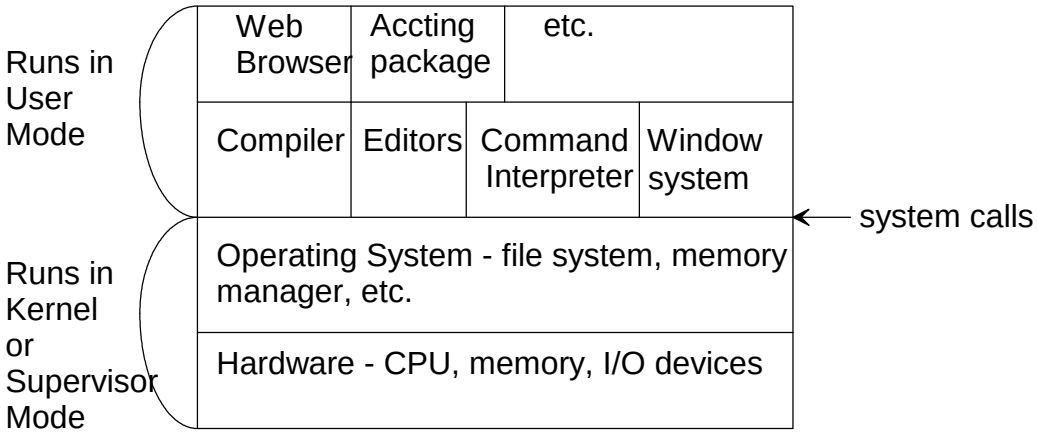
What is an operating system (OS)?

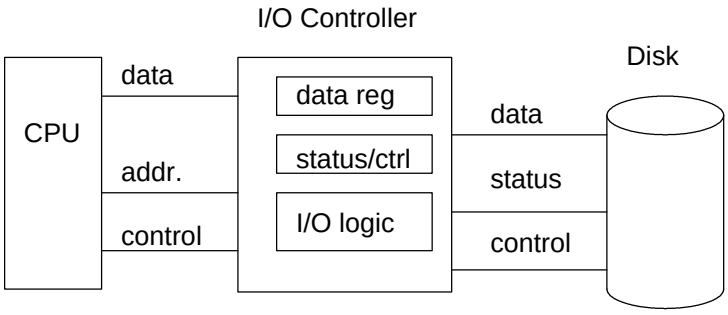
- Program that operates as the interface between the user and the hardware



Goals of OS

1. Make computer <u>convenient</u> to use by providing a virtual/extended machine that is easier to program than the underlying hardware,

   e.g., writing/reading to file on disk



   OS provides high-level system calls so programmer does not need to know details of disk

2. Use computer equipment <u>efficiently</u>
   Resources - processor(s), memory, timers, disks, network

Resources competed for by several programs

Examples:
- which programs are loaded in limited memory
- restricts access to memory used by other programs and the operating system
- which program can run on the CPU

Can view OS as resource manager. Responsible for resource allocation, tracking resources, accounting, and mediating conflicting requests

Hardware support for Operating Systems

Need protection from user programs that:

1. go into an infinite loop
2. access main memory of other programs or the OS that are currently in memory
3. access files of other users on the system

Protection Techniques

1) **CPU Timer** - OS sets a timer to expire and interrupt a user pgm before the user pgm is started. Remember that only one program (in a single CPU system) can be executing at a time so when the OS turns control over to a user program it has "lost control." Modifications to the CPU timer are privileged

2) **Dual-Mode Operation** - the CPU has two (or more) modes of operation: user mode and system(/supervisor/monitor/privileged) mode with some privileged instructions only executable in system mode. A mode-bit within the CPU's *processor-status-word* (PSW) register is used to indicate whether the CPU is executing in user or system mode. The set of all machine-language instructions are divided into:
   a) privileged instructions that can only be executed in system mode, and
   b) non-privileged instructions that can be executed in any mode of operation.

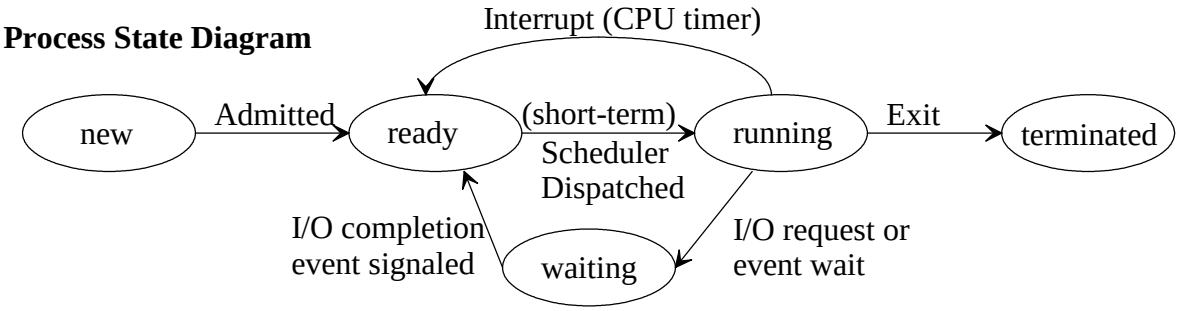
   Every time an instruction is executed by the CPU, the hardware checks to see if the instruction is privileged and whether the mode is user. Whenever this case is detected, an exception (internal interrupt) is generated that turns CPU control back over to the operating system.

3) **Restrict a user program to its allocated address space.** In a simple computer, a user program might be allocated a single continguous address space in memory. The two special purpose CPU registers: StartMemory and EndMemory can bracket the user program's address space. All memory addresses that the user program performs can be checked by hardware in the CPU to make sure that they fall between the values in these registers. If the user program tries to access memory outside the range of addresses indicated by these registers, an interrupt/exception is raised to return control back to the operating system. On more complex computers, a memory-management unit (MMU) provides a more sophicated address mapping scheme (paging, segmentation, paged segments, none). Modifications to the memory-management registers are privileged.

OS manages processes (running programs):

Many processes execute concurrently, but only one can be executing on a CPU at a time. When the CPU switches to another process, a *context switch* occurs which involves saving the complete state of the previously executing process before loading the state of the next process to execute into the CPU. Depending on the hardware, this can take 1 to 1000 microseconds (i.e., very slow in computer terms).

Process State Diagram



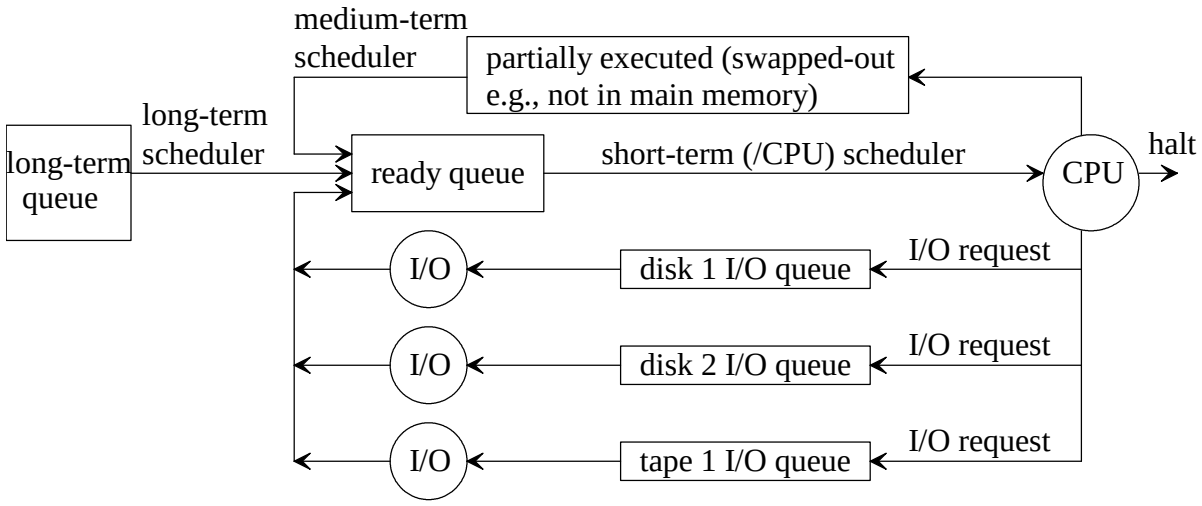
Queues are used to hold *process control blocks (PCB)* that represent processes internally to the OS.

Process Control Block

Next PCB in queue pointer
Process State
Program Counter
Registers
Memory Mgt. Info
CPU Scheduling Info.
Accounting Info.
I/O Status Info

OS maintains queues and does scheduling:

The PCB for a process moves around from queue to queue depending on its state.



I/O queues - since I/O is so slow, several programs might have outstanding requests to use an I/O device so a queue for each I/O device is necessary.

Ready (Short-term) queue - programs that are in memory and ready to execute. All they need is the CPU to run.

Medium-term queue - programs that are partially executed, but have been swapped out of memory to disk

Long-term queue - user has requested the a program be executed, but it has not yet been loaded into memory