-
- Two general types of software: System and Application
- Most important function: process, device, memory and file management

- **System Interface Call:** Used by application programmers to request os services.
- **Process Manager:** Os component that is responsible for creating and destroying processes as well as scheduling them
- **Memory Manager:** Os component that allocated and de-allocates memory and protects against unwanted memory access
- **File Manager:** Os component that creates and deletes files/directories , reads and writes
- **Device Manager:** Os component that uses a device driver to control hardware devices and handles devices
- **Multiprogramming:** When an OS manages several process in memory and shares the CPU
- **Context Switching:** When the OS changes the process currently executing on the CPU

- **Resource Manager:** Allocation and de allocation of resources to components that need it (sharing resources across processes that need them)
- **Process:**
  - System Process: Exists to execute OS actions
  - User Process: User started process
- **Process State:**

Threads

- **Thread:** A software computational unit
  - Each process has a base thread
  - All threads in one process share code and resources

- **Thread Control Block:** Contains information for each thread
  - Contains an instruction counter and id
  - 1 to 1 app between kernel threads and the app
  - Also can be many to many

- Pool of pre allocated kernel threads are used to service app threads

- **Multithreading:** Process with multiple threads
- **Multiprogramming:** An OS that can manage multiple processes
  - Process can be interrupted for another process (context switching)

Modeling

- **Types of OS Models:**
  - Physical: Small scale physical representation
  - Graphical: Diagrams
  - Mathematical: Algorithms

- **Process Steps:**
  - Memory manager takes program instance at head of queue and allocates memory
  - Processor services program
  - Program terminated, memory is deallocated

- **Way of Measuring System Performance:**
  - Measure the actual system
  - create simulations
  - create analytical models

- **Types of Simulation Models:**
  - Deterministic: No randomness in the test
  - Stochastic: Has randomness included by having random values inputed

Multiprogramming

- Multiple processes at once
- Degree of multiprogramming: Number of processes a system supports

- If there is no memory, the job goes back into the queue
    - If the queue is full the job is lost forever

- **CPU Burst:** Short CPU requests
- **I/O Burst:** Short I/O request

Processor Scheduling

- **Long Term Scheduling:** The Os creates a new process from the job waiting in the input queue
- **Mid Term Scheduling:** The OS decides when and which process to swap in from or to memory
- **Short Term Scheduling:** The OS decides which OS to execute next, also called CPU scheduling

- **CPU Service Requests:**
    - Join the ready queue
    - Execute the CPU burst
    - Join I/O queue
    - Terminate and exit

- **Single Class System:** When a OS treats all processes in the same manner
    - “fair” because there is only one group of processes and they are all treated equally
    - Processes have a priority. Higher priority processes are executed first

- **Synchronization:** coordination of the activities of two or more processes that carry out the following activities
    - Compete for resources in a mutually exclusive manner
    - Cooperate in sequencing or ordering specific events in their individual activities

- **No Synchronization:** When two processes execute without synchronization, which causes problems

- **Mutual Exclusion:** Only one process can use the shared resources at a time. To achieve mutual exclusion:
    - Only one process is executing its critical sections at a time
    - No starvation - Processes wait a finite time interval to enter their critical sections
    - No deadlocks - Processes should not block each other out indefinitely
    - A process will execute its critical section in a finite time interval

- **Critical Sections:** Parts of the code in a process that accesses and uses a shared resource

- **Semaphores:** An abstract data type that functions as a software synchronization tool
    - Binary Semaphore: Value can be 0 or 1, used to allow mutual exclusion
    - Counting Semaphore: Can be any nonnegative integer value

- **The Bound Buffer Problem:**
    - A producer inserts into buffers, consumer takes out
    - Both go at their own speeds so there is a need to synchronize them
    - Problem: Producers can't add to a full buffer and consumers can't take from an empty buffer so you need to have the buffer be mutually exclusive
    - Solution: Have a counting semaphore to check if the buffer is full, have another to check if the buffer is empty and have a mutex for mutual exclusion

- **The Reader/Writer Problem:**
    - A writer can't access a resource if multiple readers have access so there is a possibility of starvation

- **POSIX Threads:** Each thread has
    - Thread ID
    - A start function
    - An attribute object
    - An argument of the start function

- **Monitors:** A mechanism that implements mutual exclusion

    - abstract data types implemented as a class

- **Deadlock:** When the OS is attempting to allocate resources to a set of processes requesting these resources
    - Can cause infinite waiting
    - Conditions for a deadlock:
        - Mutual exclusion:only one process can have a resource
        - Hold and wait: a process holds a resource and requests another
        - CIrcular wait: a condition that is represented by a closed path of resource allocation and requests
        - No preemption: a process cannot be interrupted and forced to release its resources
            - These are necessary but not sufficient
- **How to handle deadlocks:**
    - Prevention
    - Detection and recovery
    - Avoidance

- **Prevention:**
    - Hold and wait: Two approaches
        - A process must acquire all resources before starting
        - A process must release all resources before getting new ones
    - Circular wait: CIrcular wait: Assign a total ordering to all resource types
- **Avoidance:** CHecks the current resource allocation state to make sure it is a "safe state"
    - The system checks a process "maximum claim" which is the total number of resource instances that a process will request
    - Bankers Algorithm: Checks to see if the state is safe
        - A state is defined by the total number of resources, the number of allocated resources and the max resource claim of the process
- **Detection And Recovery:** The OS allocates resources to the process and periodically checks for deadlock
    - Early detection: CHeck after every resource allocation
    - Once detected there are a few options:
        - Terminate all deadlock processes
        - Allocate resources to the deadlocked process
        - Terminate all other processes and run the deadlock process
        - The roll back to before the deadlock

Memory Management:

- **Logical Address**: Reference to some location of a process
- **Process Address:** Set of logical addresses that a process references in its code

- **Symbolic Address:** The address used in a source program
    - ex: Variable names
- **Relative Address:** A compiler converts the symbolic address into a relative address
- **Physical Address:** The final address generated when a program is loaded and ready to execute into memory
- **Partitions:** Blocks of contiguous memory, each one allocated to an active process
- **Fixed Partitions:** Memory is divided into fixed size partitions
    -