1. Basic OS Tasks
   1. Allocate memory
   2. Manage CPU usage
2. Microkernels vs Monolithic
   1. Microkernel
      1. Bare minimum kernel
         1. Only provides basic operations
         2. Little consensus on what should be in kernel
         3. Broken in to servers
            1. Some run in kernel space some run in user space
         4. Maintenance is easier
         5. Easier to add new software
         6. Larger chunk of memory used
         7. More software required
         8. Process management can become complicated
         9. Uses message passing to communicate
         10. Used in OSX and Windows NT
   2. Monolithic
      1. Large kernel, exists in static binary file
      2. Not portable
      3. Can become too large to maintain
      4. If a driver is down it can take out the entire system or have system-wide side-effects
      5. Less software involved so it is faster
      6. All kernel services exist and are executed in the kernel address space
      7. Used in Linux and Unix
3. Process state
   1. Basic 5 state flow chart
   2. Exit normal or abnormally
      1. End()
      2. Abort()
   3. Other functions
      1. Load()
      2. Execute()
      3. Create process
      4. Terminate process
      5. Get process attributes
      6. Set process attributes
      7. Wait for time
      8. Wait for event
      9. Signal event
      10. Allocate free memory
   4. Process queue
      1. Linked list form
4. What is a thread
   1. Basic unit of CPU utilization
      1. Contains thread ID
      2. Program counter
      3. Resister set
      4. Stack
5. Benefits of POSIX threads
   1. Increase degree of multiprogramming easily
   2. Less overhead that typical fork() or createProcess()
      1. Does not make complete copy of parent
   3. Less time to terminate thread than process
   4. Context switching between threads is faster than processing
   5. Less communication overhead
6. Learn to read C
7. Producer consumer problem
   1. Producer has limited resources
   2. Consumer needs multiple producer resources
   3. Creates a cycle when multiple consumers need producer assets
   4. Can lead to deadlock
   5. Currently dealt with by looking at cycles or by using mutex locks
8. Mutex and semaphores to handle concurrency
   1. Mutex is used to lock a limited resource
   2. Semaphores can be used to count up or down to block or allow access to certain resources
      1. Resources are only available when the semaphores are in a certain range
9. Multitasking vs multiprocessing
   1. Multasking
      1. The method in which multiple tasks share common processing resources like the CPU
      2. Multitasking schedules which task can be running at a given time
      3. Also in charge of context switching (switching from one task to another)
   2. Multiprocessing
      1. A term for two or more CPUs on a single system
      2. Refers to the execution of multiple concurrent software processes in a system as opposed to a single process at any one instant
10. Monolithic vs Kernel
    1. Monolithic Categories of functions
       1. Device drivers
       2. Scheduler
       3. Memory handling
       4. Files systems
       5. Network stacks
    2. Microkernel functions
       1. Memory management
       2. Access to hardware
       3. Inter-process communications
    3. Microkernel decoupled functions
       1. Networking
          1. Easier implemented in user programs known as servers
11. Process state
    1. Running -> Ready
       1. Object currently being run ends time slice
    2. Running -> Blocked
       1. Object being run needs further information (I/O)
    3. Blocked -> Ready
       1. Object in wait queue meets condition to move back to ready queue