6.033 - Operating Systems + Virtual Memory
Lecture 3

Katrina LaCurts, lacurts@mit.edu

0. Previously
- Modularity reduces complexity
- Naming is necessary for modularity

1. Operating Systems
- Job: enforce modularity on a single machine
  - Also: multiplexing, isolation, cooperation, portability, performance, ...
- To enforce modularity on a single machine, need to:
  - protect programs' memory from each other
  - allow programs to communicate
  - allow programs to share a single CPU
- Virtualization is how we do that
- Today: virtualize memory. assume one CPU per program and that programs don't need to communicate.

2. Virtual memory
- Two components: main memory, CPU
- CPU holds instruction pointer (EIP)
- Naive method: two programs can just point to each other's memory (bad)
- Another method: force programs to only use particular blocks of memory by having them address only part of the space. Complicated.
- Virtual memory addressing: let each program address the full 32-bit space. MMU translates virtual to physical addresses.

3. Page tables
- Idea 1: Store physical addresses, use virtual addresses as an index into that table
- Problem: table is too big
- Solution: virtual address = page number + offset. MMU maps virtual page numbers to physical page numbers. Keeps offset the same.
- Page table entries contain other stuff. Among that stuff:
  - Present bit
    - This bit lets us know if a page resides in RAM or storage. That's how the OS deals with not actually having 2^32 * (number of programs) physical addresses in RAM: pages can live on disk when necessary.
  - R/W bit
  - U/S bit
  - These bits let the OS know when to trigger page faults

4. Hierarchical Page Tables
- "Normal" page tables (described above) still use a lot of space

- Page tables have to be allocated all at once or not at all
- Hierarchical page tables solve this by creating a hierarchy of page tables and allocating each table only when it's needed.
  - Virtual addresses get divided into multiple parts, one part per level in the hierarchy + an offset.
- Downside? Speed. Multiple lookups instead of one. More page faults.

5. Kernel
   - Virtualized memory doesn't protect the page table
   - Kernel mode vs. user mode does this
   - Switch between user and kernel modes via interrupts

6. Abstraction
   - Some things can't be virtualized (disk, network, ..)
   - OS abstractions (system calls) make these things portable
   - System calls are implemented as interrupts

7. Virtual memory as naming
   - Virtual memory is just a naming scheme
   - Gives us hiding, controlled sharing, indirection

Next lectures: get rid of our initial assumptions (one CPU per program, etc.)