How Operating Systems Work

Dan Noé
University of New Hampshire / VeloBit

Topics

- A review of how the CPU works
- The operating system “kernel” and when it runs
  - User and kernel mode
  - Device drivers
- Virtualization of memory
  - Virtual memory
  - Paging
- Virtualization of CPU
  - Threads, processes, programs, tasks and Cores
- Virtualization of Disk
  - Filesystems

Review: How the CPU works

- Memory (RAM) contains instructions coded in binary and space for data storage
- The CPU has multiple “registers” for temporary store
- The “instruction pointer” register points to the next instruction to be executed
- Interrupts
  - When a piece of hardware needs something it causes an interrupt which makes execution jump to a predetermined location

How the CPU works: Interrupts

- In order to continue execution after an interrupt the machine context is saved:
  - Save the contents of all registers (including special registers) into memory: the "**Machine Context**"
  - Update the program counter so the next instruction is a predetermined location ("Interrupt Handler")
- If execution is interrupted and the context is saved execution can be resumed later without even knowing!
- When an interrupt occurs the interrupt handler runs in "kernel mode"

User and Kernel Mode

- Application code runs in this mode
- Limited privileges
  - Cannot execute all instructions, memory access limited
- Cannot transition into kernel mode without an interrupt
  - This interrupt is called a “context switch”

User Mode

- System starts in this mode
- Interrupt handlers run in this mode
- Unlimited privileges
  - Access all memory and hardware devices
  - Execute any CPU instruction
- Can transition into user mode with an instruction

Kernel Mode

Kernel

- Term for the collection of code which forms the core of the Operating System
- Kernel code runs first at system boot time
- After the first user mode code runs, the kernel runs only as a result of interrupts
- “Operating System” also includes user mode components
  - User Interface components: Shells, explorer
  - System services

Device Drivers

- Kernel code that interacts with hardware devices
- Sends commands to device
  - Network driver: Send data to remote computer
  - Disk driver: Begin write or read or read operation
  - Video driver: Display something on screen
- Handles interrupts by that device
  - Network driver: Interrupt occurs when incoming data arrives
  - Disk driver: Interrupt occurs when operation is complete
  - Human interface device: Keyboard press causes interrupt

System call

- But how does a user application request something from the kernel?
- It causes an interrupt!
- Old way: Software interrupt instruction
  - A software interrupt (also called a “trap”) is triggered by a special instruction
- Newer way: Special “sysenter” or “syscall” instruction
  - Allows some shortcuts by the CPU and kernel for better performance

Virtualization

- **A key concept**: Each user application runs on the CPU as though it has full access to the system through virtualization of resources

Virtualization of Memory

- Each program acts as though it has universal access to the memory
  - Without this life would be difficult!
- Virtual Memory is the solution
- Address spaces
  - Physical addresses – size is the amount of physical RAM
  - Virtual addresses – size depends on the size of a pointer (32 or 64 bits)
    - 32-bit system: 4GB of virtual address space
    - 64-bit system: 16 Exabytes!

Paging

- Physical address space is divided into 4 kilobyte **Page Frames**
- Virtual address space is divided into 4 kilobyte **Pages**
  - 32 bit virtual address 0xdeadbeef
  - Page number
  - Offset within page
- A **page table** is maintained in memory for each program. It maps virtual Page Frame -> Physical Page
- For each memory access the CPU looks up the virtual address and translates it into a physical address

Paging

- The Page Fault (a trap/interrupt)
  - A virtual memory address is accessed which does not have a valid or resident physical page
- Demand Paging and nonresident pages
- File backed pages
- Shared pages
  - Copy on Write
- Making it fast: Caching and the Translation Lookaside Buffer (TLB)

Dirty Pages

- If a page has been resident in memory and modified then it is marked “dirty”
- If a dirty page is backed by a disk file then it must be written out (flushed) eventually
- If there are no free page frames the kernel can old flush pages to disk and evict them making them nonresident.
- Insufficient memory prompts “thrashing” and extremely poor performance

CPU Scheduling

- **Tasks**: Threads, Processes, Programs, Applications
- **Cores**: Multiprocessing, Multiple Cores, Multithreading
- **CPU Scheduling** maps **Tasks** to **Cores**
  - Much like virtual memory maps virtual addresses to physical!
- The Scheduler decides what task will run next after servicing an interrupt or system call

IO Bound vs. CPU Bound tasks

- Time to execute one CPU instruction: 0.4 nanoseconds
- Time to retrieve data from main memory: 18 nanoseconds
- Time to retrieve data from disk: 8+ milliseconds (8,000,000 nanoseconds)
- Time to retrieve data from California server: 70 ms (70,000,000 nanoseconds)
- A CPU which executes one instruction per second would take about 231 days to seek the hard drive!

IO Bound vs. CPU Bound tasks

- Most tasks spend most of their time waiting for IO
- The scheduler will allow CPU bound (or other IO bound tasks) to run while waiting
- A typical program might only run hundreds of instructions before blocking on more IO
- If no tasks are ready to run the CPU will instruct the processor to save power
- A timer interrupt fires regularly to interrupt any CPU bound tasks

Filesystems: More virtualization!

- User applications cannot write directly to the disk hardware
- Instead, the OS organizes files using a filesystem and allows programs to access the disk through system calls
- User programs use a standard interface to name, read, and write “files”. The filesystem translates this into actual disk block operations.

Caching

- The difference between main memory and disk access latency means that the OS must cache disk data aggressively
- Leverage virtual memory with a Page cache:
  - Each file opened by a user program is loaded by mapping file backed pages
  - Pages from the file are loaded on demand when a page fault occurs
  - Reads and writes are memory operations. The pages will be marked dirty if modified

Special variations

- Hypervisors, mobile devices, etc.

Virtualization of OS: Hypervisor

- VMWare, Parallels, VirtualBox, Xen, KVM, and Hyper-V are examples of “hypervisors” which are Operating System kernels which can run other operating systems
- Same concepts apply!
- Interrupts are initially serviced by the Hypervisor then control is passed to guest operating system kernel

Mobile devices

- Desire for low power consumption means the CPU should be powered off as much as possible
- Memory pressure means that while there may be “multitasking” applications can be killed at any time
- Cooperative multitasking/state saving helps
  - Callbacks to save state
  - Save to persistent storage immediately
- Different paradigms for file storage

Review

- An Operating System is composed of the “Kernel” and “user space” utilities
- The “Kernel” is the core code which provides **virtualization of resources**. It has several key components:
  - Virtual memory manager
  - CPU scheduler
  - Filesystems / block layer
  - Networking

Learning more

- Wikipedia
- Learn about Linux Kernel: LWN.net, kernelnewbies.org
- Install Linux (free) in VirtualBox (free – runs on top of Windows)
  - Make modifications to the kernel and the worst you do is crash your virtual machine!
- Explore the Linux kernel source: http://lxr.linux.no