Exam 2, CS – 405, Fall 2019
Iyengar

Your Name: ---------Adam Stammer---------------

Please be neat. You may use both sides of the exam sheets.

What is meant by synchronization on a shared resource by concurrent processes? (10)
When processes share a given resource they must be careful so as to not use/change said resource when another process is also using or changing it. Synchronization can be implemented through a number of methods, like blocking, but the intent is to only allow one process to work with a shared resource at a time, such that there aren't any inconsistencies about the resource between the given process. This is a very important and relatively difficult and tedious task that most operating systems must handle.

Why is it difficulty to have a shared memory model between multiple processes? How does the concept of threads alleviate this issue? (10)
Since a given process has 'complete' control over the system, it can be difficult to share data between them. The two processes need to agree on how to communicate, and it frequently relies on agreed upon hardware based decisions that rarely carry over well to other systems and may conflict with other processes. The Process Control Block is on way data can be safely passed from process to process but this is a rather expensive process. Since multiple threads can belong to the same process, this can be avoided, and the process itself can implement its own methods of shared memory.

Describe/explain the Petersons method of synchronization. (10)
Two processes share a given list of boolean flags that indicate which process wants to enter their critical section. They also share an integer that indicates whose turn it is to do so. Since the two processes might attempt to enter their critical sections at the same time, the turn integer will be written by one process and then immediately overwritten by the other, such that it can only be one process's turn.

Simple software solutions for synchronization cause busy wait. What does busy wait mean? (5)
While a process is waiting for a resources to become available, it simply waits in a loop continuously checking for the resource to be made available. In that time, the process not only fails to acomplish any tangible computations, but it also consumes cpu time that could be used by other processes. This process is waiting, but it is busy doing so.

List and describe the 3 criteria that should be satisfied for a correct synchronization solution to resource sharing concurrent processes. (15)
Mutual Exclusion – Only one process can execute its critical section at a time
Progress – only the processes wishing to enter their critical sections can decide which process gets to do so
Bounded Waiting – when a process requests executing its critical section, there is a limit to how many other processes can execute their critical sections first. This avoid indefinite waiting and balances out the waiting for critical section execution.

Describe the Test&Set (hardware) instruction and its usage to implement mutual exclusion. (10)
TestAndSet avoids concurrent processes from using an 'old' value of a resource by first checking to make sure the resource is what is expected before changing it. If a resource is altered by process A, process B would have to check and see that before changing it, thus limiting how many processes can change that resource to just one, which is mutual exclusion.

What is a Semaphore? Describe the wait() and signal() operations on a semaphore. (10)
A semaphore is a management object that holds an integer and past initialing it, can only be used between wait() and signal(). Wait() decrements the integer by one, while signal increments it by one. This integer represents the availability of a shared resource, so during the wait, the semaphore waits for that integer to be greater than one (waits for it to be available). During the signal, the integer is incremented by one indicating additional availability for that resource.

How is busy-wait avoided in the semaphore operations? (10)
When a semaphore is found to be unavailable, rather than entering a busy wait, the processes can enter a blocked state and enter a waiting queue, such that cpu scheduler will 'wake' it up when the given semaphore becomes available. This allows other active processes to use the cpu, without having to constantly share it with processes that are busy waiting for said resource to be made available.

Show the code segment that uses semaphore operations while accessing shared data (critical section). (5)

```
semaphore.wait();

//critical section using semaphore resource

semaphore.signal();
```

What is a counting semaphore? What is a signaling semaphore? (10)
Counting semaphores are used to control access to a finite resource. To begin with the semaphore starts at the number of resources available. When a process requests its use, that number is decremented, and incremented on release. If that number is 0, we know that all the resources are being used and any requesting process must wait until that number increases.

Signaling Semaphore are only either unlocked or locked. A requesting process can only access the resource if it's unlocked, and must lock it while it is being used. This is one way to implement mutual exclusion.

List and describe the conditions necessary for Deadlock to occur? (15)
Mutual Exclusion – only one process can use a resource at a time

Hold and Wait – some process is holding onto a resource while waiting for another

No Preemption – resources are only released when a process decides to do so

Circular Wait – a list of process exist such that process A is waiting for a resource currently held by process B, process B is waiting for a resource held by process ...N, and process N is waiting for a resource held by process A. This circle could be 2 or more processes.

How does resource numbering method avoid deadlock? (10)
Numbering resources is one way to give an order. By requiring all processes to request resources in order, and release before requesting the next, the circular wait will not occur. I like to picture it as, at most, an unclosed circle. Resources holding can only be spread around the potential circle, with at least one 'hole' not waiting, thus avoiding a circular wait. There is an overhead cost to this because a process must request all of the resources in order even if it only wants one. This can also result in waiting for resource A when you only want resource B and resource B is currently free, thus furthering the potential inefficiency.

List/Name three classical synchronization problems. (5)
Bounded-Buffer Problem
Readers-Writers Problem
Dining-Philosophers Problem

Why is deadlock detection costly? (10)
There is a large amount of overhead because to check for deadlock one must look for a cycle/loop within the process resource blocking. This means you must loop through each resource and with each one, again loop through the possible cycle of

waiting. Since this means every resource could be a part of a loop this results in an O(n^2) complexity. It’s generally easier to avoid deadlocks in the first place.

Using optimistic concurrency control, one may end up with a deadlocked system. Propose a method of recovery from a deadlocked system and articulate the pros and cons of your proposal. (10)
Take a random process that is a part of the deadlock. End its wait for any resources and release all of its held resources.

Pros: - Ideally, this will free up at lease one resource, leading to the freeing up of another process, which can continue to free up other processes and resources.
- Low cost; most difficult part is deadlock detection
- If it fails it will try again with a different random process, increasing the chances of success

Cons: - The chosen process now must recover itself and re-request resources. This is not only tedious to design and code, but risks falling right back into a deadlock.
- Since re-deadlock is possible, this method does not guarantee recovery