Virtual Memory

Readings: Chapter 8

Support Needed for Virtual Memory

For virtual memory to be practical and effective:

- Hardware must support paging and segmentation
- Operating system must include software for managing the movement of pages and/or segments between secondary memory and main memory

 Principle of Locality

- Program and data references within a process tend to cluster
- Only a few pieces of a process will be needed over a short period of time
- Therefore it is possible to make intelligent guesses about which pieces will be needed in the future
- Avoids thrashing

© 2017 Pearson Education, Inc., Hoboken, NJ. All rights reserved.

© 2017 Pearson Education, Inc., Hoboken, NJ. All rights reserved.
Inverted Page Table

- Page number portion of a virtual address is mapped into a hash value
  - Hash value points to inverted page table

- Fixed proportion of real memory is required for the tables regardless of the number of processes or virtual pages supported

- Structure is called inverted because it indexes page table entries by frame number rather than by virtual page number
Translation Lookaside Buffer (TLB)

• To overcome the effect of doubling the memory access time, most virtual memory schemes make use of a special high-speed cache called a translation lookaside buffer (TLB)
  • This cache functions in the same way as a memory cache and contains those page table entities that have been most recently used

• Each virtual memory reference can cause two physical memory accesses:
  • One to fetch the page table entry
  • One to fetch the data

Each segment table entry contains the starting address of the corresponding segment in main memory and the length of the segment

• A bit is needed to determine if the segment is already in main memory

• Another bit is needed to determine if the segment has been modified since it was loaded in main memory
In a combined paging/segmentation system a user’s address space is broken up into a number of segments. Each segment is broken up into a number of fixed-sized pages which are equal in length to a main memory frame.

Segmentation is visible to the programmer
Paging is transparent to the programmer

The design of the memory management portion of an operating system depends on three fundamental areas of choice:

- Whether or not to use virtual memory techniques
- The use of paging or segmentation or both
- The algorithms employed for various aspects of memory management
1- Fetch Policy

- Determines when a page should be brought into memory

![Diagram of Two main types: Demand Paging and Prepaging]

2- Placement Policy

- Determines where in real memory a process piece is to reside

- Important design issue in a segmentation system

- Paging or combined paging with segmentation placing is irrelevant because hardware performs functions with equal efficiency

- For NUMA systems an automatic placement strategy is desirable

3- Replacement Policy

- Deals with the selection of a page in main memory to be replaced when a new page must be brought in

- Objective is that the page that is removed be the page least likely to be referenced in the near future

- The more elaborate the replacement policy the greater the hardware and software overhead to implement it

Frame Locking

- When a frame is locked the page currently stored in that frame may not be replaced

- Kernel of the OS as well as key control structures are held in locked frames

- I/O buffers and time-critical areas may be locked into main memory frames

- Locking is achieved by associating a lock bit with each frame
**Least Recently Used (LRU)**

- Replaces the page that has not been referenced for the longest time.
- By the principle of locality, this should be the page least likely to be referenced in the near future.
- Difficult to implement
  - One approach is to tag each page with the time of last reference
  - This requires a great deal of overhead.

**First-in-First-out (FIFO)**

- Treats page frames allocated to a process as a circular buffer.
- Pages are removed in round-robin style
  - Simple replacement policy to implement
- Page that has been in memory the longest is replaced.

**Clock Policy**

- Requires the association of an additional bit with each frame
  - Referred to as the use bit
- When a page is first loaded in memory or referenced, the use bit is set to 1
- The set of frames is considered to be a circular buffer
- Any frame with a use bit of 1 is passed over by the algorithm
- Page frames visualized as laid out in a circle.
Page Buffering

- Improves paging performance and allows the use of a simpler page replacement policy

![Diagram of page buffer lists](image1)

**Free page list**
- List of page frames available for reading in pages

**Modified page list**
- Pages are written out in clusters

A replaced page is not lost, but rather assigned to one of two lists

Figure 8.15 Example of Clock Policy Operation

© 2017 Pearson Education, Inc., Hoboken, NJ. All rights reserved.

Replacement Policy and Cache Size

- With large caches, replacement of pages can have a performance impact
  - If the page frame selected for replacement is in the cache, that cache block is lost as well as the page that it holds
  - In systems using page buffering, cache performance can be improved with a policy for page placement in the page buffer
  - Most operating systems place pages by selecting an arbitrary page frame from the page buffer

![Comparison of fixed-allocation, local page replacement algorithms](image2)

Figure 8.16 Comparison of Fixed-Allocation, Local Page Replacement Algorithms

© 2017 Pearson Education, Inc., Hoboken, NJ. All rights reserved.
4- Resident Set Management

• The OS must decide how many pages to bring into main memory

• The smaller the amount of memory allocated to each process, the more processes can reside in memory
  • Small number of pages loaded increases page faults

• Beyond a certain size, further allocations of pages will not effect the page fault rate

Resident Set Size

Fixed-allocation

• Gives a process a fixed number of frames in main memory within which to execute
  • When a page fault occurs, one of the pages of that process must be replaced

Variable-allocation

• Allows the number of page frames allocated to a process to be varied over the lifetime of the process

Replacement Scope

• The scope of a replacement strategy can be categorized as global or local
  • Both types are activated by a page fault when there are no free page frames

Local

• Chooses only among the resident pages of the process that generated the page fault

Global

• Considers all unlocked pages in main memory

5- Cleaning Policy

• Concerned with determining when a modified page should be written out to secondary memory

Demand Cleaning

A page is written out to secondary memory only when it has been selected for replacement

Precleaning

Allows the writing of pages in batches
6- Load Control

- Determines the number of processes that will be resident in main memory
  - Multiprogramming level

- Critical in effective memory management

- Too few processes, many occasions when all processes will be blocked and much time will be spent in swapping

- Too many processes will lead to thrashing

© 2017 Pearson Education, Inc., Hoboken, N.J. All rights reserved.

Process Suspension

- If the degree of multiprogramming is to be reduced, one or more of the currently resident processes must be swapped out

- Lowest-priority process
- Faulting process
- Last process activated
- Process with the smallest resident set
- Largest process
- Process with the largest remaining execution window

Six possibilities exist: