Fall 2023 COMP 3511 Homework Assignment 2 (HW2)

Handout Date: October 5, 2023. Due Date: October. 19, 2023

Name	**Chloe Hu**
Student ID	**21044009**
ITSC email	chuap@connect.ust.hk

Please read the following instructions carefully before answering the questions:

- You should finish the homework assignment **individually**.
- This homework assignment contains **four** parts:

1) Multiple choices 2) Short Answer 3) C programs on pthread 4) CPU Scheduling

- **Homework Submission:** submit to **Homework #2** on **Canvas**.
- TA responsible for HW2: Yue Deng(ydengbi@cse.ust.hk)

1. [20 points] Multiple choices

Write your answers in the boxes below:

MC1	MC2	MC3	MC4	MC5	MC6	MC7	MC8	MC9	MC10
D	**A**	**A**	**D**	**C**	**A**	**C**	**B**	**B**	**D**

1.1 Which of the following statements is not true for a multi-threaded process?
A) A thread within a process is an active entity, each with its own unique thread identifier, program counter, registers, and stacks
B) The thread within a process is referred as a lightweight process
C) Threads belonging to one process share resources such as code, data, and files
D) Multiple single-threaded processes cannot complete the tasks of one multi-threaded process

1.2 If an application is 70% parallel and 30% serial, which of the following values is a possible speedup when moving the application from one core to four cores, according to Amdahl's Law?
A) 2.1
B) 3.5
C) 5.8
D) 10.6

TA responsible for HW2: Yue Deng(ydengbi@cse.ust.hk)

1.3 Which of the following statements is not true between user threads and kernel threads?
A) User threads are only visible to programs
B) A user thread needs to be mapped a kernel thread before execution
C) A user thread is uniquely determined by a thread control block or TCB
D) OS manages kernel threads including allocating the required resources

1.4 Which of the following statements is true for `clone()` in Linux?
A) It does not duplicate the address space of the parent process
B) It can create a task that only shares open files with the calling task, i.e., parent process
C) It offers the flexibility to specify the content of sharing in the newly created task
D) All of the above

1.5 Which of the following conditions is considered to be preemptive in CPU scheduling?
A) A process terminates
B) A timer expires
C) A process blocks when issuing I/O operation
D) A process execute wait()

1.6 Which of the following scheduling algorithms do not incur starvation?
A) FCFS
B) SRTF
C) MLFQ
D) SJF

1.7 If the CPU burst times of all processes are the same, which of the following statements is true for different scheduling algorithms in term of the average turn-around time?
A. FCFS performs the worst
B. SJF performs better than FCFS
C. RR with a quantum smaller than the CPU burst time performs the worst
D. SRTF performs better than SJF

1.8 Which of the following statements is not true on thread scheduling i.e., process-contention scope (PCS)?
A) There is no need for thread scheduling under one-to-one mapping
B) Thread library is responsible for adjusting the priority of a user thread
C) Thread scheduling determines how user thread is mapped to kernel thread
D) Thread scheduling is typically preemptive in favor of a higher-priority user-level thread

1.9 Which of the following scheduling is handled by CPU scheduling?
A. A user thread is scheduled onto a kernel thread
B. A software thread (also a kernel thread) is scheduled on a hardware thread (also called a logical CPU)
C. A hardware thread is scheduled to run on a CPU core
D. All of the above

1.10 Which of the following statements is not true for EDF scheduling?

A. EDF does not require that processes be periodic
B. EDF requires that a process announces its deadline to the scheduler in advance
C. Processes have dynamic priorities in EDF scheduling
D. EDF requires that the CPU burst time of each process is a constant

2. [20 points] Please answer the following questions in a few short sentences

2.1 (5 points) Please describe the pros and cons of many-to-many mapping scheme.

Many to many provides a certain level of concurrency for process execution, allowing the OS to create enough kernel threads in advance. Many to many is useful when there are resource limitations or when there are less important tasks to carry out. The cons are that many to many is more complicated than one to one. It is harder to debug and can cause blocking if one thread can't run, holding up other threads in the queue.

2.2 (5 points) Please briefly compare a task in Linux and a process in Windows, single-threaded or multi-threaded

In Linux, a thread is a special process, called a task, which may share resources (ie. address space) with other tasks. Each thread has a unique task_struct and appears as a normal process. In Linux the clone() system call can be called to specify the specific resources to be shared. Clone() only duplicates the resources when data is changed, otherwise data is shared with he parent task.

On the other hand, a thread in Windows represents a basic unit of CPU utilisation that includes a threadID, a register set, program counter, separate user and kernel stacks and private storage. Window threads uses one to one mapping. The thread has three primary data structures: Ethread, Kthread and the TEB. Multiple threads can exist within the same process and share resources such as memory.

To compare the two, consider a process containing two threads: On Windows, there would be 1 PCB describing the shared resources that points to two different threads. Each thread has its own TCB that describes the resources it alone possesses. In Linux, there would be two task_structs which share resources.

2.3 (5 points) Please describe the advantages in MLFQ scheduling

MLFQ scheduling does not need prior knowledge on the next CPU burst time, it handles interactive jobs well and is fair by making progress on CPU bound jobs. It is also flexible as it can integrate both aging and different scheduling algorithms for each queue. It can handle starvation by reshuffling jobs to different queues periodically.

2.4 (5 points) Please briefly contrast FCFS and RR scheduling in terms of response time and the average turn-around time.

In FCFS processes are executed in the order they arrive, accordingly, the average response time is larger compared to FF since the latter processes need to wait for the earlier processes to finish completely. On the other hand, RR scheduling is preemptive and allocates a fixed time quantum to each process in a cyclic manner. In RR scheduling, every process gets an equal share of the CPU for a fixed time, causing the average response time for each process to be lower compared to FCFS.

In terms of average turn-around time, FCFS may lead to longer turn-around times for processes due to the Convoy Effect. Contrastingly, RR aims to provide better average turn-around times by giving each process a fair share of the CPU. Since each process gets a fixed quantum, the waiting time for each process is distributed more evenly, leading to reduced turn-around times, especially for processes that have shorter burst times.

3. [10 points] C Programs on Pthread

For all the C programs below, you can assume that necessary header files are included, and `pthread_create()` always successfully creates a new process or thread. Consider the following code segments:

```
int a[3] = {1, 2, 3};
void *increment(void *rank) {
    int index = *(int *)rank;
    a[index] = a[index] + index + 1;
    printf("%d", a[index]);
    pthread_exit(0);
}
int main() {
    pthread_t tid[3];
    int rank[3] = {0, 1, 2};
    for (int i = 0; i < 3; i++) {
    pthread_create(&tid[i], NULL, increment, (void *)&rank[i]);
    }

    for (int i = 0; i < 3; i++) {
        pthread_join(tid[i], NULL);
    }
    return 0;
}
```

Please write all possible outputs of the program (6 points) and explain them (4 points)
Possible outputs of the program: 246, 264, 426, 462, 624, 642 (all the permutations of the numbers 2,4,6)
Explanation:

TA responsible for HW2: Yue Deng(ydengbi@cse.ust.hk)

The different outputs are generated because the threads in the program are scheduled in a non-deterministic fashion. The thread scheduler, responsible for assigning CPU time to threads, behaves unpredictably due to factors such as interrupts, thread priorities, and the current CPU usage. Each time the program runs, the scheduler may decide to schedule the threads in a different order. This variability in scheduling leads to different outputs being produced.

4 [50 points] CPU Scheduling

4.1 (20 points) Consider the following single-thread process, arrival times, and CPU process requirements:

Process	Arrival Time	Burst Time
P_1	0	8
P_2	2	10
P_3	6	12
P_4	12	4
P_5	18	1

- We consider 4 algorithms: FCFS, RR, SJF, SRTF.
- The time quantum of the Round-Robin (RR) is 4.
- Assume that context switch overhead is 0.
- When a process arrives, it is immediately eligible for scheduling, e.g., process 2 that arrives at time 2 can be scheduled during time unit 2.
- Whenever there is a tie among processors (same arrival time, same remaining time, etc), they are inserted into the ready queue in the ascending order of process id. That is, if process 1 and process 2 arrive at the same time, process 1 is inserted first, and process 2 second in the ready queue.

For each scheduling algorithm, draw the Gantt charts depicting the sequence of the process execution, and calculate the average turnaround time.

FCFS:

TA responsible for HW2: Yue Deng(ydengbi@cse.ust.hk)

	FCFS				
	p1	p2	p3	p4	p5
0	8	18	30	34	35

	WT	TAT
P1	0	8
P2	6	16
P3	12	24
P4	18	22
P5	16	17
	AVG TAT	17.4

RR:

	RR									
	p1	p2	p1	p3	p2	p4	p3	p5	p2	p3
0	4	8	12	16	20	24	28	29	31	35

	WT	TAT
P1	4	12
P2	19	29
P3	17	29
P4	8	12
P5	10	11
	AVG TAT	18.6

SJF:

	SJF				
	p1	p2	p3	p4	p5
0	8	18	30	34	35

	WT	TAT
P1	0	8
P2	6	16
P3	12	24
P4	18	22
P5	16	17
	AVG TAT	17.4

SRTF:

	SRTF							
	p1	p2	p4	p2	p5	p2	p3	
0	8	12	16	18	19	23	35	

	WT	TAT
P1	0	8
P2	11	21
P3	17	29
P4	0	4
P5	0	1
	AVG TAT	12.6

TA responsible for HW2: Yue Deng(ydengbi@cse.ust.hk)

4.2 (20 points) Consider a multi-level feedback queue scheduler with 3 queues:

- Q_0 - RR with time quantum 4 ms.
- Q_1 - RR with time quantum 8 ms.
- Q_2 - FCFS.

For the following single-thread processes, calculate the turn-around time and decide on which queue the process terminates. Draw the Gantt chart depicting the scheduling procedures for these processes.

Process	Arrival Time (ms)	Burst Time (ms)
P_1	0	7
P_2	6	3
P_3	8	17
P_4	15	8
P_5	24	5

Fill in the turn-around time and the terminating queue in the following table:

Process	Turn-around Time (ms)	Terminating Queue
P_1	14	Q1
P_2	3	Q0
P_3	32	Q2
P_4	19	Q1
P_5	11	Q1

Draw the Gantt Chart:

	MLFQ													
	p1	p1	p2	p3	p1	p3	p4	p3	p5	p3	p4	p5	p3	
0	4	6	9	13	14	15	19	24	28	30	34	35	40	
	WT	TAT												
P1	7	14												
P2	0	3												
P3	15	32												
P4	11	19												
P5	6	11												
	AVG TAT	15.8												

TA responsible for HW2: Yue Deng(ydengbi@cse.ust.hk)

4.3 (10 points) Consider a system with 5 processes and a scheduling algorithm that combines Priority Scheduling with Round Robin. The processes have the following burst times and priorities:

Process	Burst Time (ms)	priority
P_1	5	1
P_2	9	2
P_3	3	1
P_4	6	2
P_5	3	3

Assume a time quantum of 2 ms for the Round Robin scheduling. Whenever there is a tie among processes, they are executed in the ascending order of process id. Draw the Gantt chart depicting the scheduling procedures for these processes and determine the average turn-around time.

	PS + RR															
	p1	p3	p1	p3	p1	p2	p4	p2	p4	p2	p4	p2	p2	p5	p5	
0	2	4	6	7	8	10	12	14	16	18	20	22	23	25	26	
	WT	TAT														
P1	3	8														
P2	14	23														
P3	4	7														
P4	14	20														
P5	23	26														
	AVG TAT	16.8														