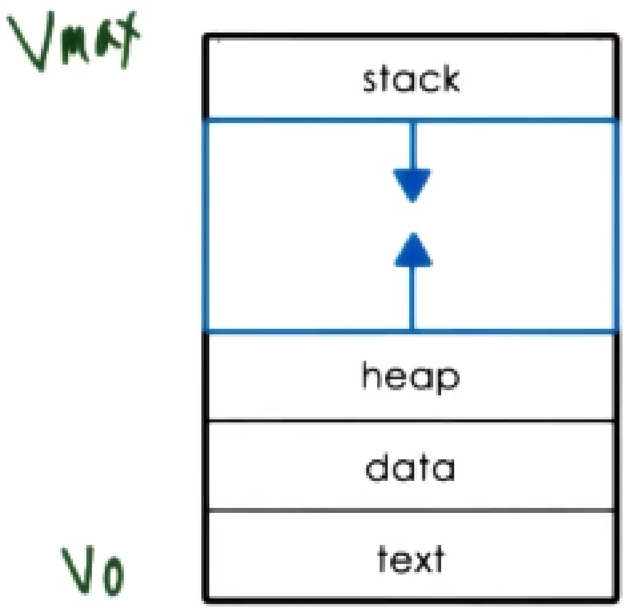
Process and Process Management

What is a process?

1. A process is an instance of an executing program ("task" or "job")
   - State of execution - Program counter, stack
   - Parts/temporary holding area - Data, register state, occupies state in memory
   - May require special hardware - I/O devices
2. OS manages hardware on behalf of applications
   - Application is a program on disk, flash memory... (static entity)
   - Process is state of a program when executing while loaded in memory (active entity)
3. Process contains all the state an application needs to run
   - Stack, heap, data, code
4. Types of state:
   - Text and data (static state when process first loads)
   - Heap (dynamically created during execution)
   - Stack (grows and shrinks, LIFO queue)



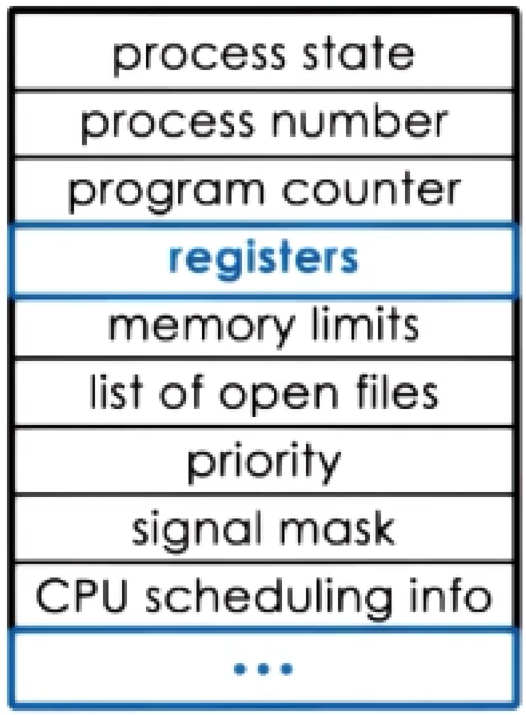
Layout of a Process

Virtual Memory

1. Address space is the "in memory" representation of a process
2. Page tables are the mapping of virtual to physical addresses (in DRAM)
3. A process only has knowledge of virtual addresses
   - The OS manages the translation of virtual to physical addresses using page tables
4. There is likely not enough physical memory for all state
   - 32 bits of addressability yields a 4 GB address space (per process)
   - OS manages the swapping of memory between physical memory and disk for each process
5. Two processes can have identical virtual address spaces (OS will map correctly)

Process Control Block (PCB)

1. Components
    - Program Counter - Keeps track of instruction currently being executed
    - CPU Registers - Holds variables local to the program
    - Stack pointer - Keeps track of stack-allocated state
    - List of open files
    - Memory information for virtual to physical address translation
    - Priority and scheduling information
2. Certain fields are updated when process state changes (memory limits)
3. Other fields update too frequently to track every change (program counter)
    - Instead of updating the PCB, the program counter is maintained in a register on the CPU
4. When processes switch, OS stores all relevant state in the PCB (context switch)



Process Control Block Structure

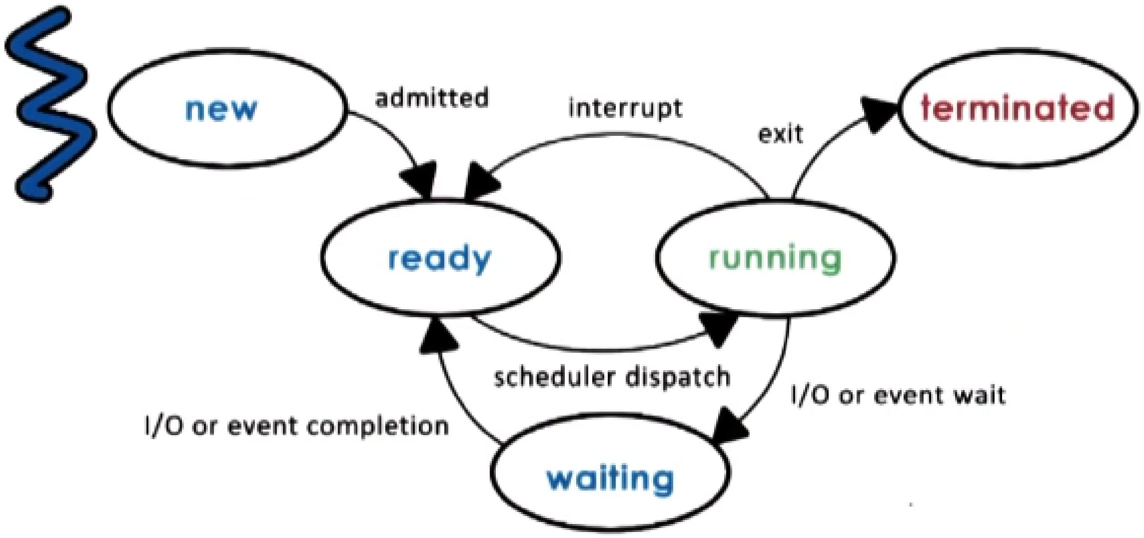
Context Switching

1. Mechanism used by the OS to switch execution from context of one process to another
2. Typically quite expensive
    - Direct costs - Number of cycles for load and store instructions
    - Indirect costs - Cache misses/ cold cache
        - Accessing cache is cycles vs 100s of cycles for memory
        - Context switching will replace cached data with new process's data

Process Lifecycle

1. New - OS performs admittance control, allocates PCB and initial resources
2. Ready - Process has resources and is ready to run, but isn't yet
3. Running - Process is executing

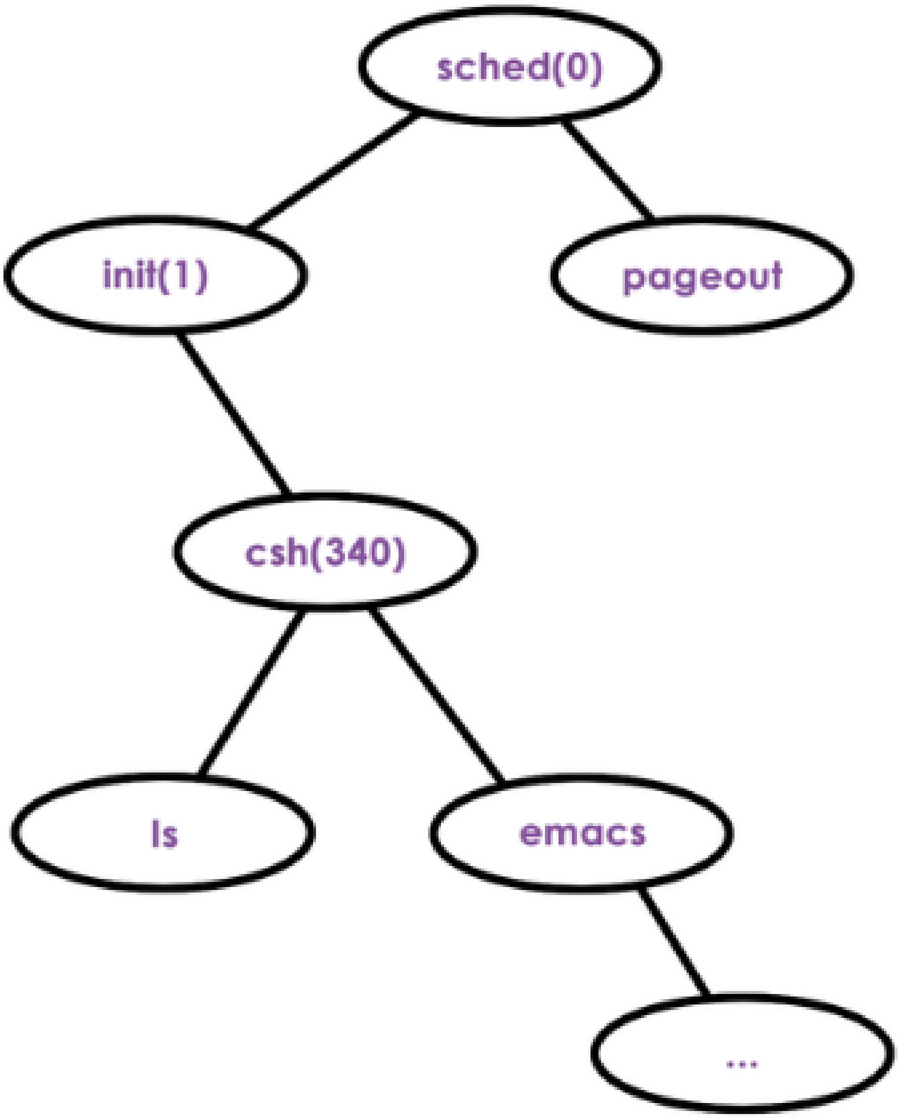
4. Terminated - Process is completed
5. Waiting - IO operation or waiting for event



Process Lifecycle Diagram

Process Creation

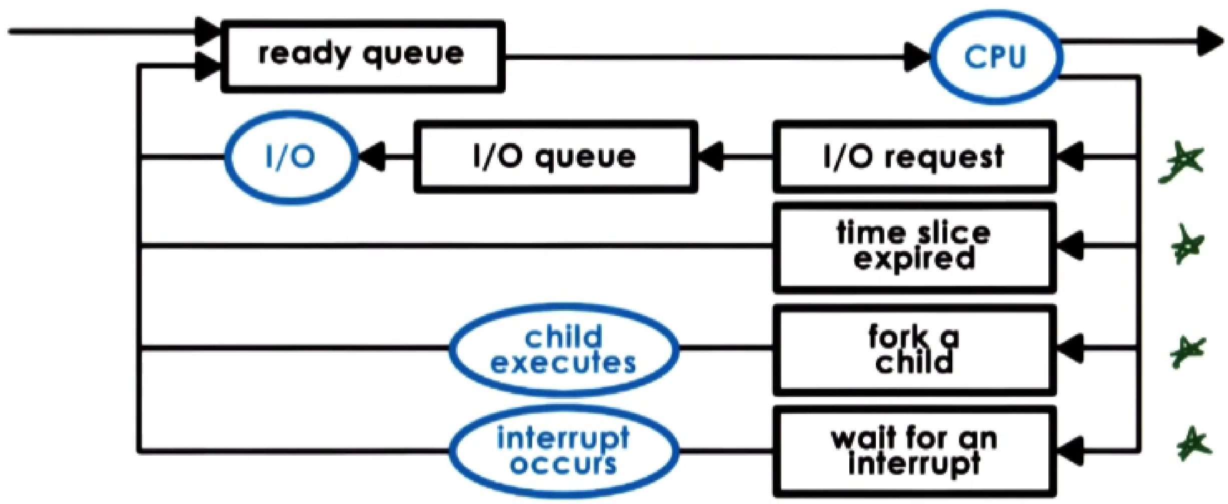
1. Fork - Copies parent PCB into new child PCB
   - Child continues execution at instruction after fork
2. Exec - Replace child image
   - Load new program and start from first instruction



Process Creation Diagram

Scheduling

1. CPU scheduler determines which one of the currently ready processes will be run next
   - Also determines how long it should run
2. Three steps to scheduling (must be done efficiently):
   - Preempt - Interrupt and save the current context
   - Schedule - Run scheduler to choose next process
   - Dispatch - Dispatch process and switch to its context
3. CPU work = Tprocess / (Tprocess + Tscheduling)
   - Timeslice - Duration of time allocated to a process (Tprocess)



Ready Queue Flowchart

Inter-Process Communication (IPC)

1. OS must provide mechanisms for processes to interact (web server/database)
2. Transfer data/information between address spaces
3. Maintain protection and isolation
4. Provide flexibility and performance
5. Message-passing IPC
   - OS provides communication channel, like shared buffer
   - Processes write (send) and read (recv) messages to/from channel
   - OS manages this channel, but introduces some overhead
6. Shared Memory IPC
   - OS establishes a shared channel and maps it into each process address space
   - Processes directly read/write from this memory
   - OS is out of the way (no overhead), but doesn't support well-defined API
   - Memory-mapping is an expensive process
     - Only makes sense when startup cost can be amortized over many messages