Fall 2022 COMP 3511 Homework Assignment 2 (HW2)

Handout Date: October 10, 2022, Due Date: October 24, 2022

Name	Casper Kristiansson
Student ID	20938643
ITSC email	cok@connect.ust.hk

Please read the following instructions carefully before answering the questions:

- You should finish the homework assignment **individually**.
- All programs should be executed in a CS Lab 2 machine.
- **<u>Homework Submission</u>:** submitted to **Homework #2** on **Canvas**.
- TA responsible for HW2: Weijie Sun (wsunan@cse.ust.hk)

1. [20 points] Multiple Choices.

Please write down your answers in the boxes below:

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
D	A	C	C	A	D	B	D	A	B

1) Which of the following is the statement is TRUE during process termination?
A. All processes transition into *Zombie state* before termination
B. The corresponding entry in the *Process Table* or *Process List* still exists after a process call exit() before its parent process call wait()
C. A process becomes an *orphan* process when its parent process terminates without invoking wait().
D. All of the above.

2) Suppose that a host with IP address 150.55.66.77 wishes to access a Web server with IP address 202.28.15.123. Which of the following socket pair is valid for a connection between this pair of hosts?
A. 150.55.66.77:1600 and 202.28.15.123:80
B. 150.55.66.77:200 and 202.28.15.123:80
C. 150.55.66.77:80 and 202.28.15.123:80
D. 150.55.66.77:1500 and 202.28.15.123:150

3) Which of the following statements on pipe communication is TRUE?
A. Named pipes are automatically deleted after the communication ends.
B. Communications are uni-directional for named pipes
C. Ordinary pipes can be used for communications between parent and child processes
D. Ordinary pipes can be used by communicating processes on different machines.

4) Which of the following statements is TRUE for a multi-threaded process?
A. The process state can be uniquely determined

B. Threads of the same process share stack with each other
C. Thread is the basic unit of CPU utilization
D. Threads belonging to a process can exist independently of the process

5) CPU scheduling may take place in the following instances; which instance is considered to be *non-preemptive*?
A. The completion of an I/O operation of a process (waiting to ready state)
B. A running process request an I/O operation
C. A running process is interrupted
D. A new process joins the system (i.e., from new to ready state)

6) Which of the following scheme is an acceptable signal handling scheme for a multithreaded program?
A. Deliver the signal to the thread to which the signal applies.
B. Deliver the signal to every thread in the process.
C. Deliver the signal to only certain threads in the process.
D. All of the above

7) Which of the following mapping scheme from user threads to kernel threads does not require scheduling, i.e., process-contention scope or PCS?
A. Many-to-One
B. One-to-One
C. Many-to-Many
D. Two-level mapping model

8) Which of the following statement on MLFQ scheduling is TRUE?
A. It is fair in the sense that all processes can make progress
B. It does not have the *convey effect*
C. Its performance resembles SJF scheduling without the need to estimate the next CPU burst time
D. All of the above

9) Which of the following statement is TRUE for a *rate-monotonic* scheduling algorithm?
A. It only deals with periodic processes or tasks
B. A process with a shorter period will have lower priority
C. It is non-preemptive in nature
D. It uses a dynamic priority policy

10) What is the inherent problem with priority scheduling algorithms?
A. complexity
B. Starvation
C. How to determine the length of the next CPU burst
D. How to determine the length of the time quantum

2. [20 points] CPU Scheduling.

Consider the following set of processes, with the length of the CPU burst time given in milliseconds:

Process	Arrival Time	Burst Time
P1	0	6
P2	1	3
P3	8	6
P4	13	6
P5	16	3
P6	20	2

Draw Gantt charts for the scheduling algorithms listed below and compute the average turnaround time and the average waiting time for each algorithm.

1) FCFS
2) SJF
3) SRTF
4) RR (time quantum = 3)

1)

FCFS

P1	P2	P3	P4	P5	P6

0 6 9 15 21 24 26

Process	Waiting	Turnaround
P1	0	6
P2	5	8
P3	1	7
P4	2	8
P5	5	8
P6	4	6

Average Waiting Time: (0+5+1+2+5+4)/6 = 2.833ms

Average Turnaround Time: (6+8+7+8+8+6)/6 = 7.167ms

Process	Arrival Time	Burst Time
P1	0	6
P2	1	3
P3	8	6
P4	13	6
P5	16	3
P6	20	2

2)

SJF

P1	P2	P3	P4	P6	P5

0 6 9 15 21 23 26

Process	Waiting	Turnaround
P1	0	6
P2	5	8
P3	1	7
P4	2	8
P5	7	10
P6	1	3

Average Waiting Time: (0+5+1+2+7+1)/6 = 2.667ms

Average Turnaround Time: (6+8+7+8+10+3)/6 = 7ms

Process	Arrival Time	Burst Time
P1	0	6
P2	1	3
P3	8	6
P4	13	6
P5	16	3
P6	20	2

3)

SRTF

P1	P2	P1	P3	P4	P5	P4	P6	P4

0 1 4 9 15 16 19 20 22 26

Process	Waiting	Turnaround
P1	3	9
P2	0	3
P3	1	7
P4	7	13
P5	0	3
P6	0	2

Average Waiting Time: (3+0+1+7+0+0)/6 = 1.833ms

Average Turnaround Time: (9+3+7+13+3+2)/6 = 6.167ms

Process	Arrival Time	Burst Time
P1	0	6
P2	1	3
P3	8	6
P4	13	6
P5	16	3
P6	20	2

4)

RR (time quantum = 3)

P1	P2	P1	P3	P3	P4	P5	P6	P4

0 3 6 9 12 15 18 21 23 26

Process	Waiting	Turnaround
P1	3	9
P2	2	5
P3	1	7
P4	7	13
P5	2	5
P6	1	3

Average Waiting Time: (3+2+1+7+2+1)/6 = 2.667ms

Average Turnaround Time: (9+5+7+13+5+3)/6 = 7ms

Process	Arrival Time	Burst Time
P1	0	6
P2	1	3
P3	8	6
P4	13	6
P5	16	3
P6	20	2

3. [15 points] Priority Scheduling with Round Robin.

Draw Gantt charts for priority scheduling with RR (time quantum = 3) and compute the average turnaround time and the average waiting time.

Note: (1) a smaller (larger) number indicates a higher (lower) priority. (2) When a low priority process is interrupted by a higher priority process, the low priority process is placed at the ***head*** of its RR queue; when it resumes execution on a CPU, it continues to run with the remaining quantum instead of getting a full (new) quantum.

Process	Arrival Time	Burst Time	Priority
P1	0	8	5
P2	1	3	5
P3	8	7	1
P4	11	6	2
P5	15	3	2
P6	16	2	1

RR (time quantum = 3) With Priority

P1	P2	P1	P3	P4	P6	P4	P5	P4	P1
0	3	6	8	15	16	18	20	23	26

(end: 29)

Process	Waiting	Turnaround
P1	21	29
P2	2	5
P3	0	7
P4	9	15
P5	5	8
P6	0	2

Average Waiting Time: (21+2+0+9+5+0)/6 = 6.167ms

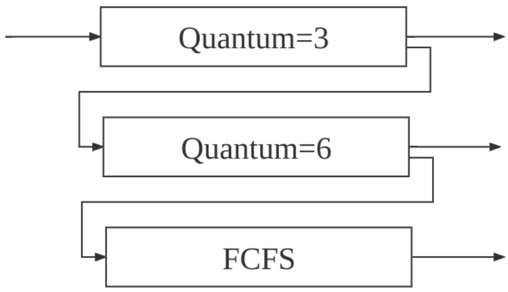
Average Turnaround Time: (29+5+7+15+8+2)/6 = 11ms

Process	Arrival Time	Burst Time	Priority
P1	0	8	5
P2	1	3	5
P3	8	7	1
P4	11	6	2
P5	15	3	2
P6	16	2	1

4. [25 points] Multi-Level Feedback Queue

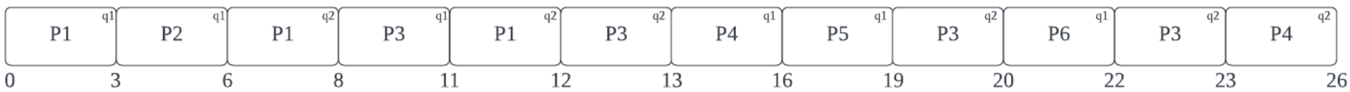
Draw Gantt charts for a MLFQ scheduling and compute the average turnaround time and the average waiting time. The three queues are defined as follows.

1) Q0: RR with time quantum 3
2) Q1: RR with time quantum 6
3) Q2: FCFS



Process	Arrival Time	Burst Time
P1	0	6
P2	1	3
P3	8	6
P4	13	6
P5	16	3
P6	20	2

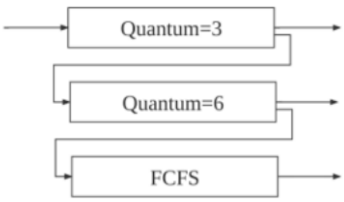
MLFQ



Process	Waiting	Turnaround
P1	6	12
P2	2	5
P3	9	15
P4	7	13
P5	0	3
P6	0	2

Average Waiting Time: (6+2+9+7+0+0)/6 = 4ms

Average Turnaround Time: (12+5+15+13+3+2)/6 = 8.333ms



Process	Arrival Time	Burst Time
P1	0	6
P2	1	3
P3	8	6
P4	13	6
P5	16	3
P6	20	2

5. [20 points] Fork and Pipe Programming

Consider the following C program:

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/wait.h>
int main() {
    char msg[50] = "empty message";
    pid_t pid = fork();
    if ( pid > 0 ) { // parent
        wait(0); // wait for the child
        printf("The message in parent is %s\n", msg);
    } else { // child
        strcpy(msg, "secret message");
        printf("The message in child is %s\n", msg);
    }
    return 0;
}
```

1) What is the output of the above program. Briefly explain the output of the program

Output:
The message in child is secret message
The message in parent is empty message

The program declares a msg char array. It then creates a child process using (`pid_t pid = fork();`). The parent will then enter the first if statement because the pid is not equal to 0 and wait until the child is finished running. The child will then copy the "secret message" into the msg char array and print the result. The parent will then print the original message. This is because the information is not shared between the processes so even though the child updates the msg char array it won't affect the parents msg char array. The shared memory is not updated.

2) Fill in the missing blanks of the following program so that the output of the program becomes:

```
The message in child is secret message
The message in parent is secret message
```

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/wait.h>

int main() {
    char msg[50] = "empty message";
    const char secret_msg[] = "secret message";
    int secret_length = strlen(secret_msg);
    int pfds[2];

    __________________________ // Hint: invoke pipe() here
    pid_t pid = fork();
    if ( pid > 0 ) { // parent
        wait(0); // wait for the child

        ______________________ // Hint: invoke close() here

        ______________________ // Hint: invoke read here
        printf("The message in parent is %s\n", msg);
    } else { // child
        strcpy(msg, secret_msg);

        ______________________ // Hint: invoke close() here

        ______________________ // Hint: invoke write here
        printf("The message in child is %s\n", msg);
    }
    return 0;
}
```

Line	Code
1	pipe(pfds);
2	close(0);
3	read(pfds[0], msg, secret_length);
4	close(0);
5	write(pfds[1], secret_msg, secret_length);