OS Chapter 5 - Concurrency: Mutual Exclusion & Synchronization

1. **Addressing - Direct**	requires destination requires sources also implicit
2. **Addressing - Indirect**	put in queue for receivers to check "port"
3. **atomic operation**	not breakable into smaller parts
4. **Can relative speed of execution of processes be predicted on a uniprocessor?**	no
5. **common concurrency mechanisms**	semaphore mutex monitor event flags messages spinlocks
6. **critical section**	section of code requiring shared resources and will fail if run when another process is in corresponding section of code
7. **deadlock**	two or more processes are each waiting for the other to do something
8. **Difficulties of concurrency**	sharing of global resources mgmt of optimal resource allocation results are not deterministic & reproducible, so difficult to locate programming errors
9. **livelock**	two processes continually change states cancelling each other out
10. **message passing**	synchronization & communication 3 types both sender and receiver are blocked until msg delivered - tight synch nonblocking sender, blocking receiver - most useful (e.g. service provides resource to other services) neither blocked
11. **monitors**	data structure - control variable like semaphore, but easier to control uses conditional variables
12. **mutual exclusion**	when one process is in crit sect, no other proc can be in a crit section accessing shared resources
13. **mutual exclusion - interrupt disabling**	+process will run until interrupt +disabling interrupt guarantees mutual exclusion -efficiency could be degraded -does not work in multi-proc architecture
14. **mutual exclusion - special machine instruction**	+applicable to any # of processors +easy to verify -busy-waiting employed, cpu time wasted -starvation possible -deadlock possible
15. **O.S. Concerns for concurrency**	OS must be able to keep track of various processes OS must be able to allocate and de-allocate resources for each active process OS must protect the data and physical resources of each process against interference by other processes OS must ensure the processes and outputs are independent of the processing speed
16. **Principles of Concurrency**	Interleaving overlappting
17. **producer/consumer problem**	producer places data into buffer consumer takes from buffer only one may access at a time
18. **race condition**	...
19. **Resource Competition**	- must be enforced - halts must not interfere with other processes - no deadlock or starvation - must not be denied access to critical section when nothing else is using it - no assumptions about processor speed - processor remains inside critical section for finite time
20. **semaphore operations**	1. initialization 2. decrease (semWait) 3. increase (semSignal)
21. **semaphore status levels**	sv>0 = free sv==0 = used, no waiting sv<0 = used, process waiting
22. **semSignal & semWait**	atomic primitives implemented in firmware also via software algorithm

23.	**starvation**	a process is overlooked indefinately by the scheduler even though it can run