CIS 505: Software Systems
Lecture Note on Naming: Part 1

Insup Lee
Department of Computer and Information Science
University of Pennsylvania

Spring 2007

Examples

- What can you do with the following names/ids:
  - phone number, driver's license number, email address, etc.
- What are their properties?
- What do you do when change your phone number?
  - Email/call friend ASAP
    - < 400 friends?
  - White page updated once a year
    - Changes are infrequent

Terminology

- A name is *bound* to the object it names
- If X names Y, then the tuple <X,Y> is a binding
- A set of bindings is a *context*
- In a context:
  - Unique = only one tuple with X = **A**
  - Ambiguous (or generic) = multiple tuples with X = **A**
  - Invalid = no tuple with X = **A**
  - Alias = multiple bindings with Y = **B**

Naming in (distributed) computer systems

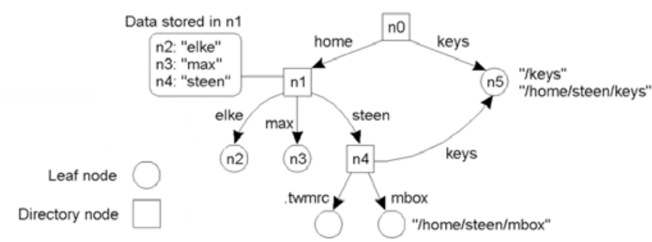
- Need to name operations and objects
- Name
  - A string of bits or characters
  - Refers to an entity
    - Hosts, printers, disks, files, users, processes, web pages, etc.
  - Access point - to operate on an entity, need to access it
- Address
  - An entity which is the name of an access point
  - Refers to an access point of an entity
  - An entity may have more than one access points
    - Person with many phone numbers
  - Could change for the same entity
    - IP address of FTP server for cis.upenn.edu
- Location independent names
- True identifier
  - An id refers to at most one entity
  - Each entity is referred to by at most one id
  - An id always refers to the same entity (i.e., never reused)

Name Space

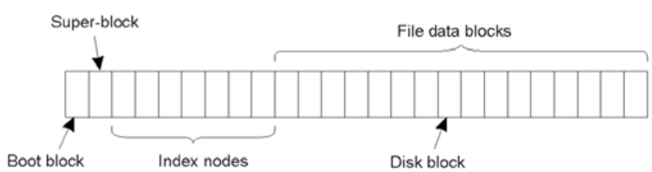
- Name space – organization of names
  - Need to create names and lookup names
  - Represented as a labeled, directed graph with two types of nodes:
    - Leaf node – a named entity (and stores information)
    - Directory node – a directory table of (edge label, node id)
  - A naming path – N: <label-1,label-2, …,label-n>
  - Path name
    - Absolute path name vs. relative path name
  - Global name vs. local name

Example: Hierarchical Name Spaces

- A general naming graph with a single root node.



Example: Disk Block Name Spaces



- The general organization of the UNIX file system implementation on a logical disk of contiguous disk blocks.

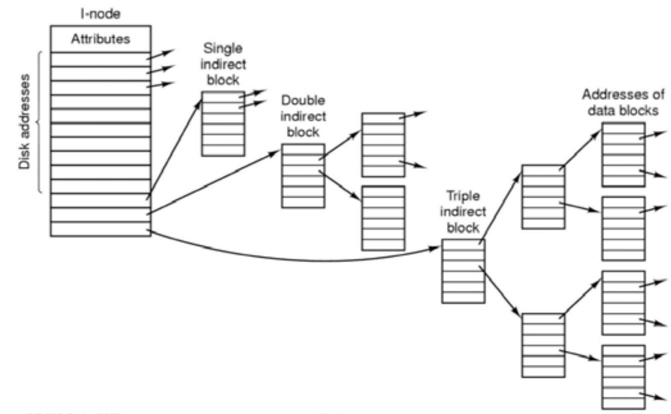
The Classic Unix File System

- Each disk partition has 4 Regions
  - block 0: boot block
  - block 1: super block. Contains the size of the disk and the boundaries of the other regions
  - i-nodes: list of i-nodes, each is a 64-byte structure
  - free storage blocks for the contents of files.
- Each i-node contains owner, protection bits, size, directory/file and 13 disk addresses.
- The first 10 of these addresses point directly at the first 10 blocks of a file. If a file is larger than 10 blocks (5,120 bytes), the 11th points at a block for secondary indexing – single indirect block

Classic Unix File System Cont.

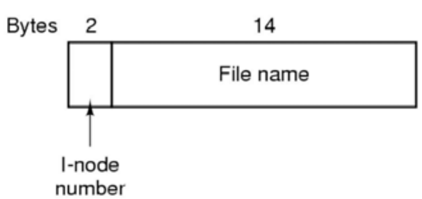
- Second-level index block contains the addresses of the next 128 blocks of the file (70,656 bytes)
- Two levels of indirection:12th entry (double indirect block) points at up to 128 blocks, each pointing to 128 blocks of the file (8,459,264 bytes)
- Three levels of indirection: 13th address (triple indirect block) is for a three layered indexing of 1,082,201,087 bytes.
- A directory is accessed exactly as an ordinary file. It contains 16 byte entries consisting of a 14-byte name and an i-number (index or ID of an i-node). The root of the file system hierarchy is at a known i-number (2).

Unix i-node



Directory Entry: Unix V7

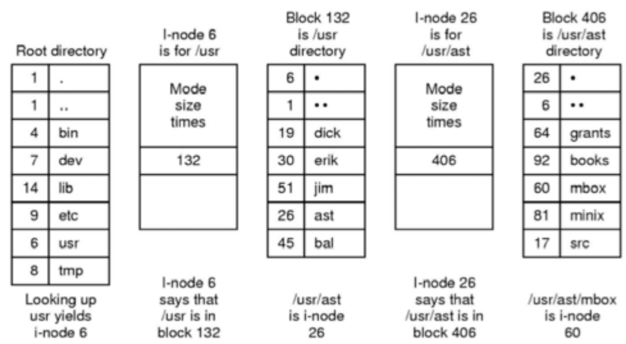
- Each entry has file name and an I-node number
- I-node has all the attributes
- Restriction: File name size is bounded (14 chars)



Traversing Directory Structure

- Suppose we want to read the file /usr/ast/mbox
- Location of a i-node, given i-number, can be computed
- i-number of the root directory known, say, 2
- Read in the i-node 2 from the disk into memory
- Find out the location of the root directory file on disk, and read the directory block in memory
  - o If directory spans multiple blocks, then read blocks until usr found
- Find out the i-number of directory usr, which is 6
- Read in the i-node 6 from disk
- Find out the location of the directory file usr on disk, and read in the block containing this directory
- Find out the i-number of directory ast (26), and repeat

The UNIX V7 Directory Lookup

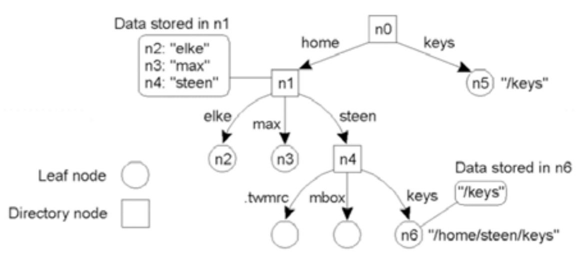


The steps in looking up */usr/ast/mbox*

Name Resolution

- Closure mechanism
  - o Where to start name resolution?
    - Root, HOME
- Merging name spaces
  - o Link and mounting
  - o Create a global name space
- Link and mounting
  - o Aliases
    - Node n5 can be referred two different pat names
    - /keys and /home/steen/keys
  - o Hard links
  - o Symbolic links
  - o How to deal with multiple name space
    - Mount point

Example: Linking and Mounting (1)

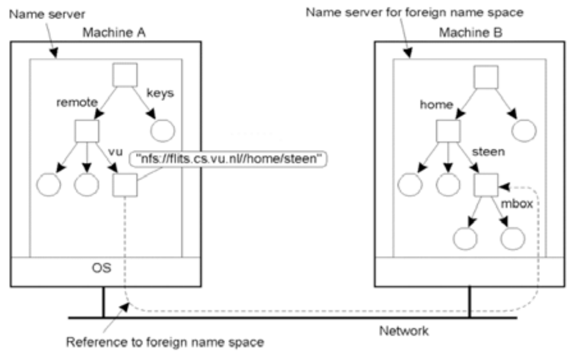


- The concept of a symbolic link in a naming graph.

Mounting name space

- Mount a foreign name space NS2 into NS1
- Need
  - o The name of an access protocol
  - o The name of the server
  - o The name of the mounting point in the foreign name space
- Sun's NFS (Network File System)
  - o nfs://flits.cs.vu.nl//home/steen
  - o Consider: /remote/vu/mbox

Linking and Mounting (2)

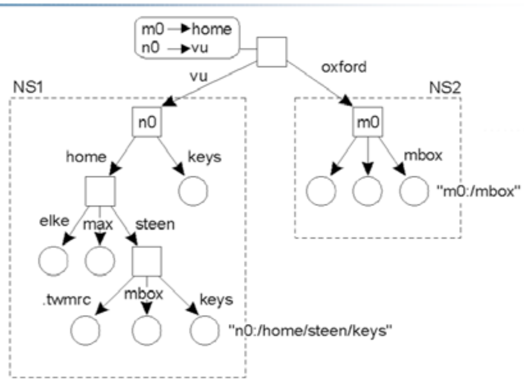


- Mounting remote name spaces through a specific process protocol.

Global Name Space

- An alternative way to merging different name spaces
- Approach
  - o Create a super-tree
  - o DEC's Global Name Service
- Issues
  - o Existing names need to be changed
    - Could hide this by using default prefix
  - o Name space can be cluttered
    - Use hierarchical structure

Global Name Space



- Organization of the DEC Global Name Service

Alternatives to names

- Descriptions
- Contents
- Object itself
- Advantages of name:
  - o Efficiency: shorter, easier to manipulate
  - o Consistency: multiple copies of description may cause problems when an object is modified
  - o Flexibility: a level of indirection allowing symbolic reference, rebinding, changing details without changing object.

Proper Names, Common Names, and Descriptions

- Denotation (reference, identity) vs. Connotation (sense, meaning)
- Proper name = simply identifies an object
  - denotation without connotation,
- Description = a set of properties or predicates that, if satisfied, identifies object
  - connotation without (necessarily) denotation
  - Definite description = denotes unique object
- Common Name = name bound to description or definition *only if we use the description to find the object.*
  - denotation of a connotation
  - Common name have (but proper names don't):
    - synonyms, ambiguity, substitutability, redundancy

Pure Names, Impure Names, and Structured Names

- Continuum from pure name to pure description
  - Pure name = no encoding of a description or entity
  - Impure name = includes some description of named entity
  - Structured name = includes components
  - Pure Description = description with no redundancy
- Tradeoffs between Pure and Impure names
  - Impure can encode info to make mapping easier: phone number
  - Impure names allow reader to intuit info about object w/o accessing object.

System specific trade-offs

- Internal vs. External names
  - Users prefer meaningful names, ASCII: external names
  - Systems prefer concise names, short, efficient (integers, bit strings): internal names
  - Both internal and external are often impure: external for meaning, internal as performance hint.
- Universal vs. Context dependent names
  - Global (universal) names are often long, and unwieldy, and represent mgmt problems
  - Local (context dependent) names must not escape their context
- Locality of names
  - Both external and internal names exhibit high locality; therefore caching and aliases or abbrev's useful (for *both)*

Implementing a Name Space

- Functions of a naming authority
  - Searching for a binding: authority gives *definitive* answer (may delegate, but answer may be out-of-date)
  - Allocation: may delegate, but this can never be out-of-date
  - Validating and invalidating names: for mapping, or revoking privileges, or garbage collection, or ...
- Structures of naming authority
  - Centralized repository
  - Hierarchical
  - Federated (authority module per object)
  - Distributed (e.g., query by broadcast and take first response)

Centralized & Decentralized naming authorities

- Centralized: a single authority governs creation of names, and interpretation of names
  - o Centralized authority may delegate subsets of name-space, or may have caches, or may have distributed implementation --- but there is always a single binding from name to object.
- Decentralized: multiple authorities govern a single name. May be different bindings from different authorities (can view it as a system with only loose or approximate consistency).
  - o Decentralized authority may look up names in multiple contexts, or have multiple managers for a given context.
- Efficiency: centralized more efficient, but harder to extend. Decentralized more expensive (multiple authorities and multiple responses) but easily distributed and more robust.

Lampson's Global Name Service: Design

- "Designing a Global Name Service" (Lampson 1986)
- Database, but slowly changing (both names and bindings)
- Loose integrity / weak consistency
- Requirements:
  - o Large size
  - o Long Life
  - o Highly available
  - o Fault isolation
  - o Tolerance of mistrust
- Use hierarchy to accommodate growth and isolate faults
- Replicate for availability and performance

Lampson's Global Name Service: Implementation

- Centralized or decentralized?
  - o Centralized: need to atomically update and reach (expensive) agreement. (remember scale) But many replicas and "read any, update all" for performance.
  - o Decentralized: read returns approximately up-to-date version, with guarantees that it will never be *too* out-of-date.
- Implementation:
  - o Timestamp every update to allow a sequential order.
  - o Define ring of copies of directory using exact agreement
  - o Periodically "sweep" the ring, propagating all updates
    - additions, modifications, deletions ("absences" or "tombstones")
  - o Use timestamps to resolve conflicts

Different *Types* of Name Service

- "Decentralizing a Global Naming Services for Improved Performance and Fault Tolerance," (Cheriton & Mann,1989)
- Three-level naming architecture
  - o Global: names of administrations, organizations, groups of organizations (e.g., companies, countries, NGO's)
  - o Administrative: Names within an administration people, machines, protocols/services, mailboxes, web servers
  - o Managerial: objects managed by individual servers files, mailing lists, URLs, bank accounts, shopping carts, public keys

Global Name Service

- Properties:
  - o changes slowly, little data
  - o High availability required, since *everyone* interested in up-to-date
  - o Mistrust, since spans many competing administrations
- Matches Lampson's requirements, so can use his design

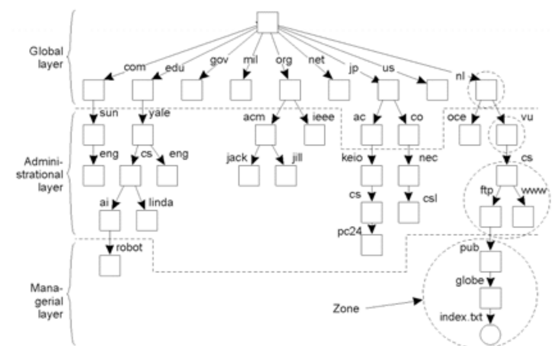
Administrative Name Service

- Single administration, so either no mistrust or hierarchy of trust (trust your boss, but not your peers)
- Availability needed --- but mainly *inside* administration
- Rate of change moderate, but almost strictly *local*

Managerial Name Service

- Possibly very high rate of change and of access
- Availability: usually not needed to be any more available than the object being named.
- Trust --- depends on object (consider files, with per-file protection modes).

Name Space Distribution (1)



- An example partitioning of the DNS name space, including Internet-accessible files, into three layers.

Name Space Distribution (2)

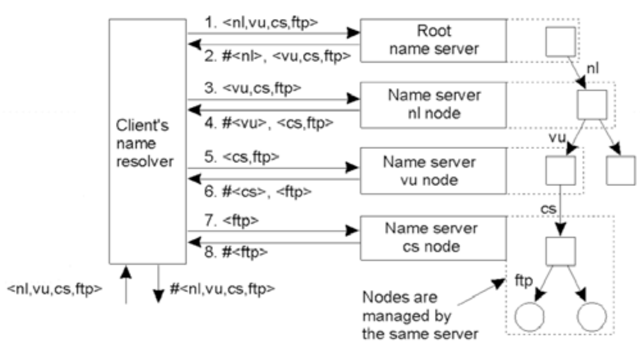
Item	Global	Administrational	Managerial
Geographical scale of network	Worldwide	Organization	Department
Total number of nodes	Few	Many	Vast numbers
Responsiveness to lookups	Seconds	Milliseconds	Immediate
Update propagation	Lazy	Immediate	Immediate
Number of replicas	Many	None or few	None
Is client-side caching applied?	Yes	Yes	Sometimes

- A comparison between name servers for implementing nodes from a large-scale name space partitioned into a global layer, as an administrational layer, and a managerial layer.

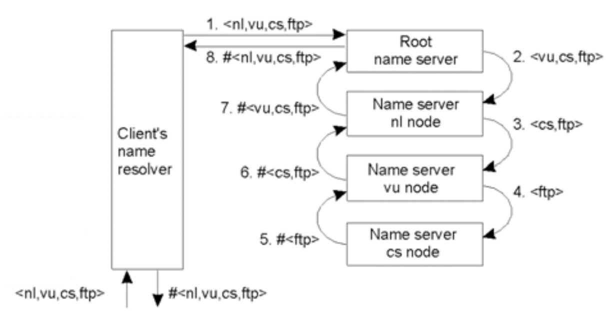
Implementing name resolution

- Iterative name resolution
- Recursive name resolution
- Ex:
  - o root:<nl, vu, cs, ftp, pub, globe, index.txt>
  - o ftp://ftp.cs.vu.nl/pub/globe/index.txt

Implementation of Iterative Name Resolution



Implementation of Recursive Name Resolution

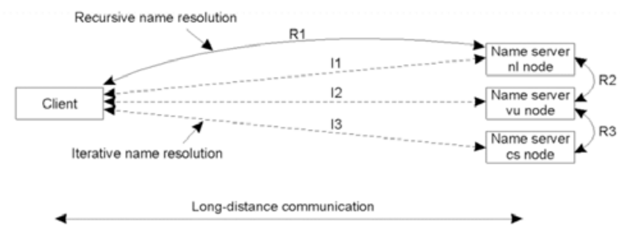


Implementation of Name Resolution (3)

Server for node	Should resolve	Looks up	Passes to child	Receives and caches	Returns to requester
cs	<ftp>	#<ftp>	--	--	#<ftp>
vu	<cs,ftp>	#<cs>	<ftp>	#<ftp>	#<cs> #<cs, ftp>
ni	<vu,cs,ftp>	#<vu>	<cs,ftp>	#<cs> #<cs,ftp>	#<vu> #<vu,cs> #<vu,cs,ftp>
root	<ni,vu,cs,ftp>	#<nl>	<vu,cs,ftp>	#<vu> #<vu,cs> #<vu,cs,ftp>	#<nl> #<nl,vu> #<nl,vu,cs> #<nl,vu,cs,ftp>

- Recursive name resolution of <*nl, vu, cs, ftp*>. Name servers cache intermediate results for subsequent lookups.

Implementation of Name Resolution (4)



- The comparison between recursive and iterative name resolution with respect to communication costs.

Global Name Design Space

- Design considerations
  - Size of local directory
  - Delay and cost of update propagation
  - Security
  - …
- Challenges
  - Scale
  - Frequency of change
  - Precision/consistency
  - Performance
  - Robustness
  - Rate of update and lookup
  - Trust: many people update