CS3270-Chapter 6-Concurrency: Mutual Exclusion and Synchronization

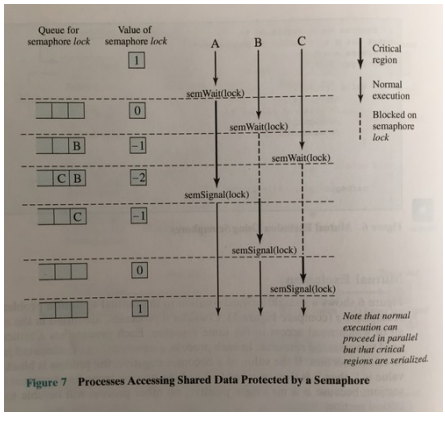
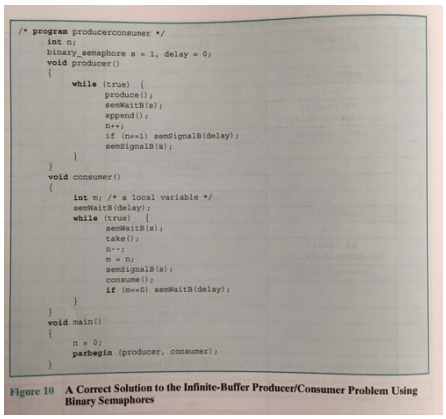
1. **atomic operation** — Sequence of one or more statements that appears to be indivisible - no other process can see an intermediate state or interrupt the operation
2. **compare and swap instruction** — -also called a "compare and exchange instruction"
-a compare is made between a memory value and a test value
-if the values are the same a swap occurs
-carried out automatically
3. **Concurrency in different contexts** — ...
4. **critical section** — Section of code in a process that accesses and/or modifies data which is shared with other processes
5. **deadlock** — A situation in which two or more processes are unable to proceed because each is waiting for the others to do something
6. **difficulties of concurrency** — -Sharing of global resources
-Difficult for the OS to manage the allocation of resources optimally
-Difficult to locate programming errors as results are not deterministic and reproducible
7. **Distributed processing** — is a phrase used to refer to a variety of computer systems that use more than one computer (or processor) to run an application. This includes parallel processing in which a single computer uses more than one CPU to execute programs.
8. **Figure 5.7 Processes Accessing Shared Data Protected by a Semaphore** — 

Figure 7 Processes Accessing Shared Data Protected by a Semaphore
9. **livelock** — A situation in which two or more processes continuously change their states in response to changes in the other processes without doing any useful work.
10. **Multiple processes** — Operating System design is concerned with the management of processes and threads:
-Multiprogramming
-Multiprocessing
-Distributed Processing
11. **Multiprocessing** — the running of two or more programs or sequences of instructions simultaneously by a computer with more than one central processor
12. **Multiprogramming** — increases CPU utilization by organizing jobs (code and data) so that the cpu always has one to execute
13. **mutual exclusion** — the requirement that when one process is in a critical section that accesses shared resources, no other process may be in a critical section that accesses any of those shared resources.
14. **Principles of concurrency** — Interleaving and overlapping:
-can be viewed as examples of concurrent processing
-both present the same problems

Uniprocessor: the relative speed of execution of processes cannot be predicted
-depends on activities of other processes
-the way the OS handles interrupts
-scheduling policies of the OS
15. **race condition** — A situation in which multiple threads or processes read and write a shared data item and the result depends on the relative timing of their execution. The last to finish will be the result
16. **Semaphore** — A variable that has an integer value upon which only three operations are defined.
1) May be initialized to a nonnegative integer value
2) The semWait operation decrements the value
3) The semSignal operation increments the value

17.	**Semaphore implementation**	 Review class exercise
18.	**Special Machine Instruction: Advantages**	-Applicable to any number of processes on either a single processor or multiple processors sharing main memory -Simple and easy to verify -It can be used to support multiple critical sections; each critical section can be defined by its own variable
19.	**Special Machine Instruction: Disadvantages**	-Busy-waiting is employed, thus while a process is waiting for access to a critical section it continues to consume processor time -Starvation is possible when a process leaves a critical section and more than one process is waiting -deadlock is possible
20.	**starvation**	A situation in which a runnable process is overlooked indefinitely by the scheduler; although it is able to proceed, it is never chosen
21.	**strong semaphore**	the process that has been blocked the longest is released form the queue first (FIFO)
22.	**weak semaphore**	the order which processes are removed from the queue is not specified