OS Concurrency

1. **Describe how a fatal race condition could occur in a simple solution to the consumer producer problem**	1) Count starts at one 2) consumer starts loop, decrements to 0 3) consumer restarts loop, count=0, sleep 4) producer runs, count +=1, count = 1, wake up consumer 5) preemption calls consumer to sleep
2. **Describe mutual exclusion using critical regions?**	When a process enters a critical region of processing, if any other process attempts to enter critical region, it will wait until the original is finished.
3. **Describe recovering through killing processes.**	Deadlock recovery where we kill a process in the cycle (one or more) until the deadlock is resolved. However, the victim can be a process not in the cycle.
4. **Describe recovery through preemption.**	Deadlock recovery where a resource is temporarily taken away from its owner and given to another process. It is highly dependent on the nature of the resource (is it preemptable?) and is frequently impossible.
5. **Describe recovery through rollback**	Deadlock recovery where a process is checkpointed periodically by having its state written to a file. In case of deadlock, a process owning a needed resource is rolled back to that state before it owns that resource.
6. **Describe the message passing solution to the producer-consumer problem.**	The producer and consumer send and receive messages from each-other. Every time the producer receives an empty message, it builds a message with the item and sends it to the consumer. We can set the number of empty items with an initial for loop of empty messages in the consumer.
7. **Describe the semaphore data structure.**	int value; - counter List queue; - list of procs sleeping in semaphore void P(); - down(), wait() void V(); - up(), signal()
8. **Give some examples of a consumable resource.**	Interrupts, signals, messages, and information. I/O buffers
9. **Give some examples of a reusable resource.**	processors, I/O channels, memory, devices, data structures
10. **How can we achieve mutual exclusion with busy waiting?**	- Disable interrupt (user too much privilege) - Lock variable
11. **How can we attack the circular wait deadlock condition?**	1) Only allow a process to access a single resource at a given time. 2) Globally number all resources and process resource requests in numerical order, so the resource allocation graph will never have cycles.
12. **How can we attack the hold and wait deadlock condition?**	Prevent processes holding resources from waiting for more, they must request all resources ahead of time. OR require a process requesting a resource to first temporarily resource all their resources and get everything at once.
13. **How can we attack the mutual exclusion problem?**	Spooling, which can be done for some resources (print) but not all.
14. **How can we attack the no preemption deadlock condition?**	Virtualize some resources that are capable of being virtualized (printer, not database)
15. **How can we deal with deadlocks?**	1) ignore them (ostrich algorithm) 2) let them occur, detect and fix 3) dynamic avoidance by careful allocation 4) prevention by structural negation on 1 deadlock condition
16. **How can we detect a deadlock for one resource of each type?**	Construct a resource graph and do a depth first search, logging each node encountered. If a node is a dead end, remove it. Then, if a nod is encountered twice, we have deadlock.
17. **How does a semaphore avoid the race condition?**	P() and V() are atomic, because the first lines of both P() and V() can disable interrupts, while the last lines of P() and V() re-enable these interrupts.

18. **How does the banker's algorithm work?**	Algorithm checks if it has enough resources to satisfy some customers. If so, it assumes process closest to limit is done, and the resources are back. Then it starts again. If all resources can eventually be repaid, the state is safe.
19. **How is deadlock avoidance achieved?**	Resources are requested one at a time, and are granted only if it is safe to do so.
20. **How is mutual exclusion achieved in a monitor?**	The compiler does it - no need for rogrammer to consider. This makes the monitor a programming language construct
21. **How many threads are allowed inside of a monitor?**	One
22. **Is the bankers algorithm useful?**	Theoretically nice, but it is practically useless! Processes needs are rarely known in advnace, number of processes aren't fix, and resources can suddenly vanish.
23. **What are a few options for the timing of deadlock checks?**	- Every time a resource request is made - Every k minutes - When CPU utilization has dropped below a threshold
24. **What are race conditions?**	When the results of two or more operations depends on exact order of processes running
25. **What are reusable resources?**	Ones that can be safely used by one one process at a time and is not depleted by that use.
26. **What are some ways of recovering from a deadlock?**	- recovery through preemption - recovery through rollback - recovery through killing process
27. **What are the conditions for resource deadlocks?**	1) each resource is assigned to one process or is available 2) processes currently holding resources can request new ones 3) resources previously given to a process cannot be forcibly taken away 4) There must be a circular chain of two or more processes waiting for resources held by the next
28. **What are the four conditions to prevent errors in critical regions?**	1) no simultaneous process access 2) no assumptions about speed/# of CPUs 3) no process running outside critical region may block another process 4) no process must wait forever to enter critical region
29. **What are the issues with message passing?**	- must use acknowledgements to guard against lost messages - must use authentication to guard against imposters - must use a buffer of messages called a mailbox to know when to block (send to full or receive from empty) - with bufferless messages, must rendezvous (block in wait)
30. **What are the issues with TSL and Petersons solutions?**	They are correct, but they have busy-waiting and it wastes CPU time. They also have the priority inversion problem.
31. **What are the potential solutions to the producer-consumer problem?**	- Semaphores - Monitors - Message Passing
32. **What are the two kinds of semaphores?**	Counting semaphores and Mutex semaphores
33. **What is a barrier?**	A barrier is a solution to potential deadlocks by halting processes at a certain point (the barrier) until all of the processes arrive and then all are allowed to continue.
34. **What is a consumable resource?**	One that can be produced and consumed
35. **What is a counting semaphore?**	Semaphore used for synchronization problems where the value counter is initialized to larger than 1
36. **What is a deadlock?**	Process(es) waiting on events (resources) that will never happen, which can be just one process or system wide
37. **What is a monitor?**	It is a higher level synchronization primitive written as a solution to difficult to use semaphores.
38. **What is a mutex semaphore?**	A binary semaphore used for mutual exclusion problems where the value is initialized to 1.
39. **What is an actual deadlock?**	A deadlock of locks between threads of execution where each is waiting for another to continue.

40. **What is a nonpreemptable resource?**	One which cannot be taken away from the process without causing the computation to fail.
41. **What is an unsafe state?**	It is NOT a deadlock. No such guarantee may be given that if every resource were requested at the same time, there would exist a way for all to run to completion.
42. **What is a potential deadlock?**	The potential for a set of given threads to block based on how they are scheduled and their timing of lock requests.
43. **What is a preemptable resource?**	One which can be taken away from a process with no ill effects.
44. **What is a resource?**	Anything that must be acquired, used, and released over the course of time (hardware and software
45. **What is a safe state?**	A state is said to be safe if there is a scheduling order in which every process can run to completion even if they all request maximum resources at the same time.
46. **What is IPC?**	Interprocess communication - processes or threads working together or sharing resources.
47. **What is mutual exclusion?**	Limiting access to shared memory or resource to only one process at a time.
48. **What is peterson's solution to mutual exclusion?**	Using shared memory, a process indicates interest in entering critical region by setting a flag. A process is granted entrance to the critical region if no other processes are interested, or the turn flag has been set for that process.
49. **What is starvation?**	Processes waiting for its turn but never comes.
50. **What is the bankers algorithm?**	Anther Dijkstra algorithm (of course) which checks if granting a request leads to an unsafe state. If so, the request is denied.
51. **What is the priority inversion problem?**	It occurs when a higher priority process is never able to enter into its CR because it is locked by a lower priority process. Since P1 is low priority, it wont be scheduled and P2 is busy in busy-waiting wasting CPU cycles.
52. **What is the problem with semaphores?**	It is difficult to write semaphore code. If the down() and up calls() are not maintained like down(2), down(1), up(1), up(2), there could be a potential deadlock.
53. **What is the producer-consumer problem?**	A multi-process synchronization problem where one process writes to an array buffer and other consumes the information there. The problem is to prevent the producer from writing over the data before the consumer can read it
54. **What is the XCHG instruction?**	Similar to TSL, except instead of moving lock to register then setting it to one, the register is set to 1 then XCHG with the lock. then the register is tested and if its non 0, it was locked so loop. Then upon leaving critical region, set the lock to 0.
55. **What is TSL instruction?**	Test set lock instruction - solution to mutual exclusion problem using help from hardware. When a process wants to access memory, it grabs the lock into a register, sets the lock to 1, then tests if the lock was nonzero. If so it loops until it returns as a zero, then lets the process enter the region. When it exits, it resets the lock to 0.
56. **When does a deadlock occur?**	Among processes who need to acquire resources in order to progress.
57. **When is a set of processes deadlocked?**	When each process in the set is waiting for an event that only another process in the set can cause.
58. **When is message passing useful?**	When you are on a distributed system with no shared memory, you can pass LAN messages.
59. **Why is IPC needed?**	- Sending Information - Mitigate contentions over resources - synchronize dependencies
60. **Why use deadlock prevention?**	Because deadlock avoidance is essentially impossible. If we can ensure at least 1 deadlock condition is never satisfied, then deadlocks will be structurally impossible.