CSE 378, 06wi – Lecture 13 Main Points
Support the OS

February 6, 2006

- **Boot**
  - A small non-volatile RAM holds a program run at power up
  - At its simplest, that program loads the "boot sector" from disk and branches to it
  - Boot sector program loads OS and branches to its entry
  - OS initializes and goes into its idle loop

- **Processes**
  - A running program is called a process
  - A process has state: the contents of its memory, its registers, and its PC
  - The OS can (and does) suspend execution of processes by saving all of its CPU state into a structure known as a process control block
  - The process can be restarted by restoring the state

- **IO**
  - Devices are connected to controllers. The OS communicates with (issues commands to) the controller to cause IO operations to take place.
  - Devices can be polled or direct memory access (DMA). DMA is more complicated, but is required for faster devices (like disks).
  - For DMA devices, the OS issues some kind of "start" command. The DMA device later notifies the OS that the IO has completed, when it is done. (How?)

- **Memory Protection**
  - The OS must be able to read/write all of memory (why?), but a process shouldn't be able to read or write outside of the memory allocated to it.
  - Hardware support is required. Possible approaches:
    - Bounds registers – min and max addresses accessible
    - Address translation (also called address mapping) – translate addresses issued by process (virtual addresses) into physical addresses in a way that guarantees the process can never access memory outside of its allocation

- **Other Protection**
  - A special "status register" indicates the protection level the CPU is operating at.
  - Some operations can be performed only when executing in privileged mode
    - E.g., setting the address mapping registers for a process
  - How?

        - Privileged instructions: special instructions (opcodes) used to affect privileged state. The instructions will execute without error only if the CPU is in privileged mode.
        - Memory mapping: the interface to the privileged state registers is through load/store instructions. The apparent "address" of the privileged state is outside what is accessible by any process. Thus, the memory protection mechanism also protects privileged state.

- **Interrupts / Exceptions / Traps**
    - (Interrupts are asynchronous events; exceptions are synchronous events; traps are intentional exceptions)
    - When running a user process, the CPU is in unprivileged mode. At some point, the OS must run, and the privilege bit must be turned on. Turning it on is privileged, though.
    - A "trap architecture" is used:
        - When an exception/interrupt occurs, the current PC is saved and the PC is set to a predefined address (pointing to the OS trap handler code).
        - At the same time, the privilege bit is turned on.
        - Both happen in hardware, during a single cycle
        - Result: the only way into the OS is through this mechanism, which results in execution in privileged mode at an address determined by the OS itself (not whatever was running immediately before the trap)