Name Mappings

Peter J. Denning

© 2022, Peter J Denning

Purpose

- Enable a process to perform an operation on an object given its name
- Two questions:
  1. How to find the object in the (vast) memory space?
  2. How to tell if requesting process is authorized?
- This chapter focuses on question 1
- Next chapter focuses on question 2

© 2022, Peter J Denning

Name Space

- Set of all names satisfying a given syntax format and mappable to objects
  - All possible addresses in a virtual memory
  - All possible pathnames in a directory hierarchy
  - All possible hostnames in Internet
  - All possible IP addresses
  - All possible URLs
- Can be bounded or unbounded in size

© 2022, Peter J Denning

Name Map

- A table (or tables) associating names with pointers to their locations in memory or network
  - Examples: page table, interrupt vector, kernel entry vector, process control block list, semaphore control block list, file index table, file directory, + more
- Translation: process of using the table to map a name to its location pointer

© 2022, Peter J Denning

Mapping Tables

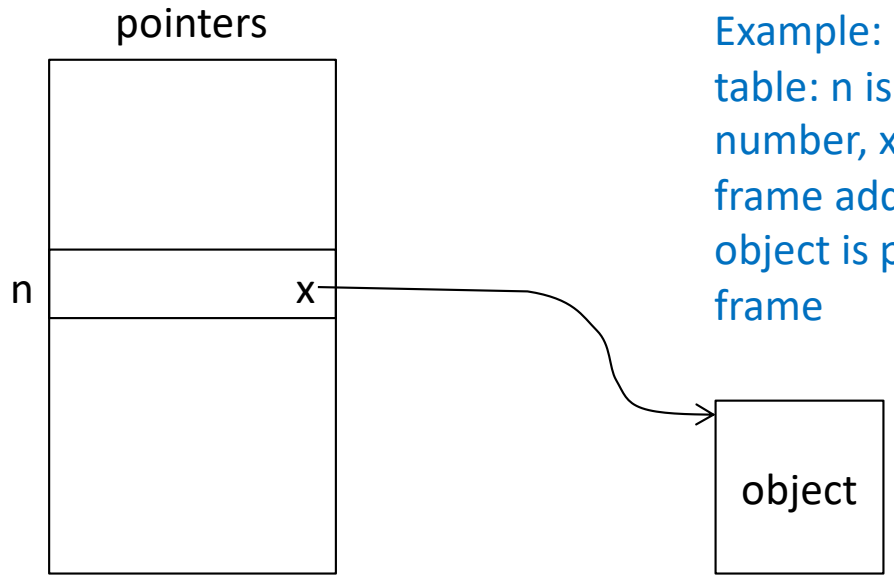
- A table whose entries associate names of objects with with object locations
  - Names have fixed maximum length k bits
  - Name space is 2^k names
- Two table types:
  - If 2^k small enough, use index table
  - Otherwise, use associative or hash table

© 2022, Peter J Denning

Index Table

- k-bit names: name n is one of 0,1,2,…,2^k-1
- Table of size 2^k, one row for each possible name

If names are integers 0,1,…,2^k-1 use name n as offset index



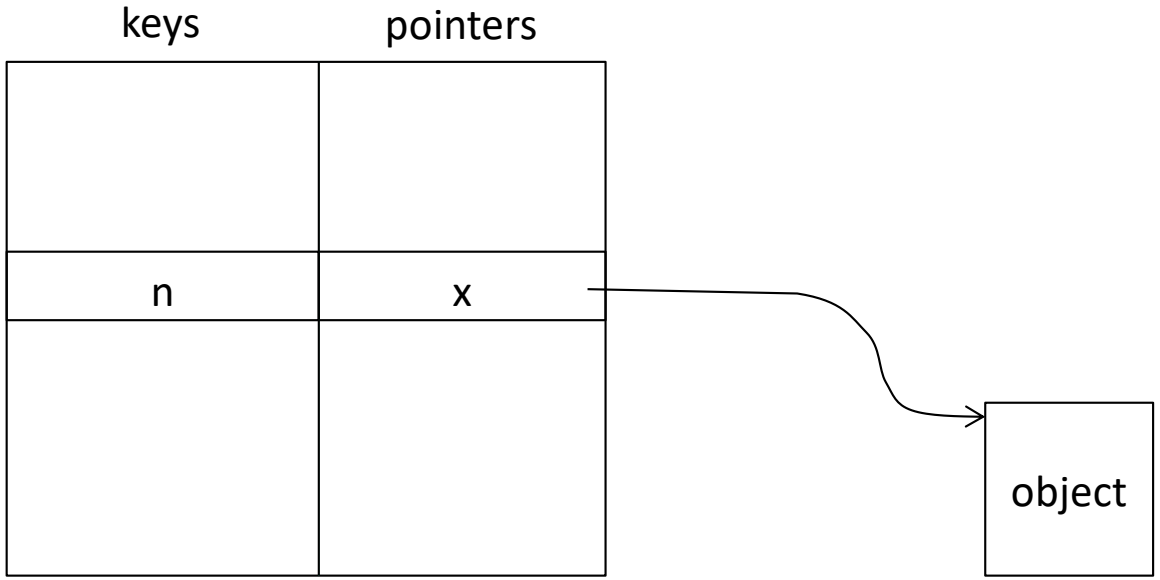
Example: page table: n is page number, x is frame address, object is page frame

© 2022, Peter J Denning

Associative Table

- Used when 2^k too large but number of names is not
- Associative table with one row for each known name
- No duplicate names allowed
- How to efficiently search key column for a match on n?

For small tables, such as directories, linear search is fine

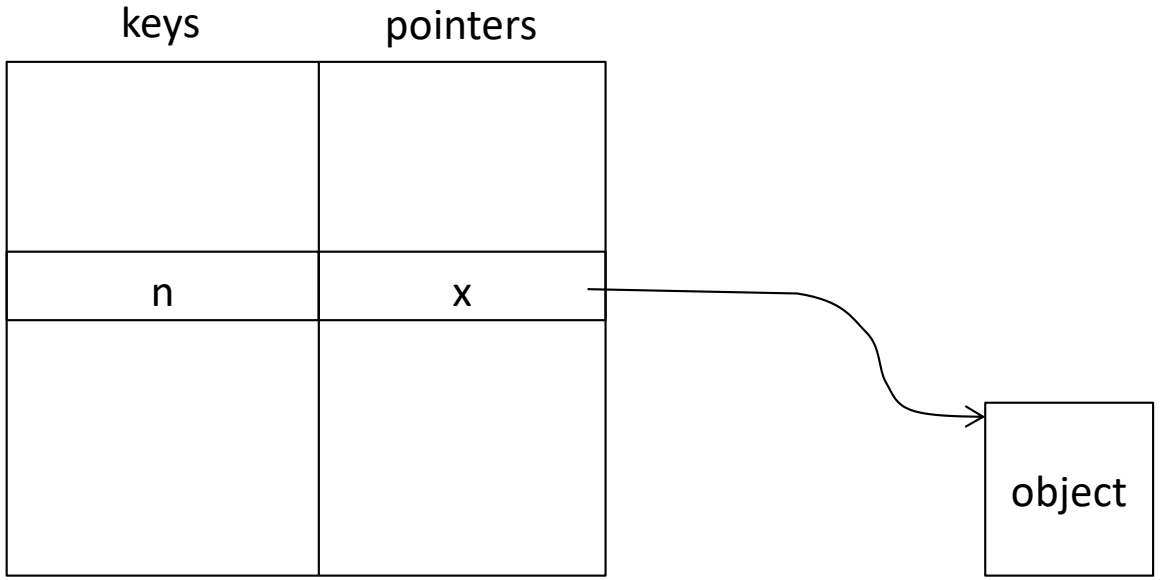


© 2022, Peter J Denning

Associative Table

- Used when 2^k too large but number of names is not
- Associative table with one row for each known name
- No duplicate names allowed
- How to efficiently search key column for a match on N?

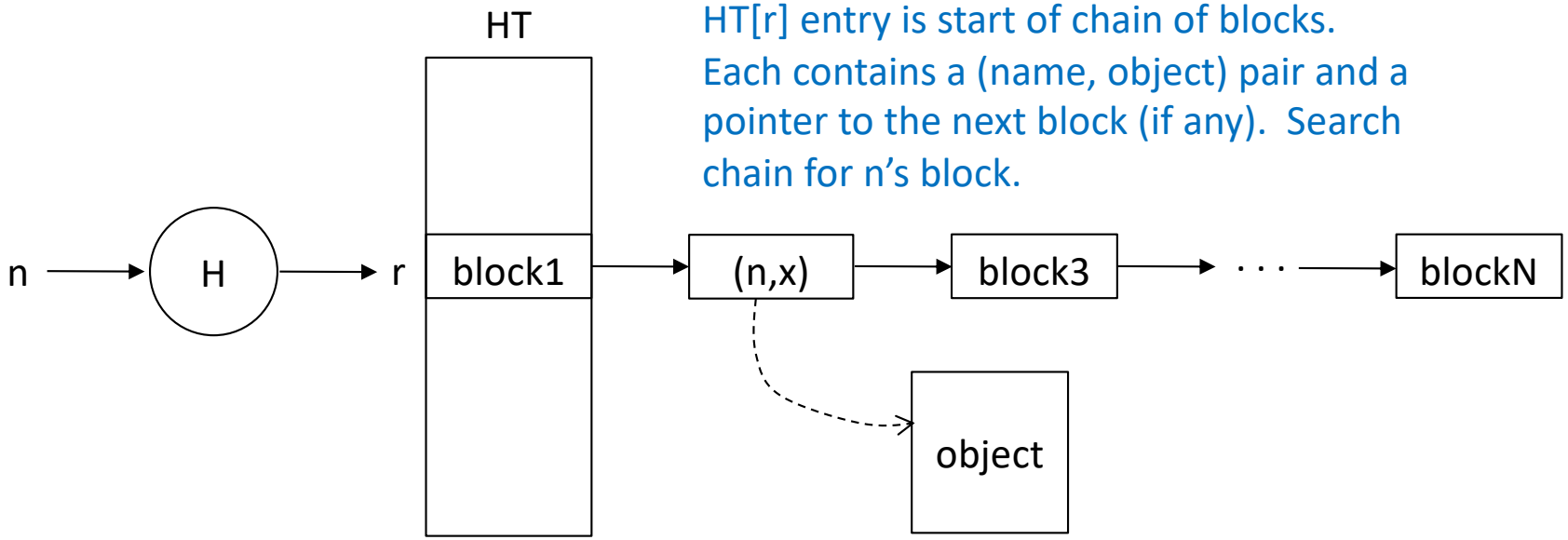
For large tables such as all file handles, linear search would take average of 2^{k-1} comparisons to complete the translation – way too slow



© 2022, Peter J Denning

Hash Table

- Use hash function H to reduce bits of name to an m-bit hash r
- Choose m so that 2^m is manageable
- Hash table is index table with indices 0,1,…,2^m-1
- Each entry a linked list of all items that hash there



© 2022, Peter J Denning

Hash Table - 2

- Set m approx $\log_2$(number of names actually used)
- Then most chains will be 1 or 2 long – search finds the block for n with just 1 or 2 comparisons
- Very good standard hash functions available

© 2022, Peter J Denning

Mapping Accelerators

- A cache can accelerate the mapping process
- A cache is a stored copy of an object (or pointer to) in local fast memory
- Some mapping methods follow long chains
  - Pathnames in directory trees
  - Multiple levels of tables (to be discussed)
- After first access (following the full chain) store the begin and end points in a cache, bypass the chain on future references

© 2022, Peter J Denning

Mapping Accelerators - 2

- Examples
  - TLB in virtual memory: bypass PT lookup and increase average speed by nearly 2x
  - Alias on your desktop
  - Recent file lists in apps
  - Web browser cache of recent web pages

© 2022, Peter J Denning

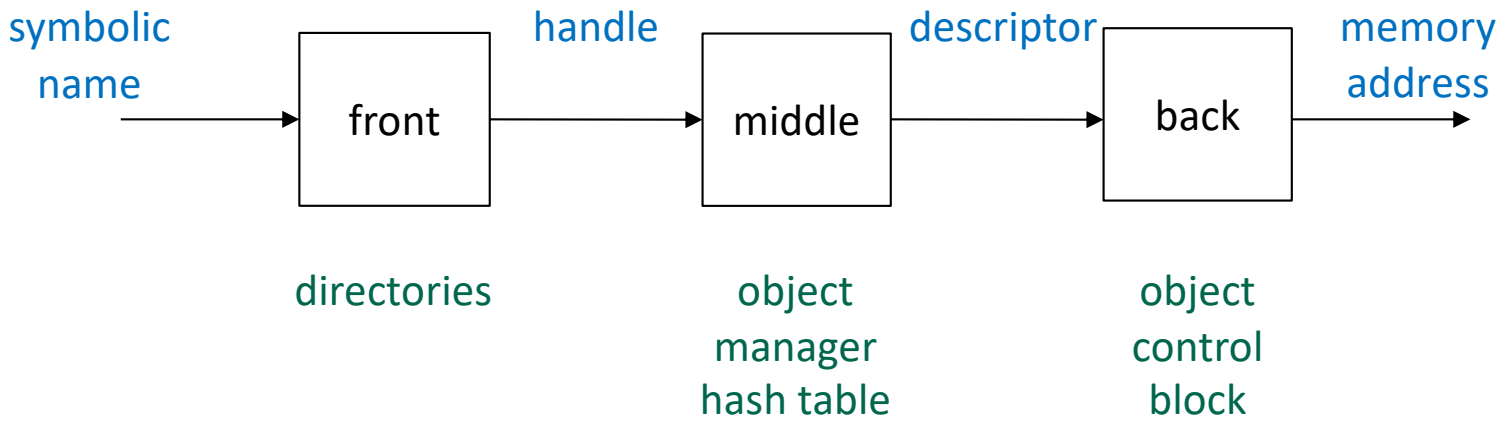
Three kinds of map

- Front end: convert name to handle
  - Intelligibility
- Middle: convert handle to descriptor
  - Location independence
  - Sharing
- Back end: convert descriptor to memory address
  - Memory access

Reminder: handle is OS generated bit string unique for all time (never reused)

© 2022, Peter J Denning

Used Together as Building Blocks

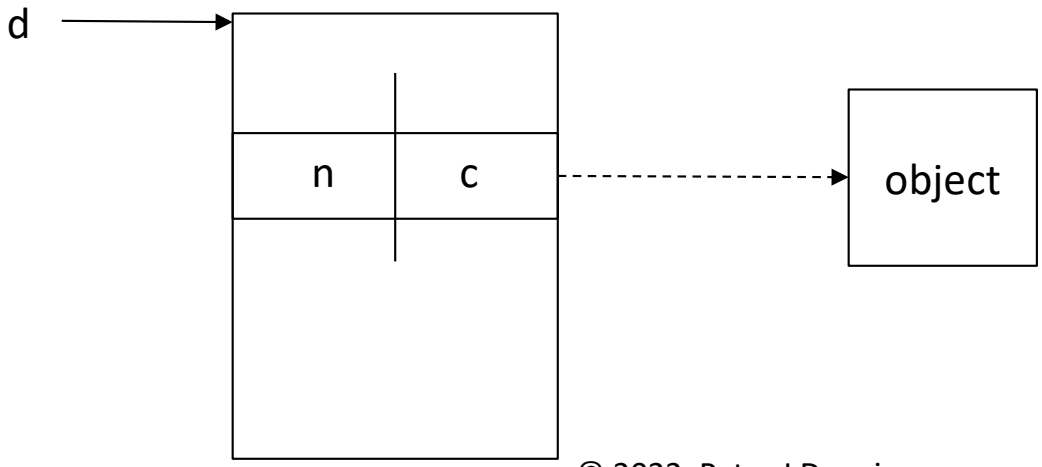


Example:
Map symbolic file name to file handle
then to file control block
then to disk address of the file

© 2022, Peter J Denning

Front End Mappings

- OS example: directories
- Each entry pairs a symbolic name n with a capability c=(t,a,h)=(type,access,handle) pointing to an object
- No duplicate names in a directory



d = directory capability

c = SEARCH(d,n)

c.h points to object, via middle and back end mappings

pathname equivalent to series of directory handles

© 2022, Peter J Denning

Front End Mappings - 2

- Internet example: Domain Name Service (DNS)
- Converts symbolic string hostname to an IP address
  - IP address a handle
  - E.g., nps.edu maps to 205.155.65.226
- DNS lookup part of http protocol

© 2022, Peter J Denning

Front End Mappings - 3

- Web example: URL = hostname/pathname
- http maps hostname to IP with DNS
  - http sends pathname as string to IP
  - That machine resolves the pathname by a series of directory searches in its local directory tree, yielding the handle of the named object
  - Contents of object transmitted back to sender

© 2022, Peter J Denning

Middle Maps

- Convert handle to descriptor
- Object managers
  - Generate unique handles for objects at their creation
  - Maintain internal table mapping handles to descriptors
  - Embed handles inside capabilities before returning them
- User processes (e.g., shell) place new capabilities in directories where they cannot be altered

© 2022, Peter J Denning

Back End Maps

- Extract location information from descriptor
- Descriptor is an “object control block” that contains all information needed to locate object in memory or network
- Single descriptor for each object
- Relocate object? Update descriptor – immediately effective for all processes using object

© 2022, Peter J Denning

Example: virtual memory

- Page numbers are handles
- Page table entries are capabilities with access=access bits, handle=frame-number
- MMU gets frame address from page table entry

© 2022, Peter J Denning

Example: file system

- File identifiers are handles
- File system maps handles to file descriptors
- Extracts file location on disk from descriptor

© 2022, Peter J Denning

Example: directory system

- Directory stores pairs (name, capability)
- Names are keys to directory search
- Search finds entry, returns associated capability
- Capability can then be presented to the object manager for its kind of object

© 2022, Peter J Denning

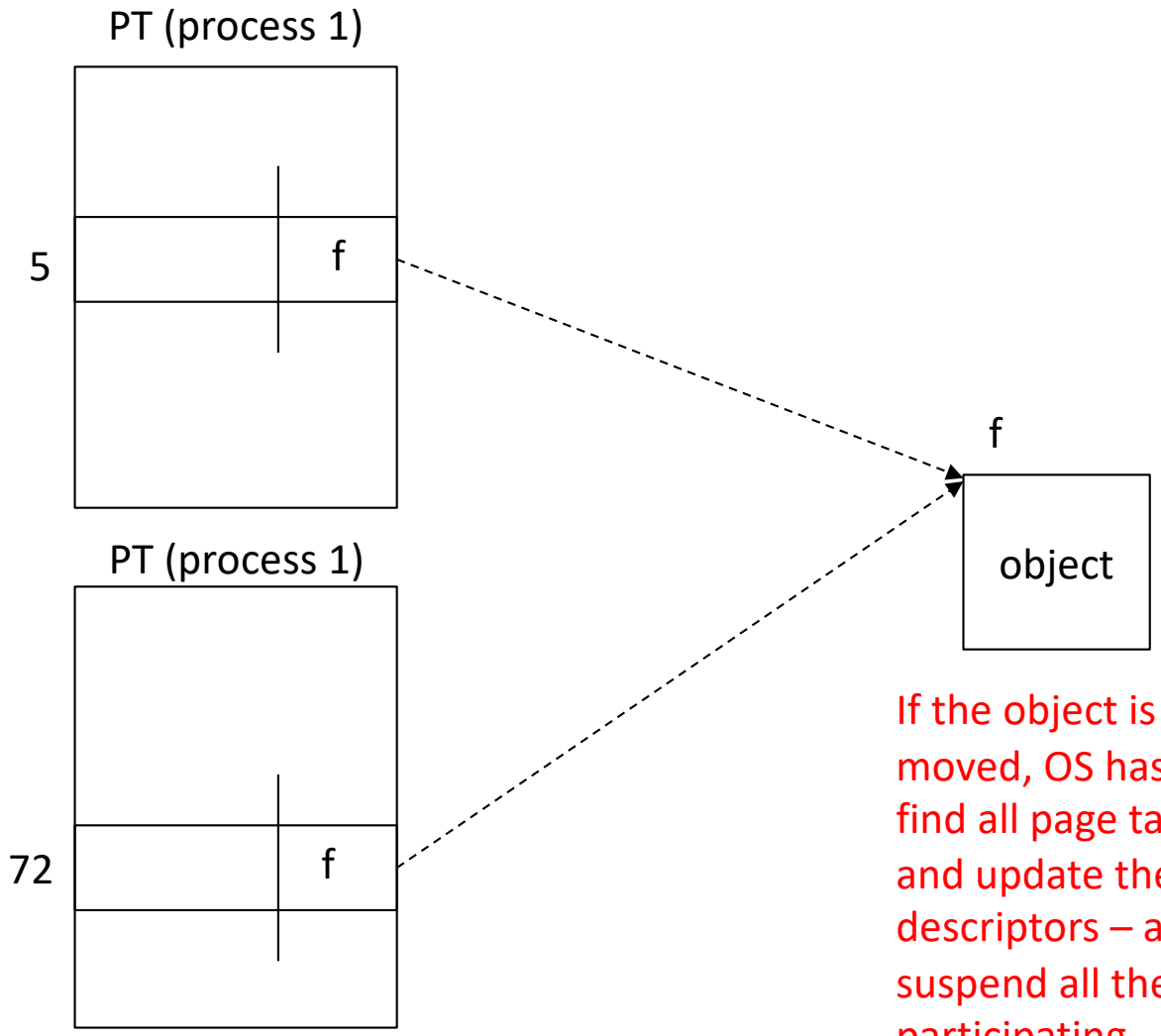
Enabling sharing

- Sharing accomplished by giving capabilities containing the same handle to all processes using the digital object
- Each capability can have its own access code
- All map to the same, single descriptor
- Change the object location or length: update descriptor, then all those sharing see the change immediately
- What goes wrong if you allow multiple copies of a descriptor?

© 2022, Peter J Denning

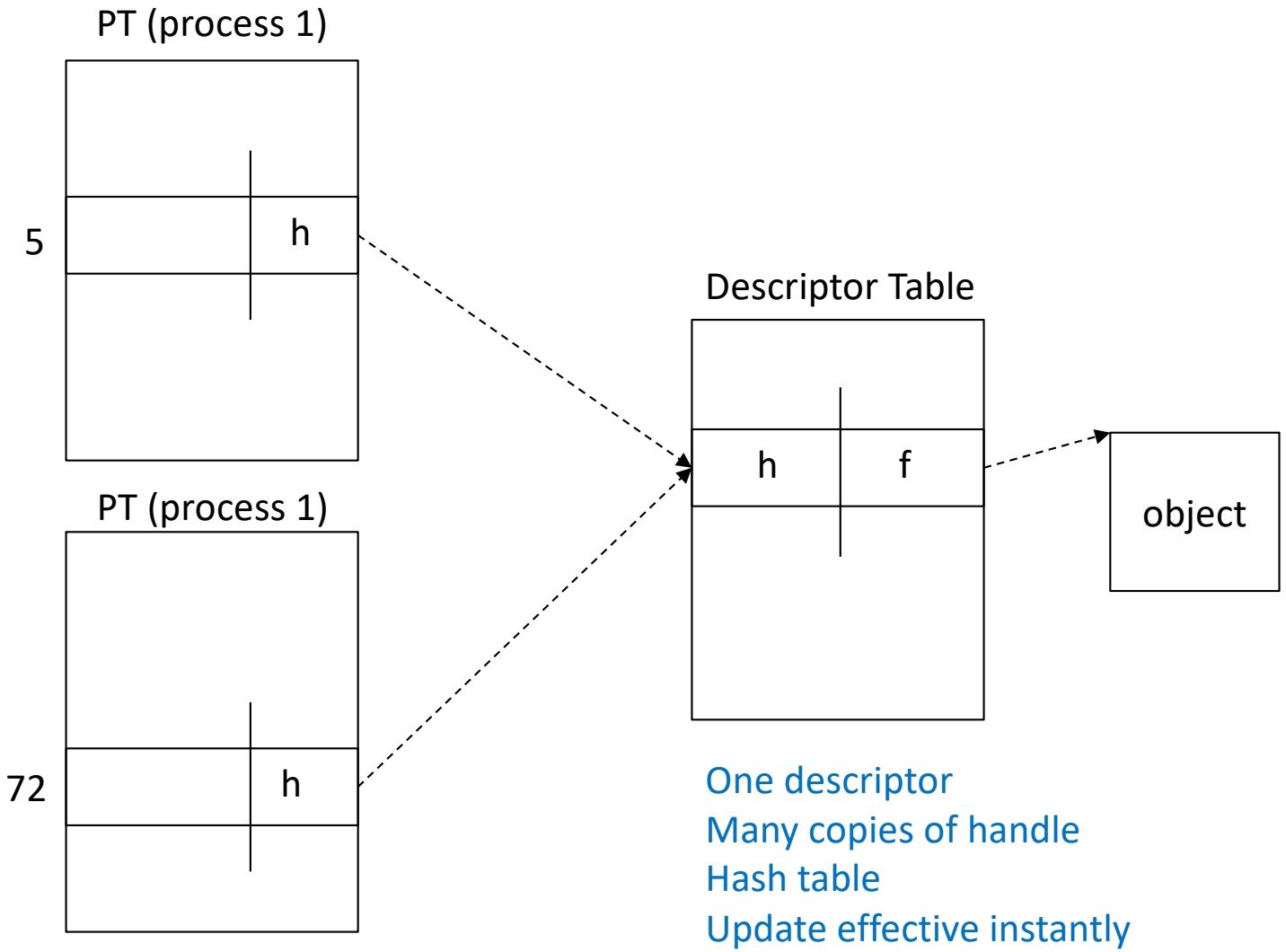
Two processes using different page numbers point to the same object.

The descriptor (here, frame number) is stored in each page table.



If the object is moved, OS has to find all page tables and update their descriptors – and suspend all the participating processes until the update is done.

© 2022, Peter J Denning



One descriptor
Many copies of handle
Hash table
Update effective instantly
Delete DT entry = delete object

Each process caches its recent mapping paths, such as (5,f) or (72,f)

© 2022, Peter J Denning

Summary

- Names and name spaces
- Maps as tables
- Caching mapping paths
- Three levels of maps
  - Front map: string to handle
  - Middle map: handle to descriptor
  - Back map: descriptor to memory
- Handles embedded in capabilities
- Sharing enabled by sharing of capabilities

© 2022, Peter J Denning