Quizlet

Chapter 5 - Concurrency: Mutual Exclusion and Synchronization

Study online at quizlet.com/_2ma6et

1. **atomic operation** a function or action that cannot be interrupted or preempted while running

2. **basic requirement for the execution of concurrent processes** the ability to enforce mutual exclusion

3. **blocking send, blocking recieve** both the sender and receiver are blocked until the message is delivered

4. **the central themes of OS design**
- multiprogramming
- multiprocessing
- distributed processing

5. **critical section** a section of code within a process that requires access to shared resources

6. **deadlock** a situation in which two or more processes are unable to proceed because each is waiting for one of the others to do something

7. **direct addressing** send primitive includes a specific identifier of the destination process (using either explicit or implicit addressing)

8. **distributed processing** the management of multiple processes executing on multiple, distributed computer systems

9. **implementation of semaphores**
- semWait and semSignals be implemented as atomic primitives
- can be implemented in hardware or firmware
- software schemes or algorithms can be used

10. **indirect addressing** messages are sent to a shared data structure consisting of queues (mailboxes) that can temporarily hold messages

11. **interleaving** interleaving instructions from code from different processes

12. **message passing** when processes interact with one another, these requirements must be satisfied:
- synchronization to enforce mutual exclusion
- communication to exchange information

13. **message passing primitives**
- send (destination, message)
- receive (source, message)

14. **monitor** a software module consisting of one or more procedures, an initialization sequence, and local data

15. **monitor characteristics**
- local data vars are accessible only by the monitor's procedures and not by any external procedure
- process enters monitor by invoking one of its procedures
- only one process may be executing in the monitor at a time

16. **multiprocessing** the management of multiple processes within a pultiprocessor

17. **multiprogramming** the management of multiple processes within a uniprocessor system

18. **mutual exclusion** the requirement that when one process is in a critical section that accesses shared resources, no other process can be as well

19. **nonblocking send, blocking recieve** although the sender may continue on, the receiver is blocked until the requested message arrives (most useful combo)

20. **nonblocking send, nonblocking receive** neither party is required to wait

21. **OS Concerns (the OS must...)**
- be able to keep track of various processes
- allocate and de-allocate resources for each active process
- protect the data and physical resources of each process against interference by other processes
- ensure that the processes and outputs are independent of the processing speed

22. **overlapping** processes running separately

23. **producer/consumer (reader/writer) problem**
- ensure that the producer can't add data into full buffer and consumer can't remove data from an empty buffer
* **semaphore is not good for P/C problem, consumer can get ahead, leads to deadlock** *
- any number or readers may simultaneously read, one writer at a time may write to the file, if a writer is writing, no reader may read it

24. **race condition** a situation in which multiple threads or processes read and write a shared data item and the final result depends on the relative timing of their execution (think of class example)

25. **requirements for mutual exclusion**
- must be enforced
- a process that halts must do so without interfering with others
- no deadlock or starvation
- a process must not be denied access to a critical section when no other process is using it
- no assumptions
- remain in critical section for a finite time only

26. **resource competition**
concurrent processes come into conflict when competing for use of the same resource, three control problems must be faced:
- need for mutual exclusion
- deadlock
- starvation

27. **semaphore**
an integer value used for signaling among processes
- initialized to a nonnegative int value
- semWait operation decrements value (sends processes to blocked queue)
- semSignal operation increments value (unblocks processes)

28. **starvation**
a situation in which a runnable process is overlooked indefinitely by the schedule

29. **strong semaphore**
includes the removal policy of FIFO, the process that has been blocked the longest is released first from the queue

30. **synchronization**
(same concept at semaphores) achieved by the use of condition variables that are contained within the monitor and accessible only within the monitor (cWait, cSignal)

31. **three contexts in which concurrency arises**
- multiple applications
- structured applications
- operating system structure

32. **three degrees of awareness between processes**
- unaware of each other (competition)
- indirectly aware of each other (cooperation, may share I/O buffer)
- directly aware of each other (cooperation, can communicate by process ID, work together)

33. **weak semaphore**
a semaphore that does not specify the order in which processes are removed from the queue