Introduction to Naming

COS 316: Principles of Computer System Design

Amit Levy & Wyatt Lloyd

- Today: Naming schemes in systems
- Next, naming in:
  - UNIX file system
  - Git
  - Naming in Networking

Assignment 1 is out Due 9/16. *Hint: you really want to attend precept!*

Why do systems need to name things?

Why do systems need to name things?

- Abstract complexity: expose underlying resources using simpler names

Why do systems need to name things?

- Abstract complexity: expose underlying resources using simpler names
  - File names instead of subdivisions of magnetic disks

Why do systems need to name things?

- Abstract complexity: expose underlying resources using simpler names
  - File names instead of subdivisions of magnetic disks
- Abstracts differences in implementation: expose different resources using the same names

Why do systems need to name things?

- Abstract complexity: expose underlying resources using simpler names
  - File names instead of subdivisions of magnetic disks
- Abstracts differences in implementation: expose different resources using the same names
  - File names for data on disks, solid-state drives, or network storage

Why do systems need to name things?

- Abstract complexity: expose underlying resources using simpler names
  - File names instead of subdivisions of magnetic disks
- Abstracts differences in implementation: expose different resources using the same names
  - File names for data on disks, solid-state drives, or network storage
- Isolate applications: the same name refers to different resources for different applications

Why do systems need to name things?

- Abstract complexity: expose underlying resources using simpler names
  - File names instead of subdivisions of magnetic disks
- Abstracts differences in implementation: expose different resources using the same names
  - File names for data on disks, solid-state drives, or network storage
- Isolate applications: the same name refers to different resources for different applications
  - `chroot`: give each application it's own root of the file system

So many examples of names!

http://www.princeton.edu/news	URL
www.princeton.edu	hostname
140.180.223.42	IP address
`main`	procedure name
`pc`	ARM register name
aalevy@cs.princeton.edu	email
`~/Documents/lecture.pdf`	file name

Why are names important?

- The primary way applications/users interact with a system

Why are names important?

- The primary way applications/users interact with a system
- Often the first step in designing a system

Why are names important?

- The primary way applications/users interact with a system
- Often the first step in designing a system
    - A central design decision

Why are names important?

- The primary way applications/users interact with a system
- Often the first step in designing a system
  - A central design decision
  - Can make implementation easier or obvious

Why are names important?

- The primary way applications/users interact with a system
- Often the first step in designing a system
  - A central design decision
  - Can make implementation easier or obvious
- A good place to start understanding systems

Why are names important?

- The primary way applications/users interact with a system
- Often the first step in designing a system
  - A central design decision
  - Can make implementation easier or obvious
- A good place to start understanding systems
- Choice of naming scheme can dictate system trade-offs:

Why are names important?

- The primary way applications/users interact with a system
- Often the first step in designing a system
  - A central design decision
  - Can make implementation easier or obvious
- A good place to start understanding systems
- Choice of naming scheme can dictate system trade-offs:
  - Performance

Why are names important?

- The primary way applications/users interact with a system
- Often the first step in designing a system
  - A central design decision
  - Can make implementation easier or obvious
- A good place to start understanding systems
- Choice of naming scheme can dictate system trade-offs:
  - Performance
  - Security

Why are names important?

- The primary way applications/users interact with a system
- Often the first step in designing a system
  - A central design decision
  - Can make implementation easier or obvious
- A good place to start understanding systems
- Choice of naming scheme can dictate system trade-offs:
  - Performance
  - Security
  - Portability

Why are names important?

- The primary way applications/users interact with a system
- Often the first step in designing a system
  - A central design decision
  - Can make implementation easier or obvious
- A good place to start understanding systems
- Choice of naming scheme can dictate system trade-offs:
  - Performance
  - Security
  - Portability
  - Generality

Naming and system trade-offs

Let’s compare alternate naming systems from the physical world:

Building numbers in a city block

Naming and system trade-offs

Let’s compare alternate naming systems from the physical world:

Building numbers in a city block

New York -vs- Tokyo

Figure 1: How do we name each of these buildings?

How do buildings get their name?

How do buildings get their name?

1. Subdivide the block into N plots numbered `1-N`
2. Each building fits in exactly one of the plots
3. Name buildings with the number of their plot

How do buildings get their name?

1. Subdivide the block into N plots numbered `1-N`
2. Each building fits in exactly one of the plots
3. Name buildings with the number of their plot

So: `344 10th Ave` is one plot south of `346 10th Ave`

How do buildings get their name?

1. Subdivide the block into N plots numbered `1-N`
2. Each building fits in exactly one of the plots
3. Name buildings with the number of their plot

So: `344 10th Ave` is one plot south of `346 10th Ave`

But... what happens when we build the `N+1`th building?

Tokyo

1987

1993

2012

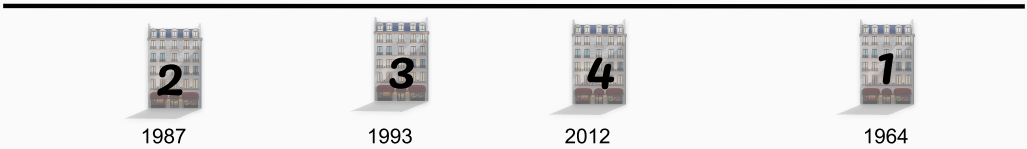
1964

How do buildings get their name?

How do buildings get their name?

Buildings numbered from oldest to newest.

Whenever a new building is built, give it the next highest number on the block

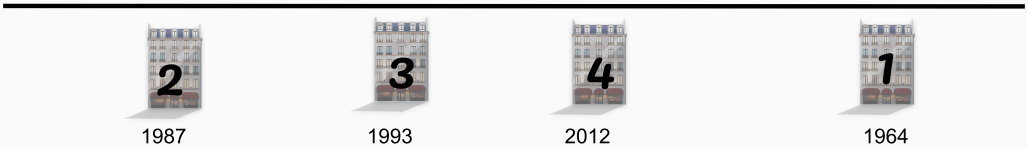


How do buildings get their name?

Buildings numbered from oldest to newest.

Whenever a new building is built, give it the next highest number on the block

Finding a building is hard! Need to check buildings until you find the right number.



How do buildings get their name?

Buildings numbered from oldest to newest.

Whenever a new building is built, give it the next highest number on the block

Finding a building is hard! Need to check buildings until you find the right number.

But... virtually infinite possibility for expansion!

Different naming and trade-offs

	New York Addressing	Tokyo Addressing
Easy allocation	no :(	YES
Easy lookup	YES	no :(

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?

Applying Framework to Building Numbers

	New York Addressing	Tokyo Addressing
Values	Physical buildings	Physical buildings
Names	Numbers 1-N	Numbers >=1
Allocation	Pre-defined plot numbers	New building gets next highest number
Lookup	Search ordered set	Visit every building and check

Virtual Memory

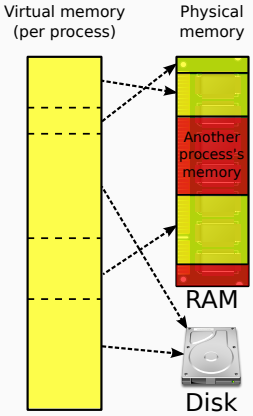


Figure 2: Virtual Memory

What does virtual memory give us?

- Isolation: `0xdeadbeef` maps to different values for different applications
- Abstraction over storage media: virtual addresses can name memory, storage, remote memory, HW register, ...
- Flexibility in provisioning
  - Resource efficiency: physical memory can be less sparse than virtual memory
  - Overcommit physical memory (e.g. swap to disk when necessary)

Virtual Memory as a Naming Scheme

- Values?
- Names?
- Allocation?
- Lookup?

Virtual Memory: Values

- *Physical memory address (i.e. 32-bit or 64-bit address up to size of RAM)*

Virtual Memory: Values

- *Physical memory address (i.e. 32-bit or 64-bit address up to size of RAM)*
- On-disk file and offset in file

Virtual Memory: Values

- *Physical memory address (i.e. 32-bit or 64-bit address up to size of RAM)*
- On-disk file and offset in file
- Some hardware registers (e.g. a network card configuration registers)

Virtual Memory: Values

- *Physical memory address (i.e. 32-bit or 64-bit address up to size of RAM)*
- On-disk file and offset in file
- Some hardware registers (e.g. a network card configuration registers)
- Remote memory

Virtual Memory: Names

Pointer-sized (e.g. 32-bit or 64-bit) addresses and process identifiers

$(PID, virtual\_address)$

e.g.

$(3487, 0xdeadbeef)$

Virtual Memory: Allocation

`void *mmap(void *addr, size_t length)` (simplified)

- Application chooses an unused name: an address not yet allocated for it
- Kernel keeps a list of unused physical 4KB memory pages
- Kernel allocates “value” by removing a physical page from the list
- Kernel adds new mapping between virtual address and physical to the application’s “page table”
  - in-memory data structure understood by the virtual memory hardware that maps virtual addresses to physical addresses

Virtual Memory: Lookup

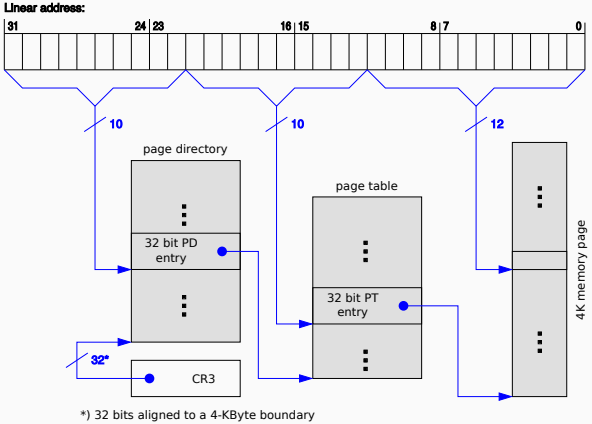


Figure 3: Two-level page table structure in x86

Virtual Memory: Lookup

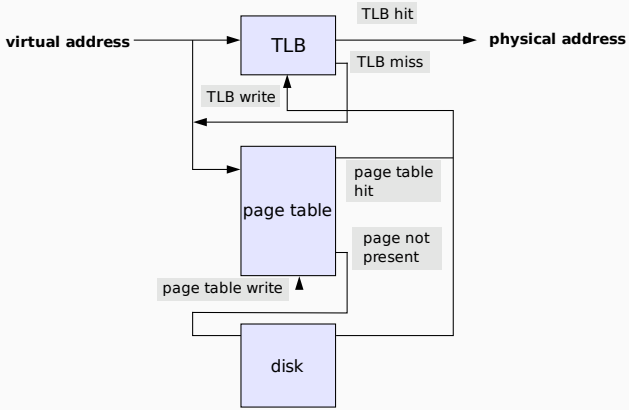


Figure 4: Virtual-to-physical translation

Virtual Memory: Lookup

For this to work, the OS needs to do some housework when context switching:

- Set CR3 register to point to process's page table
- Invalidate the TLB
  - Mark entire TLB as invalid—simple but can cause unnecessary slow down
  - Associate process IDs with each TLB entry

Virtual Memory: Trade-offs with Alternative Naming Schemes

- Single shared address space (identity mapping)
  - Better performance, since no translation hardware/software needed
  - But can't isolate using names
- Swap out all memory for one process at a time (original UNIX)
  - Simple and efficient to implement in hardware
  - Can't run applications in parallel
  - Expensive to switch between applications
- Segmentation: virtual address 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

- Systems use names to abstract complexity, abstract differences, isolate applications
- Naming choices can dictate system trade-offs
  - performance: fast allocation vs. fast lookup
  - isolation
- Framework for naming:
  - Values
  - Names
  - Allocation mechanism
  - Lookup mechanism