Why Kernel Mode?

- Services that need to be provided at kernel level
  - System calls: file open, close, read and write
  - Control the CPU so that users won’t stuck by running
    ```c
    while ( 1 ) ;
    ```
  - Protection:
    - Keep user programs from crashing OS
    - Keep user programs from crashing each other

How to Provide Kernel Mode?

- **CPU mode bit** added to computer hardware to indicate the current CPU mode: 0 (=kernel) or 1 (=user).
- When an interrupt occurs, CPU hardware switches to the kernel mode.
- Switching to user mode (from kernel mode) done by setting CPU mode bit (by an instruction).

```
Exception/Interrupt/Fault

kernel

Set user mode

user
```

*Privileged instructions* can be executed only in kernel mode.
Three Interrupt Classes

- Interrupts caused by hardware failures
  - Power outage
  - Memory parity error
- Interrupts caused by external events:
  - Reset
  - I/O devices
- Interrupts caused by executed instructions
  - Exceptions
  - System calls
Interrupts Caused by Instruction Execution

- **Exceptions**: caused by errors during instruction execution:
  - Address Error: a reference to a nonexistent or illegal memory address;
  - Reserved Instruction: An instruction with undefined opcode field or a privileged instruction in User mode;
  - Integer Overflow: An integer instruction results in a 2’s complement overflow;
  - Floating Point Error: e.g. divide by zero, overflow, and underflow;

- **Special instructions**:
  - MIPS processors: Syscall instruction executed
  - Intel processors: INT n instruction executed

---

Hardware Handling of Interrupts

- **Save the addresses of the interrupted instruction**
- **Transfer control to the appropriate interrupt service routine (software)**
- **Sets CPU to kernel mode**
- **May do some security checks here**
System Call Steps

Example:
read (fd, buffer,nbytes)

Users, Programs, Processes

- Users have accounts on the system
- Users launch programs
  - Many users may launch same program
  - One user may launch many instances of the same program
- Processes: an executing program
- Question: Real life analogy?
Windows Task Manager

Unix Processes: ps
So What Is A Process?

- It’s one executing instance of a “program”
- It’s separated from other instances
- It can start (“launch”) other processes
- It can be launched by other processes

The Process Model

- Multiprogramming of four programs
- Conceptual model of 4 independent, sequential processes
- Only one program active at any instant
Process Control Block (PCB), Why?

- PCB contains information associated with a given process:
  - Process state
  - Process identification (Pid)
  - Program counter
  - CPU registers
  - CPU scheduling information
  - Memory-management information
  - Accounting information
  - I/O status information
  - Pid of parent process

---

Process States

- As a process executes, it changes its state:
  - new: The process is being created.
  - running: Instructions are being executed.
  - waiting (blocked): The process is waiting for some event to occur.
  - ready: The process is waiting to be assigned to a CPU.
  - terminated: The process has finished execution.