

Intro to Operating Systems

Virtual Memory

Naming



COS 316: Principles of Computer System Design

Lecture 3

Wyatt Lloyd

Operating Systems by Example

- Mobile OSes:
 - iOS, Android
- Laptop/Desktop OSes:
 - Windows, macOS, Linux, ChromeOS
- Server OSes
 - Linux, Windows
- Embedded OSes:
 - Pi OS, Tock OS

Operating Systems by Counterexample

- Running Firefox on a laptop without an OS
- Need to rewrite Firefox to replace all its syscalls
 - recvmsg/sendmsg – talk to the NIC to recv / send messages over the network
 - This is specific to the particular type of NIC in your laptop!
 - mmap – allocate memory in a virtual address space
 - ...

Operating Systems by Counterexample

- Running Firefox and Zoom on a laptop without an OS
- Need to rewrite Firefox to replace all its syscalls
- Need to rewrite Zoom to replace all its syscalls
- Need to join Firefox and Zoom together into one binary that does both
 - When does Firefox functionality run vs Zoom?
 - How do we keep Firefox from goofing up Zoom's memory?
 - What happens when there is a buffer overflow in Zoom and then you access your bank account via Firefox?

Operating Systems to the Rescue 1

- Abstract underlying resources
 - CPU, Memory, Network, Disk
- Much easier to use: implement 1x well, use 10^6 times
- Portable: run on different CPUs, different NICs, different Disks

Operating Systems to the Rescue 2

- Share underlying resources
- Firefox and Zoom and Slack and Terminal can all use the CPU, memory, network, and disk
- With **isolation**: Zoom can have a buffer overflow, but even when compromised can't read Firefox's memory
 - And Slack can't make your terminal crash

Operating Systems in this Class

- Virtual Memory
 - How can we isolate the memory of different applications?
 - How can we have applications think they have more memory than actually exists on the machine?
 - Introduce **Naming** an important general systems concept
- Unix File System
 - How to build a hierarchical file system over a disk
 - Introduce **Layering** an important general systems concept
- Reasoning about Performance, OS style

Virtual Memory

Let's Have Applications Use Memory



Applications need a way to **name** memory

- Where are reads and writes going?!

Here?

Here?



Why does naming matter?

- Naming is a central design choice in the interface of a system
- Recall: Systems provide an interface to underlying resources
 - Mediate access to shared resources
 - Isolate applications
 - Abstract complexity
 - Abstract differences in implementation
- We always need some way for applications (or other clients) to name those resources

Why does naming matter?

- The names systems use to expose underlying resources affects every other aspect of the system:
 - Performance of the system implementation
 - Application performance and flexibility
 - Security & Isolation
 - Portability
 - Resource sharing and concurrency

Naming Memory #1: Geometric memory (strawman)

- **Values:** Words of memory
- **Names:** DIMM 1; BANK 3; ROW 1200; COLUMN 4;
 - Specifies the precise location of the word(s)
- **Lookup** mechanism: direct in simple hardware

Naming Memory #2: Physical memory

- **Values:** Words of memory
- **Names:** 0xDEADBEEF
 - Integer up to the maximum size of memory in words
- **Lookup** mechanism: direct in simple hardware

Comparing Geometric and Physical

- Performance of the system implementation
 - Application performance and flexibility
 - Security & Isolation
 - Portability
 - Resource sharing and concurrency
-
- All essentially the same
 - But physical is more **portable** than geometric
 - (Geometric is not real for memory, dominated by physical)

Issues with Physical Memory

- How can I isolate one application from another?
 - Slack can directly access memory used by Firefox
- How can I run an application that wants more memory than a machine physically has?
 - How can I run a collection of applications that want more memory than a machine physically has?
- **Hide the physical names and instead use virtual names!**

Naming Memory #3: Virtual memory

- **Values:** (type, address)
 - Type is a type of storage; address is storage specific
 - (Memory, memory address)
 - (File, file name and offset in file)
 - (Remote memory, remote node and memory address)
 - ...
- **Names:** 64 bit address & process ID
 - E.g., (0xDEADBEEF, 1337)
 - process ID is typically implicit
- **Allocation:** mmap

mmap system call

- `void *mmap(void *addr, size_t length)` (simplified)
- Application chooses an unused name: an address not yet allocated for it
 - (Or can pass in NULL if it doesn't care)
- OS Kernel (the system!)
 - keeps a list of unused physical 4KB memory pages
 - allocates "value" by removing a physical page from the list
 - adds mapping between virtual address and physical to the application's "page table"
 - in-memory data structure understood by virtual memory hardware that maps virtual addresses to physical addresses

Virtual memory lookup

- Lookup virtual address in “page table”
 - Stored in memory (where it is “pinned”)
 - OS maintains one page table per process
 - Page table maps virtual address to physical memory address
OR file and location OR remote machine and memory address
- Performance implications?

Naming Memory #3: Virtual memory

- **Values:** (type, address)
 - Type is a type of storage; address is storage specific
 - (Memory, memory address)
 - (File, file name and offset in file)
 - (Remote memory, remote node and memory address)
 - ...
- **Names:** 64 bit address & process ID
 - E.g., (0xDEADBEEF, 1337)
 - process ID is typically implicit
- **Allocation:** mmap
- **Lookup:** TLB, page table, disk, ...

Virtual memory lookup

- Lookup virtual address in TLB (Translation lookaside buffer)
 - Small hardware implemented cache
 - Hit → translates to physical address
- TLB miss goes to “page table”
 - Stored in memory (where it is “pinned”)
 - OS maintains one page table per process
 - Page table maps virtual address to physical memory address
OR file and location OR remote machine and memory address

Comparing Physical and Virtual

- Application performance
- Application flexibility
- Security (Isolation)
- Resource sharing and concurrency
- Portability

Winner?

- Physical
- Virtual
- Virtual
- ~Same
- ~Same

What type of memory naming to use?

1. On your laptop
2. For a tiny power constrained microcontroller
3. For a supercomputer that runs one massive simulation at a time
4. On your phone

Virtual Memory and Naming Summary

- Names are the way systems expose resources to applications
- Central to designing and understanding systems
 - Performance
 - Security
 - Caching
 - Resource sharing
- Virtual Memory
 - Names are unique to each process → isolation
 - Names are translated to physical addresses → can have larger address space than physical memory
 - Hardware support (TLB) makes the implementation efficient