OS Chapter 5: Concurrency: Mutual Exclusion and Synchronization

1. **2 solutions to readers/writers problem**	Readers Have Priority (writers subject to starvation) Writers Have Priority
2. **3 Control Problems with Competing Processes**	1. Mutual exclusion (only one program allowed access to critical resource at once) 2. Deadlock 3. Starvation
3. **3 Levels of Process Inter-awareness**	1. Processes unaware of each other (multiprogramming of multiple independent processes) 2. Processes indirectly aware of each other (share access to some object (like an I/O buffer); exhibit cooperation in sharing the object) 3. Processes directly aware of each other (communicate with each other using process IDs, work jointly; exhibit cooperation)
4. **6 Requirements for Mutual Exclusion**	1. Mutual exclusion must be enforced 2. A process that halts in non critical section must do so without interfering with other processes. 3. It must not be possible for a process requiring access to a critical section to be delayed indefinitely; no deadlock, no starvation allowed. 4. When no process currently in critical section, any process requesting access to critical section must be permitted to enter without delay. 5. No assumptions made about number of processors or relative process speeds. 6. Process remains inside its critical section for finite time only.
5. **advantages of enforcing mutual exclusion with special machine instruction**	1. Applicable to any number of processes (either single processor or multiple processors sharing main memory) 2. Simple, easy to verify 3. Can be used to support multiple critical sections
6. **Basic problem in multiprogramming systems?**	relative speed of execution of processes cannot be predicted
7. **Basic requirement for support of concurrent processes is ability to enforce ____**	mutual exclusion
8. **Binary semaphore**	semaphore that only takes value 0 or 1
9. **busy waiting/spin waiting**	process can do nothing until it gets permission to enter its critical section but continues to execute an instruction (or set of them) that tests the appropriate variable to gain entrance
10. **Can concurrency problems occur when there is a single processor?**	Yes, an interrupt can stop instruction execution anywhere in a process.
11. **Central themes of OS design (3)**	Multiprogramming Multiprocessing Distributed processing
12. **Concurrency arises in 3 different contexts**	1. multiple applications 2. structured applications 3. operating system structure
13. **Control of competition inevitably involves the ___ because it is the ___ that allocates resources**	Operating System
14. **Difficulties with multiprocessing AND multiprogramming**	1. difficult to locate programming errors because results not deterministic or reproducible 2. difficult for OS to manage the allocation of resources optimally 3. sharing of global resources fraught with peril
15. **Dijkstra's treatise**	Two or more processes can cooperate by means of simple signals (semaphores), such that a process can be forced to stop at a specified place until it has received a specific signal
16. **disadvantages of enforcing mutual exclusion with special machine instruction**	busy waiting is employed (wasted processor time) starvation possible (selection of waiting process arbitrary) Deadlock is possible
17. **Distributed processing**	management of multiple processes executing on multiple, distributed computer systems

18. **Event flag**	memory word used as synchronization mechanism; each bit represents event, threads check bits to check status of certain events; thread blocked until all required bits set (AND) or until one is set (OR)
19. **Fundamental to all OS themes is ____**	concurrency
20. **How are processes removed from the queue of processes waiting on semaphore?**	FIFO (if FIFO, semaphore called "strong semaphore"); guarantee freedom from starvation
21. **In a multiprocessor architecture _____ does not guarantee mutual exclusion**	preventing interruption
22. **In a uniprocessor system, concurrent processes cannot have _____ execution; they can only be _____**	overlapped; interleaved
23. **Key terms related to concurrency**	atomic operation critical section deadlock livelock mutual exclusion race condition starvation
24. **Monitor**	programming language construct that encapsulates variables, access procedures and initialization code within an abstract data type
25. **Multiprocessing**	management of multiple processes within a multiprocessor
26. **Multiprogramming**	management of multiple processes within a uniprocessor system
27. **Mutex**	mutual exclusion lock; similar to binary semaphore, but process that locks the MUTEX (sets it to 0) must be the one to unlock it (sets it to 1)
28. **Processes directly aware of each other, relationship?**	cooperation by communication
29. **Processes indirectly aware of each other, relationship?**	cooperation by sharing
30. **Processes unaware of each other, relationship?**	competition
31. **Producer/Consumer problem**	one or more producers generating some type of data and placing these in a buffer; a single consumer is taking items out of buffer one at a time; system is to be constrained to prevent overlap of buffer operations. (only one agent may access buffer at any one time); ensure that producer doesn't add data if buffer full, consumer doesn't take out if empty
32. **Race condition**	when multiple processes or threads read and write data items so that the final result depends on the order of execution of instructions in the multiple processes; "loser" of the race to write determines the final value
33. **Readers/Writers Problem**	there is a data area shared among a number of processes. The data area could be a file, block of main memory, or even a bank of processor registers. There are a number of processes that only read the data area (readers) and a number that only write to the data area (writers). The conditions that must be satisfied are: 1. any number of readers may simultaneously read 2. only one writer at a time may write 3. if a writer is writing to the file, no reader may read it
34. **Semaphore**	an integer value used for signaling among processes; may be initialized to nonnegative int, decremented, or incremented; decrement may result in process blocking, increment may result in unblocking
35. **semSignal**	operation that increments semaphore value; if resulting value less than or equal to 0, then process blocked by semWait operation, if any, is unblocked
36. **semSignalB**	operation that checks if any processes are blocked on the binary semaphore (if equal 0); if so, process blocked by semWaitB operation is now unblocked; if none blocked, value of semaphore set to 1
37. **semwait**	operation that decrements semaphore value; if value becomes negative, process executing semWait is blocked; otherwise, continue execution

38. **semWait and semSignal primitives are assumed to be ___**	atomic
39. **semWaitB**	operation that checks the binary semaphore value; if 0, process executing semWaitB is blocked; if 1, value changed to 0 and process continues execution
40. **spinlock**	mutual exclusion mechanism in which a process executes in an infinite loop waiting for value of a lock variable to indicate availability
41. **To guarantee mutual exclusion, it is sufficient to ____**	prevent a process from being interrupted
42. **What design and management issues are raised by concurrency?**	1. OS must track various processes (with process control blocks) 2. OS must allocate and deallocate various resources for each active process. Multiple processes may want same resources simultaneously 3. OS must protect data and physical resources of each process against unintended interference by other processes 4. Function and output of a process must be independent of the speed at which its execution is carried out relative to speed of other concurrent processes
43. **Why do concurrency problems occur in multiprocessor systems?**	Two processes may be executing simultaneously and both trying to access the same global variable
44. **With cooperation among processes by sharing, only ____ operations must be mutually exclusive**	writing