

9. Linking and Sharing

- 9.1 Single-Copy Sharing
 - Why Share
 - Requirements for Sharing
- 9.2 Static Linking and Sharing
 - Sharing in Systems w/o Segmentation or Paging
 - Sharing in Paging Systems
 - Sharing in Segmented Systems
- 9.3 Dynamic Linking and Sharing
- 9.4 Principles of DSM
 - The User's View of DSM
- 9.5 Implementations of DSM
 - Implementing Unstructured DSM
 - Implementing Structured DSM

ICS 143

1

Single-copy sharing

- why share
 - processes need to access common data
 - better utilization of memory (code, system tables, data bases)
- requirement for sharing
 - how to express what is shared
 - a priori agreement (e.g., system components)
 - language construct (e.g., shmget/shmat)
 - shared code must be reentrant (read-only)
 - stack, heap must be replicated per process

ICS 143

2

Static linking/sharing

- linking resolves external references
- sharing links to the same module
- static linking/sharing:
 - resolve references before execution starts

ICS 143

3

Sharing w/o segmentation/paging

- with one or no RR:
 - all memory of a process is contiguous
 - share user programs:
 - possible by partial overlap only
 - too restrictive and difficult; generally not used
 - share system components:
 - agree on a starting position
 - linker resolves references to that location

Figure 9-1

ICS 143

4

Sharing w/o segmentation/paging

- with multiple RRs
 - CBR's can point to same copy of code
 - SBR's can point to private copies of stack, etc.

Figure 9-2

ICS 143

5

Sharing in paging systems

- PT entries of different processes point to the same page frame
- data pages: unrestricted
- code pages: must have the same page#'s in all PTs to allow self-references

Figure 9-3

- Note: BR can't be used -- paged system has no concept of function boundaries

ICS 143

6

Sharing in segmented systems

- ST entries of different processes point to the same segment in PM
- data pages: unrestricted
- code pages:
 - assign same segment#'s in all STs, or
 - use BR:
 - function call loads cbr
 - self-references have the form w(CBR)
 - other references have the form (s,w)

ICS 143

7

Dynamic linking/sharing

- basic principles:
 - self-references resolved using CBR
 - external references are indirect via a private linkage section
 - external reference (S,W) is resolved to (s,w) at runtime when first used (S,W are symbolic names)
 - (s,w) is entered in linkage section of process
 - code is unchanged
 - subsequent references use (s,w) without involving OS

ICS 143

8

Dynamic linking/sharing

- compiler setup prior to execution
Figure 9-4(a)
- memory after execution of load instruction
Figure 9-4(b)

ICS 143

9

Distributed shared memory

skip rest of chapter

ICS 143

10
