

CS3103 Operating Systems
2018-2019 Semester B
Midterm Paper B Questions

Part I Multiple choice questions

- 1) The requirement that when one process is in a critical section that access shared resources, no other process may be in a critical section that accesses any of those shared resources is ____
 - a) starvation
 - b) deadlock
 - c) mutual exclusion
 - d) atomic operation

- 2) The principal advantage of the ____ approach is that the transfer of control from one thread to another within the same process does not require a mode switch to the kernel.
 - a) KLT
 - b) LWP
 - c) VAX
 - d) ULT

- 3) A ____ is an integer value used for signaling among processes.
 - a) semaphore
 - b) message
 - c) condition variable
 - d) atomic operation

- 4) When a process calls SemWait(sem), which of the following cannot happen?
 - a) a blocked process becomes ready;
 - b) the calling process blocks;
 - c) a deadlock occurs;
 - d) the calling process does not block.

- 5) In a uniprocessor system, multiprogramming increases processor efficiency by ____:
 - a) Taking advantage of time wasted by long wait interrupt handling
 - b) Disabling all interrupts except those of highest priority
 - c) Increasing processor speed
 - d) All of the above

- 6) A process is in the ____ state when it is in main memory and awaiting an event.
 - a) Blocked
 - b) Blocked/Suspend
 - c) Ready/Suspend
 - d) Ready

- 7) The ____ condition can be prevented by requiring that a process requests all of its required resources at one time and blocking the process until all requests can be granted simultaneously.
 - a) mutual exclusion
 - b) hold and wait
 - c) circular wait
 - d) no preemption
- 8) The ____ condition can be prevented by defining a linear ordering of resource types.
 - a) hold and wait
 - b) no preemption
 - c) mutual exclusion
 - d) circular wait
- 9) Synchronization is achieved in message passing by ____
 - a) not allowing a process to call receive until another process has called send;
 - b) allowing a process to call receive only after another process has called send;
 - c) making a process that calls receive to wait until the message is sent;
 - d) making a sending process check whether another process is waiting to receive a message.
- 10) ____ is when the sequence of instruction is guaranteed to execute as a group, or not execute at all, having no visible effect on system state.
 - a) Critical section
 - b) Mutual exclusion
 - c) Atomic operation
 - d) Starvation

Part II Short answer questions (3 points, 15 points in total)

- 1) Suppose a thread is running in a critical section of code. Is it possible for that thread to be switched out? Justify your answer.
- 2) Name **three** events which cause a process to be placed in the ready queue, i.e., become ready state.
- 3) Which **three** mechanisms will trigger the mode switching from user mode to kernel mode?
- 4) Suppose that two processes, P1 and P2, running in a multiple processor system. Suppose that only such two processes are running in the system and each of them will run for a long time. P1 has two threads and P2 has only one thread. We have the following two situations:
 - a) all threads are user level threads;
 - b) all threads are kernel level threads.
 Will P1 potentially get more CPU time than P2 under situation (a) or (b)? Justify your answer.
- 5) Is it true that on a single processor, a multi-threaded program can always run faster than a single-threaded program? Justify your answer.

Part III Problem solving questions (35 points in total)

1. Consider the following information about resources in a system:
 - There are two classes of resource labelled R1 and R2.
 - There are two instances of each resource.
 - There are four processes labelled P1 through P4.
 - There are some resource instances already allocated to processes, as follows:
 - one instance of R1 held by P1, another held by P3
 - one instance of R2 held by P2, another held by P4
 - Some processes have requested additional resources, as follows:
 - P1 wants one instance of R2
 - P4 wants one instance of R1
 - a) Draw a resource allocation graph (5 points)
 - b) Is this system deadlocked? If so, state which processes are involved. If not, give an execution sequence that eventually ends, showing which class of resource is released by which process and acquired by which process at each step. (4 points)
2. A system has 4 types of resources, A, B, C, and D respectively. Resource A has 4 units. Resource B has 10 units. Resource C has 6 units. Resource D has 7 units. At a time point, there are 4 processes running in the system. Their claim matrix and allocation matrix are given as follows. (10 points in total)

	A	B	C	D		A	B	C	D	A	B	C	D
P0	1	3	1	2	P0	0	2	0	2	4	10	6	7
P1	1	5	6	3	P1	1	3	0	0	Resource Vector			
P2	2	2	5	6	P2	2	0	3	4				
P3	0	6	3	1	P3	0	4	2	0				
Claim matrix					Allocation matrix								

Answer the following questions using the banker's algorithm:

- a) Is the system in a safe state? If so, specify a sequence that all processes can finish. If not, show the processes which cannot finish. (6 points)
- b) If a request from process P1 arrives for (0,1,1,0), can the request be granted immediately? Explain your answer. (4 points)

3. Consider the following program (pseudocode) running in a uniprocessor system and the two processes, P1 and P2, are running concurrently. (8 points)

```
int x;
void P(int i)
{
1      x = 9;
2      x = x - 1;
3      x = x + 1;
4      if (x != 9)
5          printf("x is %d", x);
}
void main()
{
    parbegin(P(1),P(2));
}
```

Use the above program to illustrate the quote: "In a race condition, the loser wins." by showing a sequence of interleaved **statements** such that "x is 9" is printed. For each statement, specify the process (P1 or P2) that executes the statement, statement line number and value of x after the statement is executed. The execution of the first few statements is given below.

Process	Statement number	x
P1	1	9
P2	1	9
P1	2	8
P1	3	9
P2	2	8

4. Consider the following solution to the *Bounded-Buffer Producer/Consumer Problem* using *message passing*.

```
const int
capacity = /* buffer capacity */
null = /* empty message */
int i;
void producer()
{
    message pmsg;
    while(true) {
        receive(mayproduce, pmsg)
        pmsg = produce();
        send(mayconsume, pmsg)
    }
}
void consumer() {
    while(true) {
        receive(mayconsume, cmsg);
        consume(cmsg);
        send(mayproduce, null);
    }
}
void main()
{
    create_mailbox (mayproduce);
    create_mailbox (mayconsume);
    for (int i = 1; i <= capacity; i++) send (mayproduce, null);
    parbegin (producer, consumer);
}
```

Initially, the buffer is empty and no process is yet to start. (8 points)

- (a) What will happen when a consumer wants to consume data? (4 points)
- (b) Following the first consumer, what will happen when a producer wants to produce data? (4 points)