CISC2005 Principles of Operating Systems

Assignment 2

Release date: Feb 28, 2023

Due date: Mar 12, 2023 23:59

No late assignment will be accepted

Every Student MUST include the following statement, together with his/her signature in the submitted homework.

I declare that the assignment submitted is original except for source material explicitly acknowledged, and that the same or related material has not been previously submitted for another course. I also acknowledge that I am aware of University policy and regulations on honesty in academic work, and of the disciplinary guidelines and procedures applicable to breaches of such policy and regulations.

Signed (Student YW) Date 6/3/23

Name WONG YAT WAN SID DC026157

General homework policies:

A student may discuss the problems with others. However, the work a student turns in must be created COMPLETELY by oneself ALONE. A student may not share ANY written work or pictures, nor may one copy answers from any source other than one's own brain.

Each student **MUST LIST** on the homework paper the **name of every person he/she has discussed or worked with**. If the answer includes content from any other source, the student **MUST STATE THE SOURCE**. Failure to do so is cheating and will result in sanctions. Copying answers from someone else is cheating even if one lists their name(s) on the homework.

If there is information you need to solve a problem but the information is not stated in the problem, try to find the data somewhere. If you cannot find it, state what data you need, make a reasonable estimate of its value, and justify any assumptions you make. You will be graded not only on whether your answer is correct, but also on whether you have done an intelligent analysis.

Questions: (cut down).

1. Does multi-threading programming always perform better than single-threaded programming? Why?

2. Consider a multi-programmed system with degree of 5 (i.e., five programs in memory at the same time). Assume that each process spends 80% of its time waiting for I/O. What will be the CPU utilization?

3. Consider a system that has a CPU-bound process, which requires a burst time of 40 seconds. The multilevel feedback queue scheduling algorithm is used and the system consists of five queues where the time quantum of

queues from high-priority to low-priority are 2, 7, 12, 17, 22 respectively. Then how many times the process will be interrupted and in which queue the process will terminate the execution?

4. For the following mix of jobs, lengths, and arrival times, determine the scheduling order and average wait time for the First Come First Served (FCFS), Round Robin (RR), and non-preemptive Shortest Job First (SJF) schedulers. Assume all times are in msec. For the RR scheduler, use a 5 msec time slice and a 0 msec context switch cost; assume that when new tasks arrive they are placed at the head of the queue for jobs waiting to be scheduled. Fill the value of each job in each scheme in the table.

			Completion Time			Waiting Time		
Job	Arrival Time	Length	FCFS	RR	SJF	FCFS	RR	SJF
A	0	40						
B	5	15						
C	20	25						
D	40	20						

5. Consider the following code segment:

```
pid_t pid;
pid = fork();
if (pid == 0) { /* child process */
        fork();
        thread create( . . .);
}
fork();
```

(1) How many unique processes are created?

(2) How many unique threads are created?

6. Write a multi-threaded program that calculates various statistical values for a list of numbers. This program will be passed a series of numbers on the command line and will then create three separate worker threads. One thread will determine the average of the numbers, the second will determine the maximum value, and the third will determine the minimum value. For example, suppose your program is passed the integers 90 81 78 95 79 72 85 , the program will report

The average value is 82

The minimum value is 72

The maximum value is 95

The variables representing the average, minimum, and maximum values should be stored globally. The worker threads will set these values, and the parent thread will output the values once the workers have exited.

1

In my perspective, I don't think multi-threading programming always perform better than single-threaded programming, this is because multi thread programming is introduced to solved particular problem, in which the task need share resources (data & code), like parallel programs. Therefore, we cannot conclude that multi thread programming is better, because different task have different nature of problems.

For example, if it's solving parallel program, most probably multi thread is better than single thread because it can make CPU more efficient. However, if today we are solving normal program that doesnt share resources, then multi thread might not 100% better than single thread.

2

CPU utilisation formula : 1 - p^n (CPU utilisation is given as 1 - probability of CPU to being in the idle state)
Process 1 : 80% Idle
Process 2 : 80% Idle
Process 3 : 80% Idle
Process 4 : 80% Idle
Process 5 : 80% Idle
So, 1 - (80%)^5 = 67.23%

3

Task need burst time of 40 seconds, using MFQS algorithm. Assume there is no other process in during execution.

Highest priority Queue 1 (time quantum of 2s) : when got in is 40s, execute for 2s, remaining 38s, go to lower priority queue
Queue 2 (time quantum of 7s) : when got in is 38s, execute for 7s, remaining 31s, go to lower priority queue
Queue 3 (time quantum of 12s) : when got in is 31s, execute for 12s, remaining 19s, go to lower priority queue
Queue 4 (time quantum of 17s) : when got in is 19s, execute for 17s, remaining 2s, go to lower priority queue
Lowest priority Queue 5 (time quantum of 22s) : when got in is 2s, execute for 2s, terminate

Therefore, the process is interrupted 4 times, will terminate at Lowest priority queue 5.

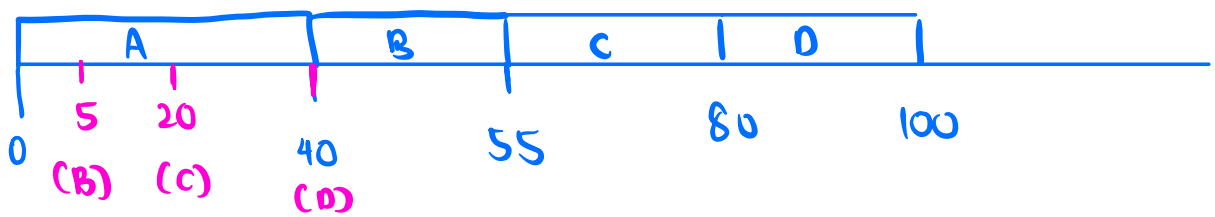
(4)

			Completion Time			Waiting Time		
Job	Arrival Time	Length	FCFS	RR	SJF	FCFS	RR	SJF
A	0	40	40	100	40	0	60	0
B	5	15	55	75	55	35	15	35
C	20	25	80	90	100	35	45	55
D	40	20	100	95	75	40	35	15

a)

FCFS

Arr indicate arrival time



A waiting time = 0

A completion time = 40

B waiting time = 40 - 5 = 35

B completion time = 55

C waiting time = 55 - 20 = 35

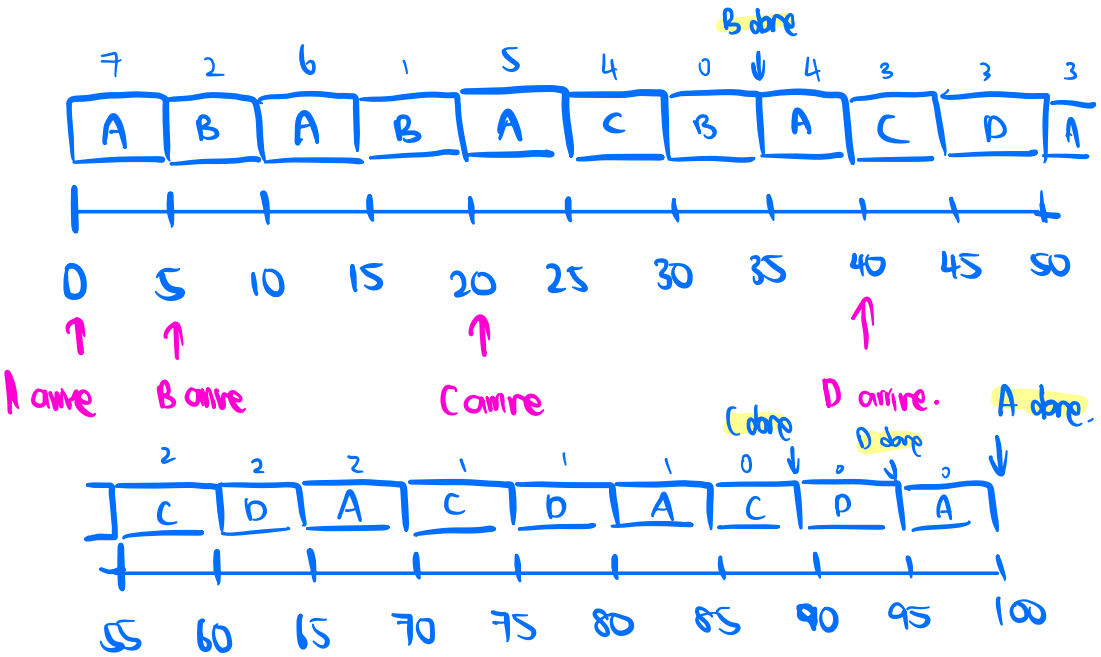
C completion time = 80

D waiting time = 80 - 40 = 40

D completion time = 100

b) RR

* turn around time = exit - arrive
  waiting time = turn around - burst time
* ready queue



A: 40/5 = 8
B: 15/5 = 3
C = 25/5 = 5
D = 20/5 = 4

ready queue

A, B, A, B, A, C, B, A, C,
D, A, C, D, A, C, D, A,
C, D, A

---

turn around time = exit - arrive

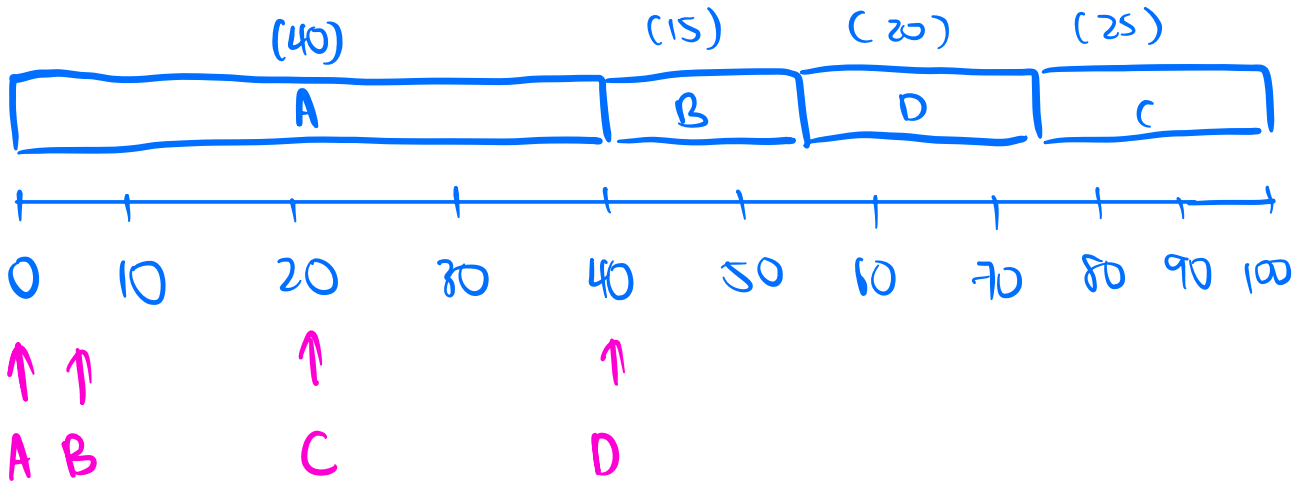
A: 100 - 0 = 100
B: 35 - 5 = 30
C: 90 - 20 = 70
D: 95 - 40 = 55

waiting time = turn around time - burst time

A: 100 - 40 = 60
B: 30 - 15 = 15
C: 70 - 25 = 45
D: 55 - 20 = 35

c)

non preemptive SJF



∴ completion time :

A : 40

B : 55

C : 100

D : 75

∴ waiting time :

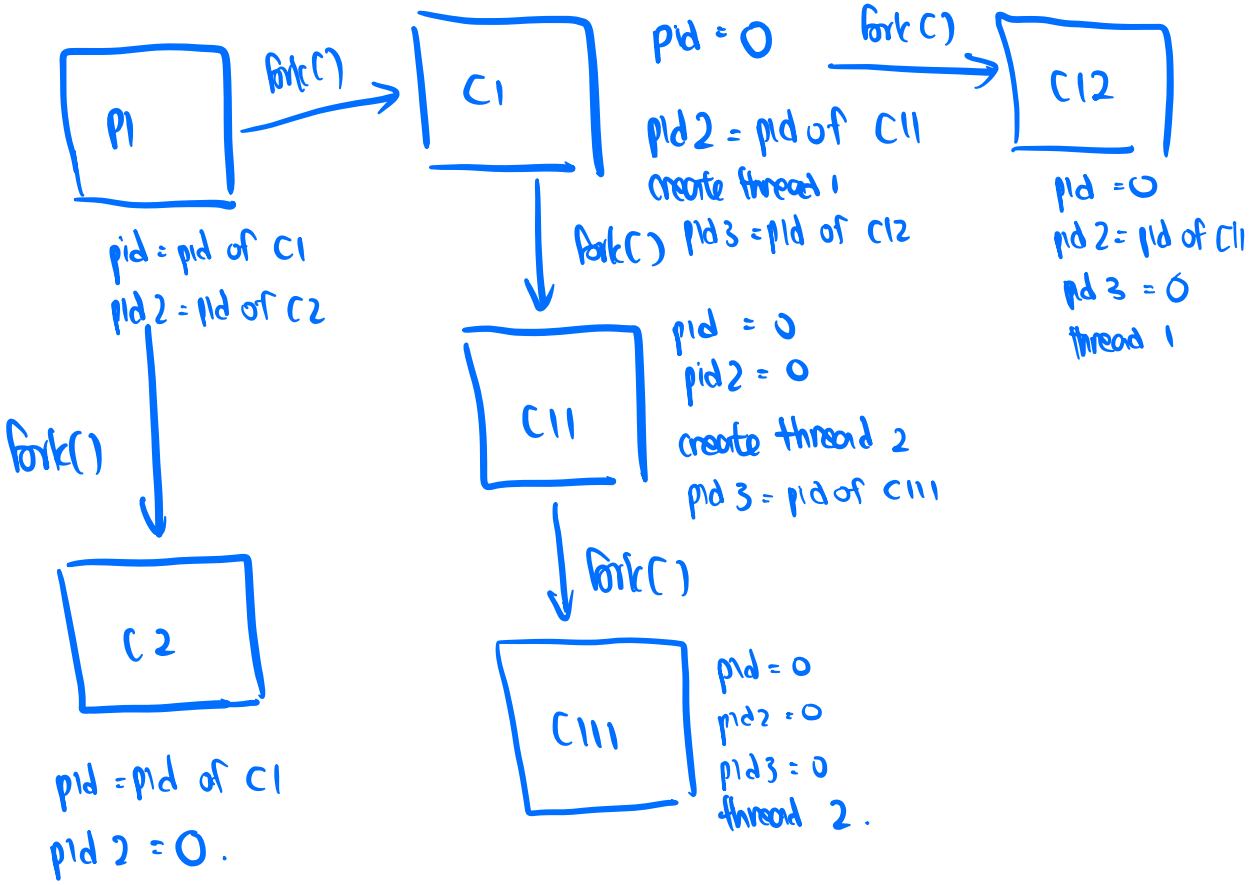
A : 0

B : 40 - 5 = 35

C : 75 - 20 = 55

D : 55 - 40 = 15

(5)



1) 6 unique processes, including parents

2) 2 unique threads