

COMP 3511 Operating System (Spring 2022)

Midterm Exam

Date: 01-Apr-2022 (Fri)

Time: 19:00 – 21:00 (2 hours)

Name (Write the name printed on your Student ID card)	
Student ID	
ITSC email	

Exam format and rules:

- It is an open-book, open-notes exam (Reference: Chapter 1)
- Allowed devices: a device to record your Zoom video and a computer
- In the last 15 minutes (i.e. starts at 8:45 pm), you can remove the camera to take pictures of your hand-written answers
- Submit your work via Canvas using doc/docx/pdf format
- Other details are already sent to students via midterm related emails

Part I. Multiple Choices [25 * 1 points]

Write down your answers in the boxes below:

MC1	MC2	MC3	MC4	MC5	MC6	MC7	MC8	MC9	MC10

MC11	MC12	MC13	MC14	MC15	MC16	MC17	MC18	MC19	MC20

MC21	MC22	MC23	MC24	MC25

MC1. Which is not part of the Von Neumann architecture ?

- A. Control unit
- B. Memory
- C. Arithmetic Logic Unit
- D. Operating system

MC2. Which of the following is false about cache ?

- A. The access time of cache is faster than register
- B. Cache hit ratio means the percentage of content found in cache
- C. If the information required are not in cache, data will be copied from a lower storage.
- D. Cache will keep the recently accessed data items closer to processor.

MC3. Which of the following is not one of the advantages of multiprocess systems compared to single process systems?

- A. Economy of scale
- B. Fast on-chip communication
- C. More stable
- D. Can share other devices such as I/O devices.

MC4. Which of the following is false about service of operating systems?

- A. Communication service of OS is responsible for the inter process communication.
- B. The logging service of OS should keep track of computer resources that each user use.
- C. Protection and security service of OS means OS must take appropriate actions to ensure correct and consistent computing
- D. I/O operations often use interrupt to handle requests from I/O device.

MC5. Which of the following is true about system calls?

- A. It is better to invoke system calls directly rather than using APIs.
- B. The parameters of system call can only be passed by using registers.
- C. Command interpreter is a kind of system call.
- D. System calls are generally available as functions written in a high-level language

MC6. Which of the following statement is false about security and protection in operating systems?

- A. OS must protect itself from user programs
- B. Security means the defence of the system against all kinds of internal attacks.
- C. OS should have the ability to against external attacks like viruses and malware
- D. An access right is the permission to perform an operation on an object.

MC7. Which of the following statement is false about the OS structure?

- A. The little overhead in the system-call interface and fast communication in layered structure result in a good efficiency of the system
- B. The stability of microkernel system is better than monolithic design because the failure of a kernel does not affect other kernels.
- C. When two user-level services communicate with each other in microkernel system, message must be copied between them.
- D. Hybrid systems combine multiple approaches to address performance, security, and usability needs

MC8. We know that a process has several different states. Which of the following options is a process state change when the scheduler selects a program from the ready queue to run?

- A. running → ready
- B. ready → running
- C. running → waiting
- D. waiting → ready

MC9. Which of the following statements about processes and threads is false?

- A. Named pipes on Windows can only be used by communicating processes on the same machine.
- B. Several processes can communicate using a named pipe.
- C. There is no parent-child relationship in the named pipe.
- D. Named pipe can be accessed without a parent-child relationship.

MC10. Which of the following is true about zombie and orphan?

- A. A parent process waiting for a terminated child is called a zombie
- B. If a parent process terminated without invoking wait(), it is called an orphan.
- C. A zombie process must be already terminated
- D. An orphan process is caused by a wait() function that calls too early.

MC11. How many processes will be created by the following C code fragment? (suppose all *fork()* are successful and necessary header files are included)

```
int main()
{
    if (fork()) fork();
    return 0;
}
```

- A. 2
- B. 3
- C. 4
- D. 6

MC12. What might be the total number of '+' that this program print? Note that this code might generate different outputs.

(suppose all *fork()* are successful and necessary header files are included. Function *fflush(stdout)* is used to print the data in the buffer immediately, so that *fork()* won't copy these unprinted result to the child process.)

```
int main()
{
    while(fork()%2!=0) {
        printf("++");
        fflush(stdout); //print the data in the buffer
    }
    printf("+");
    return 0;
}
```

- A. 1
- B. 3
- C. 4
- D. 5

MC13. Which of the following statement about many-to-one threading model is true?

- A. When a thread in a multithreaded process is blocked, the process in the kernel is blocked because the result of the thread is not available.
- B. Multithreading may not run in parallel on a multicore system.
- C. All threads share one kernel process.
- D. All of the above

MC14. In a system that supports multithreading, what cannot be shared among several threads created by process P?

- A. The code segment of process P
- B. The file opened in process P
- C. The global variable of process P
- D. The stack pointer of a thread in process P

MC15. Which of the following statements about user threads and kernel threads is false?

- A. User threads cannot be scheduled to run on the CPU
- B. The operating system only manages and schedules kernel threads
- C. Kernel thread calls are made when calling the Windows threading library
- D. There is a mapping between user threads and kernel threads.

MC16. According to Amdahl's Law, what is a possible combination of percentage of parallel components and the number of processing cores if the speedup gain is at most 1.25?

- A. percentage of parallel components: 50%, processing cores: 2
- B. percentage of parallel components: 50%, processing cores: 4
- C. percentage of parallel components: 40%, processing cores: 2
- D. percentage of parallel components: 40%, processing cores: 4

MC17. The CPU scheduling algorithm is divided into preemptive and non-preemptive. In this question we discuss the following 4 scheduling algorithms, which of them must be non-preemptive?

- A. SJF
- B. RR
- C. FCFS
- D. priority algorithms

MC18. Consider context switching of processes. Assume there are two processes P0 and P1. PCB0 and PCB1 are the process control block of P0 and P1 respectively. Which of the following time units are included in the dispatch latency during context switching between P0 and P1?

- A. P0 executing
- B. Save state into PCB0, and restore state from PCB1
- C. P1 executing
- D. All of the above

MC19. Which of the following statements about round-robin is true?

- A. The round-robin (RR) scheduling algorithm is similar to FCFS scheduling, but preemption is added to enable system to switch between processes.
- B. If the time quantum is very short, the scheduler will schedule processes frequently, increasing the system overhead.
- C. The average turnaround time does not necessarily improves as the time quantum size increases.
- D. All of the above

MC20. Which of the following can be used for the scheduling strategy of the ready queue?

- A. FCFS
- B. Round-Robin
- C. Priority scheduling
- D. All of the above

MC21. Which of the following is true of the rate-monotonic (RM) and earliest-deadline-first (EDF) scheduling algorithms?

- A. The rationale for RM is to assign a higher priority to tasks requiring CPU more often.
- B. Both RM and EDF are non-preemptive.
- C. RM uses a dynamic priority policy.
- D. EDF uses a static priority policy.

MC22. We have 6 processes arriving at the ready queue at the same time. Their estimated runtimes are: 1, 2, 3, 4, 5 and 6 seconds. Their priorities are 2, 3, 5, 6, 1, and 4. A smaller number means higher priority (e.g., priority 1 has a higher priority than priority 2). Ignoring process switching overhead, calculate the average waiting time for priority scheduling.

- A. 8.33s
- B. 8.67s
- C. 8s
- D. 7.67s

MC23. Which of the following statements about race condition is true?

- A. A race condition will happen only if the outcome of execution does not depend on the order in which instructions are executed.
- B. A race condition happens when several threads try to access the same data concurrently.
- C. A race condition happens when several threads try to access and modify the same data concurrently.
- D. None of the above

MC24. Which of the following statements is false?

- A. Spinlocks can be used to prevent busy waits in semaphore execution.
- B. Counting semaphores can be used to control access to resources with a limited number of instances.
- C. Two standard operations modify Semaphore: wait() and signal().
- D. If a semaphore value is negative, its magnitude is the number of processes currently waiting on the semaphore.

MC25. What is the correct sequence of operations to protect a critical section with a mutex?

- A. release() followed by acquire()
- B. acquire() followed by release()
- C. wait() followed by signal()
- D. signal() followed by wait()

Part II. Calculations [75 points]

1. [25 Points] Process

Suppose all *fork()* are successful and necessary header files are included.

Function *fflush(stdout)* is used to print the data in the buffer immediately, so that *fork()* won't copy these unprinted result to the child process.

1) (8 points) Consider the following program.

```
#include <stdio.h>
#include <unistd.h>

int main() {
    for (int i=0; i<4; i++) {
        fork();
        printf("-");
        fflush(stdout); //print the data in the buffer
    }
    return 0;
}
```

(a) (4 points) What is the total number of process(es) in this program? Briefly explain your answer.

(b)(4 points) How many '-' will be printed on the screen? Briefly explain your answer.

2) (10 points) Consider the following program, where **X** and **Y** will be replaced by different code in the following questions:

```
#include <stdio.h>
#include <unistd.h>

int main() {
    for (int i=0; i<Y; i++) {
        if(X) {
            break;
        }
    }
    return 0;
}
```

(a) (5 points) What is the total number of process(es) if the code in **X** is 'fork()' and **Y** is '3'? Briefly explain your answer.

(b) (5 points) What is the total number of process(es) if the code in **X** is 'i==3' and **Y** is 'fork()+1'? Briefly explain your answer.

3) (7 points) Consider the following C program, where **X** will be replaced by different code in the following questions:

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
int main() {
    pid_t pid;
    pid = fork();
    if (pid!=fork()) {
        printf("%d \n",pid);

    } else {
        printf("%d \n",getpid());
    }
    return 0;
}
```

How many different numbers will be output by the program? Briefly explain your answer.

2. [35 Points] CPU Scheduling

- 1) (13 points) The following table contains 5 processes (P1-P5). Each process has its arrival time, burst time, and priority.

Process	Arrival Time	Burst Time	Priority(Only use in RR)
P ₁	0	3	2
P ₂	1	5	3
P ₃	5	6	2
P ₄	8	3	1
P ₅	10	4	2

- Whenever there is a tie among processes (the same arrival time, remaining time, etc), they are inserted into the ready queue in the ascending order of process id.
- A smaller number means a higher priority (e.g., priority 1 has a higher priority than priority 2)

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

1. FCFS
2. SJF (non-preemptive)
3. Preemptive priority scheduling with RR (with quantum of 2 milliseconds)
4. SRJF

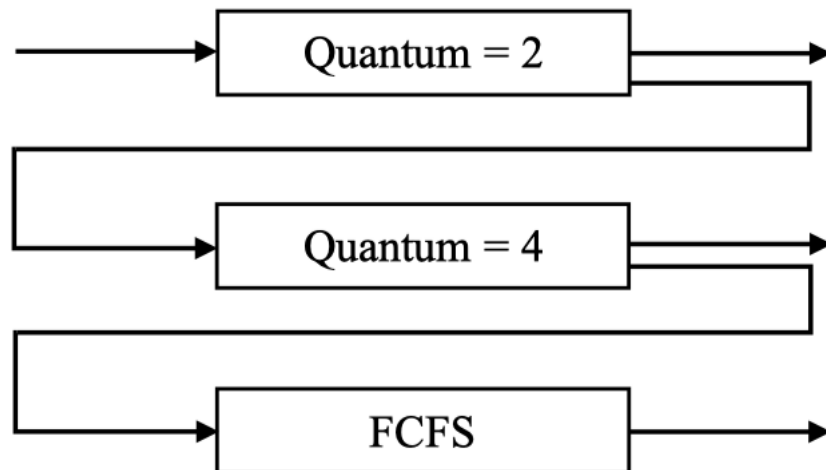
2) (10 points) Consider the following single-thread processes, arrival times, burst times and the following three queues:

Q0 - RR with time quantum 2 milliseconds

Q1 - RR with time quantum 4 milliseconds

Q2 - FCFS

Process	Arrival Time	Burst Time
P ₁	0	5
P ₂	2	15
P ₃	8	7
P ₄	9	4
P ₅	15	16
P ₆	20	6



a) (6 points) Draw the Gantt chart depicting the scheduling procedures for these processes.

b) (4 points) Calculate the average waiting time.

- 3) (12 points) Consider the following single-thread processes, executing times, deadlines and periods. Assume all processes arrive at timeslot 0. Fill in the table with the ID of the process that is running on the CPU with Rate-Monotonic (RM) scheduling and Earliest Deadline First (EDF) scheduling in the first 20 timeslots, and show how many deadlines are missed in each scheduler.

Process	Processing Time	Deadline	Period
P ₁	1	4	5
P ₂	3	7	10
P ₃	2	13	16
P ₄	2	8	8

RM																					
EDF																					
Time	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

3. [15 points] Synchronization

- 1) (5 points) Consider two threads perform non-atomic operations `counter++` and `counter--` respectively. Assume the counter equals to 10 initially. What are possible values of counter when both threads terminate? Please list all possible values.

“counter ++” and “counter --” can be implemented as:

<pre>//counter ++ : register1 = counter register1 = register1 + 1 counter = register1</pre>	<pre>//counter --: register2 = counter register2 = register2 - 1 counter = register2</pre>
--	---

2) (10 points) Readers-Writers Variant Problem

Recall the Readers-Writers Problem. There are some readers and writers share some data. Readers only read the data without any modification, while writers modify the data. Any number of readers can simultaneously read the data, whereas only one writer at a time can write the data. If a writer is writing the data, no reader or writer can access the data. If there is at least one reader reading the data, no writer can write to it.

<pre> int read_count = 0; semaphore mutex = 1; semaphore rw_mutex = 1; writer() { while (1) { wait(rw_mutex); ... /* writing is performed */ ... signal(rw_mutex); } } </pre>	<pre> reader () { while (1) { wait(mutex); read_count++; if (read_count == 1) wait(rw_mutex); signal(mutex) ... /* reading is performed */ ... wait(mutex); read_count--; if (read_count == 0) signal(rw_mutex); signal(mutex); } } </pre>
--	--

The above code segment may starve writers in the queue. This question asks you to solve the Readers-Writers Variant Problem which adds the constraint that no Reader or Writer shall be allowed to starve (ie. FCFS).

Note: We assume semaphores preserve “First Come First Serve” ordering when block and wakeup processes.

- 1) Please fill in the following code.
The code for writer is given. You need an extra semaphore **w**. The semaphore **w** is already defined in the code segment.
- 2) Please describe the role of the semaphore **w**.


```
int read_count = 0;
semaphore mutex = 1;
semaphore rw_mutex = 1;
semaphore w = 1;

void writer(){
    while (1){
        wait(w);
        wait(rw_mutex);
        signal(w);
        ...
        /* writing is performed */
        ...
        signal(rw_mutex);
    }
}

void reader() {
    while (1){
        _____;
        _____;
        _____;
        if (read_count == 1)
            wait(rw_mutex);
        _____;
        _____;
        ...
        /* reading is performed */
        ...
        _____;
        _____;
        if (read_count == 0)
            signal(rw_mutex);
        _____;
    }
}
```

Answer for 2):