Operating Systems 3120 - Week 8 - Locks & Semaphores

1. **Ensuring that a process releases all of its resources if it is not able to acquire the resources needed to execute is a method of ensuring "_____" to prevent "_____"?**	Preemption to prevent deadlocks
2. **The following are four methods for handling what problem?** **- Do nothing** **- Prevention** **- Avoidance** **- Detection & recovery**	Deadlocks
3. **The following are some advantages to what?** **- Dont have to increment/decrement counters** **- Dont have to check for boundary cases** **- Single call required** **- Same abstraction for different synchronization problems**	- Semaphores
4. **How can we mitigate the issues associated with semaphores?**	- Implementing a safe queue to queue processes waiting to acquire a semaphore - must ensure the semaphore queue is mutexed
5. **How could we prevent a deadlock from occurring?**	By removing the possibility of one of the following from happening... - Mutual exclusion - Hold and Wait - No preemption - Circular wait
6. **How does a lock address the concurrency problem?**	- Prevents more than one thread from executing a certain bit of code (eg: code to modify a variable)
7. **How does Dekker/Petersen's Solution function with regards to solving the critical section problem?**	- Utilizes a ready flag and a turn indicator variable - Whichever process reaches the critical section first waits for the other to go by setting the flag for the other
8. **How does Sleep/Wake or Suspend/Resume differ from a busy waiting scenario?**	It differs in that the onus is put on the thread in the critical section to notify the waiting (and suspended) thread when it has completed its work and the critical section can now be entered.
9. **How do monitors differ from locks or semaphores?**	- Monitors are language specific - Monitors are implemented with locks, semaphores, or other mechanisms
10. **How might direct avoidance of deadlocks be implemented with regards to the states?**	- User specifies max resource, request/release order when process runs - System checks if it can run safely before executing
11. **In the Dekker/Petersen's solution, which of the following has it addressed?** **- Mutual exclusion** **- Scheduler independent** **- Allows Progress** **- Starvation free**	- Mutual exclusion: Yes - Scheduler independent: Yes - Allows Progress: Yes - Starvation free: Yes
12. **In the detection and recovery methodology, how would a deadlock be detected?**	- Use method such as RAG or Bankers algorithm to repeatedly check if cycles have occurred or there is an over commitment of resources
13. **In the proposed solution of lock variables. Which of the following does it address.** **- Mutual exclusion** **- Scheduler independent** **- Allows Progress** **- Starvation free**	- Mutual exclusion: No - Scheduler independent: Yes - Allows progress: Yes - Starvation Free: No
14. **In the Test and Set solution, which of the following has it addressed?** **- Mutual exclusion** **- Scheduler independent** **- Allows Progress** **- Starvation free**	- Mutual exclusion: Yes - Scheduler independent: Yes - Allows Progress: Yes - Starvation free: No
15. **In what method of handling deadlocks would we allow a deadlock to occur and then carry out some method of handling the deadlock?**	Detection and Recovery

16. **A "_____" is a lock mechanism that requires a thread to spin in a loop testing a condition of some sort(waiting for another thread to unlock)**	Spin Lock
17. **Is preventing mutual exclusion a valid option for attempting to prevent deadlocks?**	No, mutual exclusion is necessary to have multiple threads navigate a non-sharable resource
18. **Preventing a process / thread from holding more than one resource at a time would be one way to prevent deadlocks, are there any issues with this method?**	Yes, It requires inefficient implementation, and also does not account scenarios where more than one resource is required by a process
19. **To prevent hold and wait, we may try and allocate all the resources a process needs at the start of execution. What are some possible issues with this method?**	- Have to know up front what resources are necessary - Starvation is possible - Low utilization of resources , lots of holding
20. **What are 3 abstractions used for mutual exclusion?**	- Locks (mutex) - Semaphores - Monitors
21. **What are some disadvantages to programming to directly avoid deadlocks?**	- Low utilization - Poor user experience
22. **What are some implementation issues with the Suspend/Resume Cycle?**	- If many threads are waiting / suspended, they must be ordered in some way - If they are to be ordered in some way (eg: in a queue) we now have to manage who gets to edit the queue
23. **What are some issues associated with locks?**	- Many implementations do not allow threads to acquire its own lock - Implementations require that a thread that owns the lock, release it - Only has two states - Can have missed wake ups or spurious wake ups
24. **What are some of the issues associated with ensuring resource release by processes that were not able to acquire the necessary resources?**	- Some resources should not be pre-empted - Starvation is possible - Under utilization of resources
25. **What are some of the problems with Dekker/Petersen's solution?**	- Only works for 2 processes - Requires busy waiting - Assumption: writes and reads are atomic
26. **What are some of the problems with lock variables as a solution?**	- Either does not work at all or depends on a scheduler - Starvation is possible - Require busy waiting
27. **What are some possible issues with implementing semaphores?**	- The internal lock on the semaphore can still cause a spin lock when a thread is trying to acquire the semaphore - Deadlocks are still possible
28. **What are some possible problems with spin locks?**	- Starvation is possible - Busy waiting
29. **What are the 4 instructions in Pthreads mutex API?**	pthread_mutex_init() pthread_mutex_destroy() pthread_mutex_lock() pthread_mutex_unlock()
30. **What are the 5 main goals of effective thread / process synchronization?**	- Avoid destructive inference - Avoid deadlock - Avoid starvation - Avoid scheduler reliance - Maximize concurrency
31. **What are the four conditions necessary to create a deadlock?**	- Mutual exclusion is possible (eg: 1 thread can lock a resource) - Hold and wait is possible (eg: thread can be holding 1 resource while waiting for another) - No preemption(eg: processes cannot be forced to release a resource) - Circular wait (eg: processes holding resources from one another while waiting on one another)
32. **What are the pros and cons of ensuring resource acquisition order ?**	- Pro: Easy to implement - Con: Code must be written to maintain the order
33. **What are the three main options for handling a deadlock once it has been detected?**	- Notify the user - Terminate the process or process tree associated - Preempt all associated resources (or Do nothing)

34. **What are the three types of states that one must be aware of in order to avoid deadlocks from occurring?**	- Safe state: requesting a resource will not risk a deadlock - Unsafe state: deadlock is possible - Deadlock: deadlock has occured
35. **What are the two atomic operations of a semaphore?**	- Acquire - Release
36. **What are the two fundamental operations for locks?**	Lock and Unlock ! Duh
37. **What are the two types of semaphores?**	- binary: can be at most 1 - counting: Initialized to a positive number, can be decremented until 0 where it locks. Lets a specified number of processes access a resource
38. **What atomic operation(s) are supported on many CPUs ?**	test and set. If the cpu is busy, it cannot execute its operation, otherwise it does.
39. **What does it mean for a read or write to be performed atomically?**	An atomic operation, is one that is un-interruptible.
40. **What information is contained in a semaphore?**	- # of pending wakeups - # of sleeping processes
41. **What is a lock ?**	- Mutual exclusion mechanism to protect critical sections
42. **What is a more general algorithm for determining deadlock safety?**	The bankers algorithm
43. **What is an issue associated with attempting to detect if a deadlock has occurred?**	- Algorithms to detect a deadlock can be expensive, how often should they be run?
44. **What is another option to try and prevent hold and wait that is similar to ensuring processes have all their resources at the beginning of execution? This methods suffers from the same downfalls as normal preemption**	- Processes may not own any resources when requesting resources eg: have to request all resources needed in batches, no holding
45. **What is a RAG and what is its purpose?**	- Resource Allocation Graph - Used to reason about deadlocks
46. **What is a solution to the busy waiting problem?**	Suspend / Resume Cycle
47. **What is a solution to the suspend resume cycle?**	- Using spin locks - Let the OS handle critical sections and ignore it
48. **What is one method to try and prevent circular wait separate from direct preemption?**	- Ensure resources are only able to be required in a predetermined order (R1, R2, etc)
49. **What is the most basic algorithm for determining deadlock safety? What are its downfalls?**	- Resource Allocation Graph (RAG) - Does not work when resources have multiple instances
50. **What will likely occur if two or more threads try to acquire semaphores incorrectly?**	A Deadlock