Naming things

Naming: map names to objects
- Helps with using, sharing, and communicating information

Examples
- User names: *used for system login, email, chat*
- Machine names: *used for ssh, email, web*
- Files
- Devices
- Objects, functions, variables in programs
- Network services
What’s a name?

**Name**: identifies what you want

**Address**: identifies where it is

**Route**: identifies how to get there

**Binding**: the association of a name with the object
What’s a name?

**Binding:** the association of a name with the object

“*choose a lower-level-implementation for a particular higher-level semantic construct*”

— Jerry Saltzer, *Naming and Binding of Objects*

```
ls.cs.rutgers.edu → 128.6.13.171
```

```
```
Pure & Impure Names

Pure names – *identify*

- The name contains no information aside from the name
- It does not identify *where* the object can be found

Examples:

- `c8:2a:14:3f:92:d1` my computer’s ethernet MAC address
- `p_k` my Twitter handle
- `908-555-3836` phone #
  (phone #s used to be impure names since they contained an area_code-exchange-line hierarchy)
Impure names – *guide*

- The name contains context information
- Object is generally unmovable
- Examples:
  - pk@pk.org, pxk@cs.rutgers.edu, happyuser@verizon.net
    - User names in different Internet domains: same person or not?
    - Context (domain name) is encoded into the name
  - /home/paul/bin/qsync
    - File pathname changes if we move the object
Uniqueness of names

• Easy on a small scale – problematic on a large scale
  – It can be difficult to make globally unique names

• Uniqueness for pure names
  – Designate a bit pattern or naming prefix that does not convey information
    • Ethernet MAC address: 3 bytes: organization, 3 bytes: controller
    • IP address: network & host (variable partition)

• Uniqueness for impure names — use a hierarchy
  – Compound name: iterative list of pure names connected with separators
    • Domain name: www.cs.rutgers.edu
    • URLs: https://pk.org/417/lectures/intro.html
    • File pathnames: /usr/share/dict/words
Naming convention determines syntax for names

Ideally, a format that will suit the application and user

- E.g., human-readable names for humans, binary identifiers for machines

  - Programming languages
    Rules for choosing variable names

  - UNIX file names:
    Parse components from left to right separated by /
    `/home/paul/src/gps/gui.c`

  - Internet domain names:
    Ordered right to left and delimited by .
    `www.cs.rutgers.edu`

  - LDAP names
    - Attribute/value pairs ordered right to left, delimited by ,
    - `cn=Paul Krzyzanowski, o=Rutgers, c=US`
A particular set of name -> object bindings

- Names are unique within the context
  - E.g., /etc/postfix/main.cf on a specific computer

- Each context has an associated naming convention

- A name is always interpreted relative to some context
  - E.g., directory /usr in a Linux file system on crapper.pk.org
Name Service

The service that performs name resolution

Allows you to resolve names

- Looking up a *name* gives the corresponding *address* as a response

Can be implemented as

- Search through file
- Database query
- Client-server program (*name server*) – may be distributed
- ...

March 9, 2023  © 2014-2023 Paul Krzyzanowski
Often completely synonymous with Name service

• Extension of name service:
  – Associates names with objects, where objects have attributes
  – Can query for specific attributes
    • Example: LDAP (Lightweight Directory Access Protocol)

• Sometimes refers to searching through a hierarchical namespace
A container for a set of names in the naming system

• A namespace has a scope
  – **scope** = region where the name exists & refers to the object
  – For example,
    • Names of all files in a directory
    • All domain names within rutgers.edu
    • E.g., Java package, local variables

• A namespace may be tree structured (hierarchical)
  – Fully-qualified or hierarchical names may be used to identify names outside the local namespace
  – **Global namespace** = root of the tree
Terms: Resolution

• **Resolution** = name lookup
  – Return the underlying representation of the name
  – Look up the **binding** of the name to its object

• For example,
  – www.rutgers.edu → 128.6.4.5

• **Iterative** resolution
  – Example: parse a pathname

• **Recursive** resolution
  – Example: parse a distribution list: each entity may be expanded
When do should you do a resolution?

**Static binding**
- Hard-coded

**Early binding**
- Look up binding before use
- Cache previously used binding

**Late binding**
- Look up just before use

These can be efficient but can create stale bindings
The End
IP Domain Names

Human readable names
  e.g., www.cs.rutgers.edu

Hierarchical naming scheme
  – Top of hierarchy on the right
  – No relation to IP address or network class