Practice Exam W24 - Results

Attempt 1 of Unlimited

Written Apr 19, 2024 10:14 PM - Apr 19, 2024 10:14 PM

Attempt Score **0 %**

Overall Grade (Highest Attempt) **0 %**

Question 1 — 0 / 1 point

The execution mode of overlapping: (select all that apply)

- ➡ ✖ ☐ can occur on a multiprocessor system
- ✔ ☐ can occur on a single-processor system
- ➡ ✖ ☐ is an example of concurrent processing
- ✔ ☐ is the same as interleaving

Question 2 — 0 / 1 point

When processes use and update shared data without reference to other processes, but know other processes may access the same data, and cooperate to ensure the data is managed properly, they exhibit this relationship:

- ◯ competition
- ➡ ◯ cooperation by sharing
- ◯ cooperation by communication
- ◯ cooperation by contention

Question 3

0 / 1 point

Consider function P1 of Peterson's algorithm for mutual exclusion, shown below. What should replace XX?

```
void P1() {
    while (true) {
        flag[1] = true;
        turn = 0;
        while (XX) /* do nothing */;
        /* critical section */
        flag[1] = false;
        /* remainder */
}   }
```

- flag [0] && turn == 0
- flag [0] && turn == 1
- flag [1] && turn == 0
- flag [1] && turn == 1

Question 4

0 / 1 point

Operations wait and signal may be performed on a semaphore. The wait operation:

- always increments the semaphore's value
- always decrements the semaphore's value
- always blocks the process
- only returns when a corresponding signal is executed

Question 5

0 / 1 point

Consider the following solution to the bounded-buffer Producer/Consumer problem. What is the purpose of semaphore e?

```
semaphore s=1, n=0, e=sizeofbuffer
void main() {
   parbegin (producer, consumer);
}
```

```
void producer( ) {
   while (true) {
        produce();
        semWait(e);
        semWait(s);
        append( );
        semSignal(s);
        semSignal(n);
} }
```

```
void consumer( ) {
   while (true) {
        semWait(n);
        semWait(s);
        take( );
        semSignal(s);
        semSignal(e);
        consume( );
} }
```

- ➡ ◯ to ensure the producer can't add data to a full buffer
- ◯ to ensure the consumer can't remove data from an empty buffer
- ◯ to ensure mutual exclusion on the buffer
- ◯ to ensure no race conditions on semaphore s

Question 6 — 0 / 1 point

Which are true about traditional Hoare monitors? (select all that apply)

- ✔ ☐ multiple processes at a time may be executing in the monitor
- ✔ ☐ it must have only global variables; no local variables are allowed
- ➡ ✖ ☐ a process enters the monitor by invoking one of its procedures
- ➡ ✖ ☐ synchronization is provided by the use of condition variables

Question 7 — 0 / 1 point

When processes use a mailbox to send/receive messages:

- ◯ they use direct addressing
- ➡ ◯ they use indirect addressing
- ◯ they perform a rendezvous
- ◯ the mailbox must be owned by the OS

Question 8 — 0 / 1 point

Which is true of the following solution to the Readers/Writers problem. (select all that apply)

```
int readcount;
semaphore x=1, wsem=1;

void main () {
  readcount=0;
  parbegin (reader, writer);
}
```

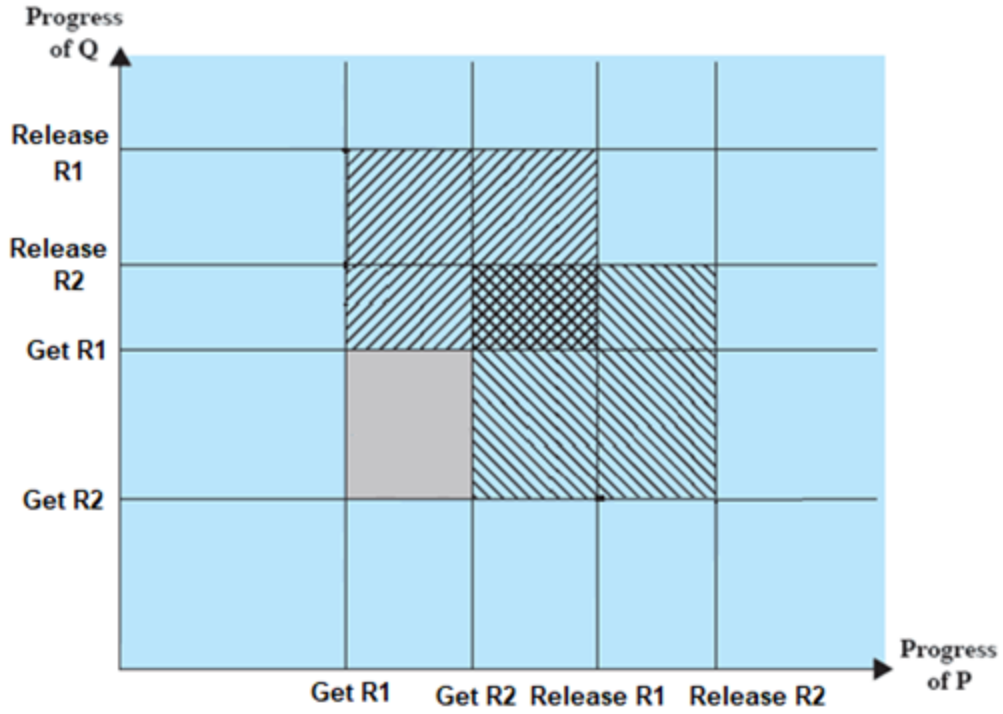
```
void writer {
  while (true) {
          semWait (wsem);
          WRITEUNIT();
          semSignal (wsem);
} }
```

```
void reader {
  while (true) {
          semWait (x);
          readcount++;
          if (readcount==1) semWait (wsem);
          semSignal(x);
          READUNIT();
          semWait(x);
          readcount--;
          if (readcount==0) semSignal (wsem);
          semSignal (x);
} }
```

- ➡ ✖ ☐ readers have priority
- ✔ ☐ writers have priority
- ✔ ☐ multiple writers may perform WRITEUNIT at the same time
- ➡ ✖ ☐ multiple readers may perform READUNIT at the same time

Question 9 **0 / 1 point**

Consider the following Joint Progress Diagram for processes P and Q below. What does the area with upward slanted lines indicate?



- both P and Q require resource R1
- both P and Q require resource R2
- both P and Q require R1 and R2
- a fatal region

Question 10 **0 / 1 point**

There are 3 conditions that are necessary, but not sufficient, for a deadlock to exist. Actual deadlock comprises these 3 conditions along with a circular wait. These 3 conditions are mutual exclusion, hold-and-wait, and:

- pre-emption
- no pre-emption
- resource allocation
- a linear ordering of resources

Question 11 **0 / 1 point**

A system contains 15 units of resource R. Maximum Claims and Current Allocations for processes P1-P5 are given below (e.g., P1 currently holds 2 units of R, although its maximum claim is 3 units). Suppose P1 and P2 execute concurrently, while P5, P4 and P3 request initiation, in that order. Using the Process Initiation Denial approach, the OS will allow which processes to begin execution?

P1	P2	P3	P4	P5
3	2	5	5	5

Maximum Claim

P1	P2
2	1

Current Allocation

- ◯ none may begin
- ◯ only P5
- ➡ ◯ only P5 and P4
- ◯ P5, P4, and P3

Question 12 **0 / 1 point**

The Claim and Allocation matrices below describe the state of a system consisting of 4 processes and 3 resources. The state would be **safe** for which of these Resource vectors? (select all that apply)

	R1	R2	R3
P1	2	2	3
P2	1	1	3
P3	1	1	1
P4	1	2	5

Claim matrix C

	R1	R2	R3
P1	1	0	1
P2	1	1	1
P3	1	1	1
P4	0	0	2

Allocation matrix A

- ✔ ☐

R1	R2	R3
3	2	5

✓ ☐

R1	R2	R3
4	2	5

✓ ☐

R1	R2	R3
3	3	5

➡ ✗ ☐

R1	R2	R3
8	7	10

Question 13 **0 / 1 point**

A major advantage of Deadlock Avoidance algorithms are that they:

➡ ◯ are less restrictive than deadlock prevention

◯ although they do rollbacks, there are less rollbacks than for deadlock detection

◯ although they do pre-emptions, there are less pre-emptions than for deadlock detection

◯ allow deadlock, but recover from it

Question 14 **0 / 1 point**

Which are true about Linux unnamed pipes? (select all that apply)

➡ ✗ ☐ may be created from a C program

✓ ☐ may be deleted using rm from a bash shell

➡ ✗ ☐ may be created from a bash shell

✓ ☐ OS provides no mutual exclusion for them

Question 15 **0 / 1 point**

Which true about Linux kernel spinlocks? (select all that apply)

✓ ☐ typically used when wait time for acquiring a lock is expected to be very long

➡ ✗ ☐ uses busy-waiting

➡ ✗ ☐ most common technique for protecting critical sections in Linux

- ✓ ☐ one spinlock may be acquired by multiple processes simultaneously

Question 16 **0 / 1 point**

What is true of the memory management scheme of Dynamic Partitioning? (select all that apply)

- ➡ ✗ ☐ a process is allocated exactly as much memory as it requires
- ✓ ☐ it has internal fragmentation
- ➡ ✗ ☐ it has external fragmentation
- ✓ ☐ small jobs will not utilize partition space efficiently

Question 17 **0 / 1 point**

Consider simple paging, with no virtual memory, as outlined in Chapter 7 of our text. Assume 16-bit addresses, and page size of 4K = 4096 = 2^{12} bytes. A process can consist of a maximum of X pages of 4K bytes. What is X?

- ◯ 4
- ◯ 12
- ◯ 15
- ➡ ◯ 16

Question 18 **0 / 1 point**

Which are true when virtual memory is used? (select all that apply)

- ➡ ✗ ☐ a process can be larger than all of main memory
- ➡ ✗ ☐ program-generated addresses are translated automatically to corresponding machine addresses
- ✓ ☐ the size of virtual storage is limited by the number of frames of main memory
- ➡ ✗ ☐ the size of virtual storage is proportional to the size of disk

Question 19 **0 / 1 point**

For virtual memory to be practical and effective, 2 ingredients are needed: (1) the OS must include software for managing the movement of pages/segments between secondary memory and main memory, and (2) hardware support for _____.

- ➡ ◯ paging and/or segmentation
- ◯ dynamic partitioning
- ◯ fixed partitioning with variable size partitions
- ◯ semaphores

Question 20 **0 / 1 point**

The page placement policy: (select all that apply)

- ➡ ✖ ☐ determines where a process is to reside in real memory
- ✔ ☐ determines where a process is to reside in secondary memory
- ✔ ☐ is relevant in pure paging systems
- ➡ ✖ ☐ is relevant for NUMA systems

Question 21 **0 / 1 point**

Suppose a process does the following sequence of page references and is allocated a fixed 3 frames. How many page faults occur using the LRU Policy after the frame allocation is initially filled? 7,8,7,6,9,8,7

- ◯ 0
- ◯ 1
- ◯ 2
- ➡ ◯ 3

Question 22 **0 / 1 point**

With this type of allocation policy, whenever a page fault occurs in the execution of a process, one of the pages of that process must be replaced by the needed page.

- ◯ fixed allocation
- ◯ variable allocation
- ◯ resident set allocation
- ◯ dynamic allocation

Question 23 **0 / 1 point**

A cleaning policy: (select all that apply)

- ☐ is concerned with when a modified page should be written out to secondary memory
- ☐ is the opposite of a fetch policy
- ☑ ☐ is concerned with where on secondary memory a modified page should be written
- ☑ ☐ is concerned with where in main memory a modified page should be placed

Question 24 **0 / 1 point**

Lab06 used POSIX semaphores. What was true about them? (select all that apply)

- ☐ named semaphores persist, even when no processes are using them
- ☑ ☐ an unnamed semaphore cannot be shared among (pthread) threads of a process
- ☐ the initial value of a named semaphore can be set by function sem_open
- ☐ they require library <semaphore.h>

Question 25 **0 / 1 point**

Which are true about a System V message queue, as used in kirk.c & spock.c in Lab08? (select all that apply)

- ➡ ✖ ☐ the first member of the message structure must have type long (int)
- ➡ ✖ ☐ msgsnd blocks when the message queue is full
- ✔ ☐ its permissions must always be exactly 600
- ✔ ☐ it is automatically deleted by Linux when no process is using it

Done