
| P2 | 4 |
| P3 | 2 |
| P4 | 5 |
| P5  P6 | 1  3 |

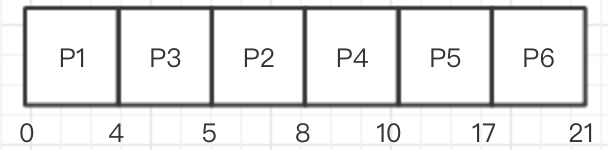
1. RR (quantum = 4)

**Answer:**

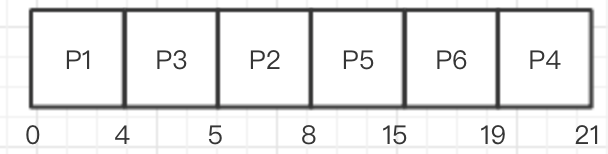
1. FCFS



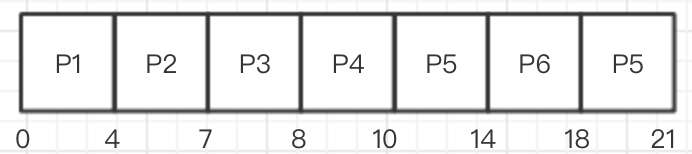
1. SJF



(iii) Non-Preemptive priority



(iv) RR (quantum = 4)



1. (4 points) What is the turnaround time of each process for each of the scheduling algorithms in part a?

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Turnaround time* | P1 | P2 | P3 | P4 | P5 | P6 |
| FCFS |  |  |  |  |  |  |
| SJF |  |  |  |  |  |  |
| Non-Preemptive priority |  |  |  |  |  |  |
| RR |  |  |  |  |  |  |

**Answer:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Turnaround time* | P1 | P2 | P3 | P4 | P5 | P6 |
| FCFS | 4 | 5 | 5 | 4 | 10 | 10 |
| SJF | 4 | 6 | 2 | 4 | 10 | 10 |
| Non-Preemptive priority | 4 | 6 | 2 | 15 | 8 | 8 |
| RR | 4 | 5 | 5 | 4 | 14 | 7 |

1. (4 points) What is the waiting time of each process for each of these scheduling algorithms in part a?

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Waiting time* | P1 | P2 | P3 | P4 | P5 | P6 |
| FCFS |  |  |  |  |  |  |
| SJF |  |  |  |  |  |  |
| Non-Preemptive priority |  |  |  |  |  |  |
| RR |  |  |  |  |  |  |

**Answer:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Waiting time* | P1 | P2 | P3 | P4 | P5 | P6 |
| FCFS | 0 | 2 | 4 | 2 | 3 | 6 |
| SJF | 0 | 3 | 1 | 2 | 3 | 6 |
| Non-Preemptive priority | 0 | 3 | 1 | 13 | 1 | 4 |
| RR | 0 | 2 | 4 | 2 | 7 | 3 |

1. (2 points) Which of the algorithms results in the minimum average waiting time (over all processes) mentioned above?

**Answer: SJF**