Introduction to Naming

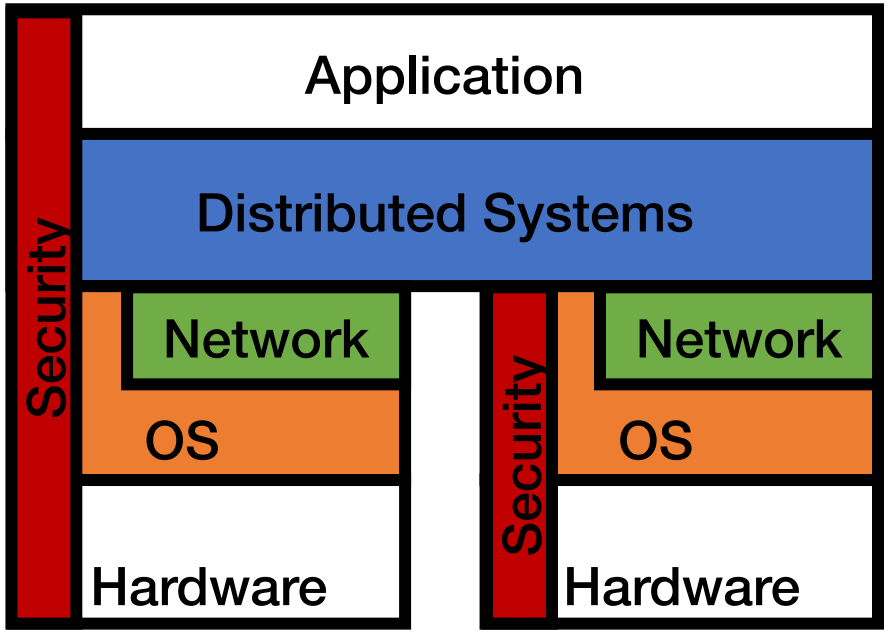
COS 316: Principles of Computer System Design

Lecture 3

Wyatt Lloyd & Rob Fish

Naming Module

- Naming Overview (today)
  - Memory Naming
    - OS, Security
- Unix File System Naming
  - OS
- Git Naming
  - OS, Distributed Systems
- Network Naming
  - Networking



Application: “I would like to send data to the Internet host, please.”

System: “Which host?”

Application: “Oh uh … `cs.princeton.edu`”

Application: “I would like to send data to the Internet host, please.”

System: “Which host?”

Application: “Oh uh … `cs.princeton.edu`”

`cs.princeton.edu` is the name for an IP address!

Application: “Can I please get the data?”

System: “You’re gonna have to be more specific.”

Application: “The data in `/home/wlloyd/316-revamp.txt`”

Application: “Can I please get the data?”
System: “You’re gonna have to be more specific.”
Application: “The data in `/home/wlloyd/316-revamp.txt`”

`/home/wlloyd/316-revamp.txt` is the name for a bunch of sectors on disk!

Application: “What is the sum of two numbers?”

System: “I really need to know which numbers…”

Application: “Fine, fine, fine: the ones in registers `r1` and `r2`.”

Application: “What is the sum of two numbers?”

System: “I really need to know which numbers…”

Application: “Fine, fine, fine: the ones in registers `r1` and `r2`.”

`r1` and `r2` are names for words of memory residing in CPU registers!

Whenever an application uses a resource, it must somehow name it.

Agenda

- Why does it matter?
- An intellectual framework for naming
- Naming memory

Why does naming matter?

- Naming is the most 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 Scheme Framework

- **Values:** What is it that we're naming?
  - Disk sectors?
  - Network nodes?
  - Users?

- **Names:** What's the format of a name?
  - Alphanumeric strings up to 32 characters
  - Non-zero integers
  - 128-bit numbers

- **Allocation** mechanism: How does the system create new names and values?

- **Lookup** mechanism: How does the system map from names to values?

Let’s Name Memory

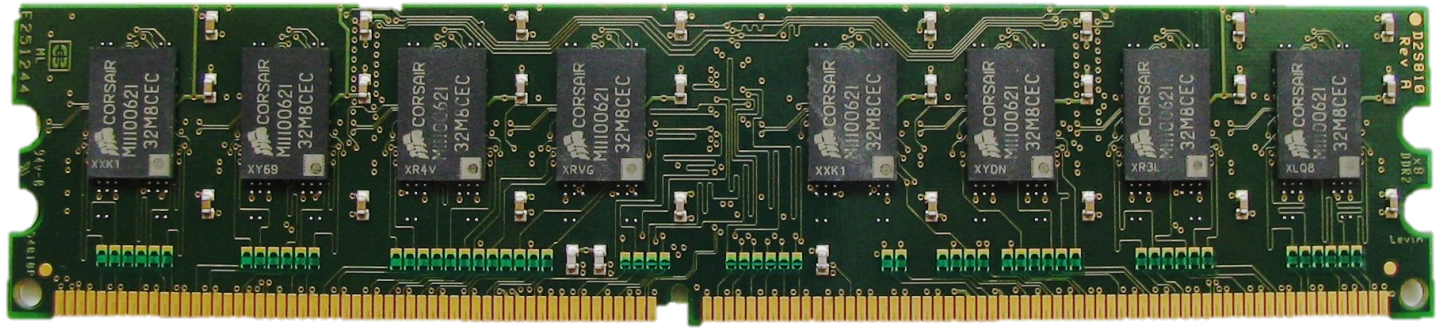


Image from: https://commons.wikimedia.org/wiki/File:DRAM_DDR2_512.jpg

Naming Memory #1: Geometric memory

- **Values:** Words of memory
- **Names:** `DIMM 1; BANK 3; ROW 1200; COLUMN 4;`
  - Specifies the precise location of the word(s)
- **Allocation:** n/a
  - Or install more memory
- **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
- **Allocation:** n/a
  - Or install more memory
- **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)

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)

- 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

	Winner?
• Performance of the system implementation	• Physical
• Application performance	• Physical
• Application flexibility	• Virtual
• Security (Isolation)	• Virtual
• Effectiveness of caching	• Physical
• Resource sharing and concurrency	• ~Same
• Portability	• ~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

Naming Memory #4: Original UNIX

- Swap out all memory for one process at a time
  - Allows using physical addresses with isolation!
  - Simple and efficient to implement in hardware
  - Can’t run applications in parallel
  - Expensive to switch between applications

Naming Memory #5: Segmentation

- Virtual addresses are low-order bits of physical address + segment register
  - Relatively simple hardware
    - (just concatenate segment register and virtual address)
  - Isolates concurrent applications using names
  - Much coarser grain: all virtual memory must be contiguous in RAM
  - Can’t share memory between applications

Summary

- Names are the way systems expose resources to applications
- Central to designing and understanding systems
  - Performance
  - Security
  - Caching
  - Resource sharing
- Framework for naming:
  - Values
  - Names
  - Allocation mechanism
  - Lookup mechanism