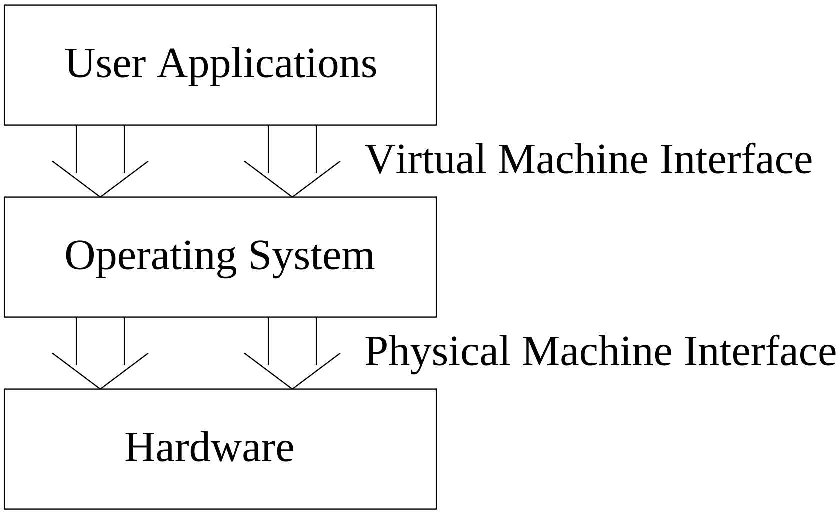
An operating system is the interface between the user and the architecture.



OS as juggler: providing the illusion of a dedicated machine with infinite memory and CPU.

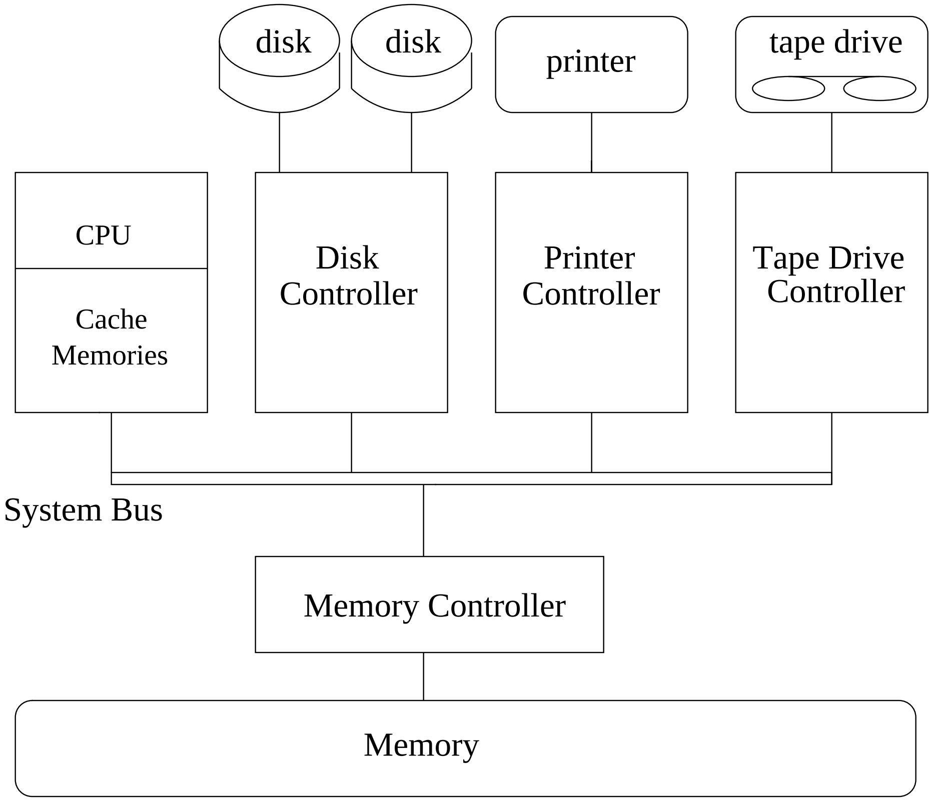
OS as government: protecting users from each other, allocating resources efficiently and fairly, and providing secure and safe communication.

OS as complex system: keeping OS design and implementation as simple as possible is the key to getting the OS to work.

Computer Architecture & Processes — Today

- Basic architecture reminder
- What the OS can do is dictated in part by the architecture.
- Course theme: architectural features can greatly simplify or complicate the OS.
- Process: unit of execution
  - How are processes represented in the OS?
  - What are possible execution states and how does the system move from one state to another?

Generic Computer Architecture



- CPU - the processor that performs the actual computation
- I/O devices - terminal, disks, video board, printer, etc.
- Memory - RAM containing data and programs used by the CPU
- System bus - the communication medium between the CPU, memory, and peripherals

From Architectural to OS to Application, and Back

Hardware	Example OS Services	User Abstraction
Processor	Process management, Scheduling, Traps, Protection, Billing Synchronization	Process
Memory	Management, Protection, Virtual memory	Address space
I/O devices	Concurrency with CPU, Interrupt handling	Terminal, Mouse, Printer, (System Calls)
File system	Management, Persistence	Files
Distributed systems	Network security Distributed file system	RPC system calls, Transparent file sharing

OS Service	Hardware Support
Protection	Kernel/User mode Protected Instructions Base and Limit Registers
Interrupts	Interrupt Vectors
System calls	Trap instructions and trap vectors
I/O	Interrupts or Memory-Mapping
Scheduling, error recovery, billing	Timer
Synchronization	Atomic instructions
Virtual memory	Translation look-aside buffers

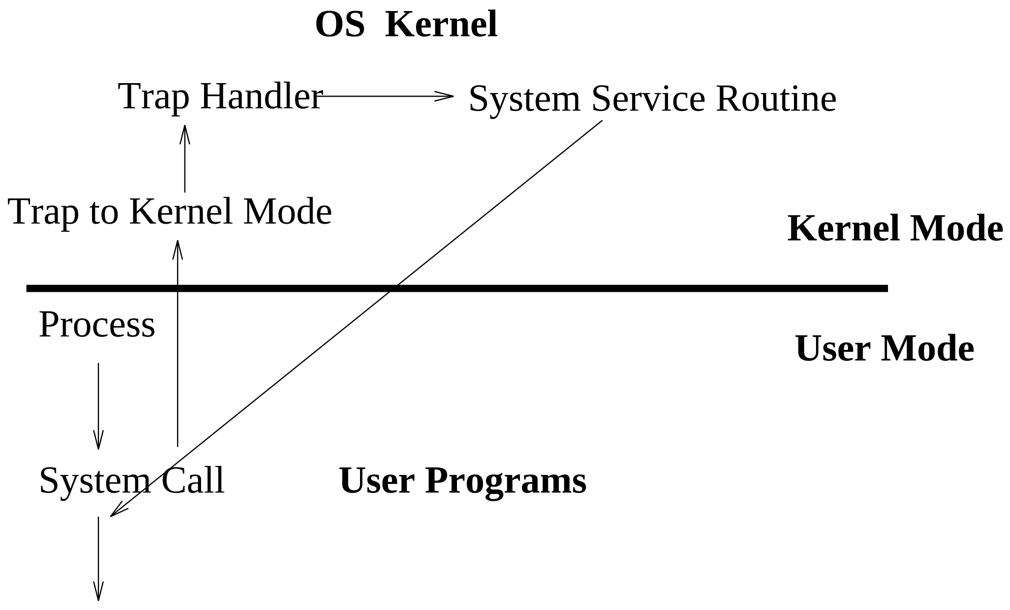
Interrupts - Moving from Kernel to User Mode

User processes may not:

- address I/O directly
- use instructions that manipulate OS memory (e.g., page tables)
- set the mode bits that determine user or kernel mode
- disable and enable interrupts
- halt the machine

but in kernel mode, the OS does all these things.

- A status bit in a protected processor register indicates the mode.
- Protected instructions can only be executed in kernel mode.
- On interrupts (e.g., time slice) or system calls


OS Kernel
Trap Handler
System Service Routine
Trap to Kernel Mode
Kernel Mode
Process
User Mode
System Call
User Programs