Basic Algorithms

- Use “use” and “modify” bits
  1. Scan for first frame with u=0, m=0
  2. If 1) fails look for frame with u=0, m=1, setting the use bits to 0 during scan
  3. If 2) failed repeating 1) and 2) will find a replacement

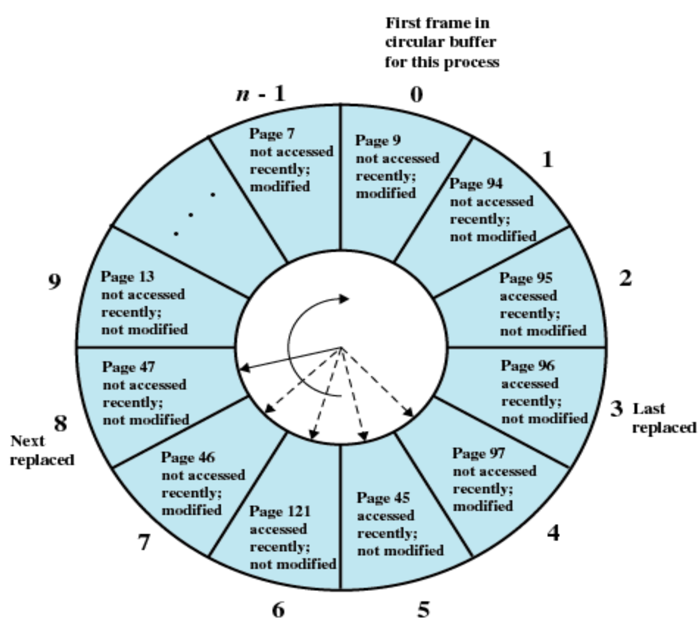

Figure 8.18 The Clock Page-Replacement Algorithm [GOLD89]

Resident Set Size

- Fixed-allocation
  - Gives a process a fixed number of pages within which to execute
  - When a page fault occurs, one of the pages of that process must be replaced
- Variable-allocation
  - Number of pages allocated to a process varies over the lifetime of the process

Fixed Allocation, Local Scope

- Decide ahead of time the amount of allocation to give a process
  - If allocation is too small, there will be a high page fault rate
  - If allocation is too large there will be too few programs in main memory

Variable Allocation, Global Scope

- Easiest to implement
- Adopted by many operating systems
- Operating system keeps list of free frames
- Free frame is added to resident set of process when a page fault occurs
- If no free frame, replaces one from another process

Variable Allocation, Local Scope

- When new process added, allocate number of page frames based on application type, program request, or other criteria
- When page fault occurs, select page from among the resident set of the process that suffers the fault
- Reevaluate allocation from time to time

Cleaning Policy

- Demand cleaning
  - A page is written out only when it has been selected for replacement
- Precleaning
  - Pages are written out in batches

Cleaning Policy

- Best approach uses page buffering
  - Replaced pages are placed in two lists
    - Modified and unmodified
  - Pages in the modified list are periodically written out in batches
    - What is the motivation behind this strategy?
  - Pages in the unmodified list are either reclaimed if referenced again or lost when its frame is assigned to another page

Load Control

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

Multiprogramming

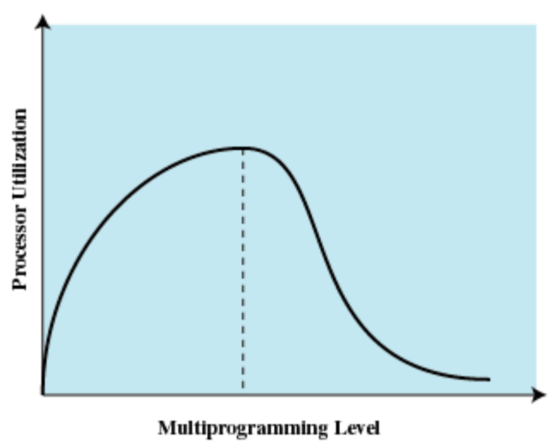

Figure 8.21 Multiprogramming Effects

Process Suspension

- If degree of multiprogramming is to be reduced, suspend:
  - Lowest priority process
  - Faulting process
    - This process does not have its working set in main memory so it will be blocked anyway
  - Last process activated
    - This process is least likely to have its working set resident

Process Suspension cont.

- Process with smallest resident set
  - This process requires the least future effort to reload
- Largest process
  - Obtains the most free frames
- Process with the largest remaining execution window

UNIX and Solaris Memory Management

- Paging System
  - Page table
  - Disk block descriptor
  - Page frame data table
  - Swap-use table

Table 8.5 UNIX SVR4 Memory Management Parameters (page 1 of 2)

Page Table Entry

Page frame number
Refers to frame in real memory.

Age
Indicates how long the page has been in memory without being referenced. The length and contents of this field are processor dependent.

Copy on write
Set when more than one process shares a page. If one of the processes writes into the page, a separate copy of the page must first be made for all other processes that share the page. This feature allows the copy operation to be deferred until necessary and avoided in cases where it turns out not to be necessary.

Modify
Indicates page has been modified.

Reference
Indicates page has been referenced. This bit is set to zero when the page is first loaded and may be periodically reset by the page replacement algorithm.

Valid
Indicates page is in main memory.

Protect
Indicates whether write operation is allowed.

Disk Block Descriptor

Swap device number
Logical device number of the secondary device that holds the corresponding page. This allows more than one device to be used for swapping.

Device block number
Block location of page on swap device.

Type of storage
Storage may be swap unit or executable file. In the latter case, there is an indication as to whether the virtual memory to be allocated should be cleared first.

Table 8.5 UNIX SVR4 Memory Management Parameters (page 2 of 2)

Page Frame Data Table Entry

Page State
Indicates whether this frame is available or has an associated page. In the latter case, the status of the page is specified: on swap device, in executable file, or DMA in progress.

Reference count
Number of processes that reference the page.

Logical device
Logical device that contains a copy of the page.

Block number
Block location of the page copy on the logical device.

Pfdata pointer
Pointer to other pfdata table entries on a list of free pages and on a hash queue of pages.

Swap-use Table Entry

Reference count
Number of page table entries that point to a page on the swap device.

Page/storage unit number
Page identifier on storage unit.

Page frame number	Age	Copy on write	Modify	Reference	Valid	Protect

(a) Page table entry

Swap device number	Device block number	Type of storage

(b) Disk block descriptor

Page state	Reference count	Logical device	Block number	Pfdata pointer

(c) Page frame data table entry

Reference count	Page/storage unit number

(d) Swap-use table entry

Figure 8.22 UNIX SVR4 Memory Management Formats

UNIX and Solaris Memory Management

- Page Replacement
  - Refinement of the clock policy

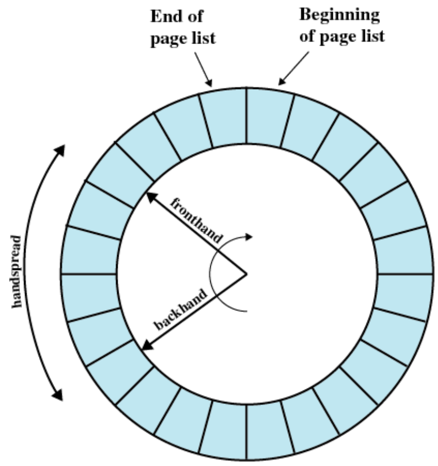

Figure 8.23 Two-Handed Clock Page-Replacement Algorithm

Kernel Memory Allocator

- Lazy buddy system

Initial value of D_i is 0
After an operation, the value of D_i is updated as follows

(I) if the next operation is a block allocate request:
    if there is any free block, select one to allocate
      if the selected block is locally free
          then $D_i := D_i + 2$
          else $D_i := D_i + 1$
    otherwise
      first get two blocks by splitting a larger one into two (recursive operation)
      allocate one and mark the other locally free
      D_i remains unchanged (but D may change for other block sizes because of the recursive call)

(II) if the next operation is a block free request
    Case $D_i \geq 2$
      mark it locally free and free it locally
      $D_i := D_i - 2$
    Case $D_i = 1$
      mark it globally free and free it globally; coalesce if possible
      $D_i := 0$
    Case $D_i = 0$
      mark it globally free and free it globally; coalesce if possible
      select one locally free block of size 2i and free it globally; coalesce if possible
      $D_i := 0$

Figure 8.24 Lazy Buddy System Algorithm

Linux Memory Management

- Page directory
- Page middle directory
- Page table

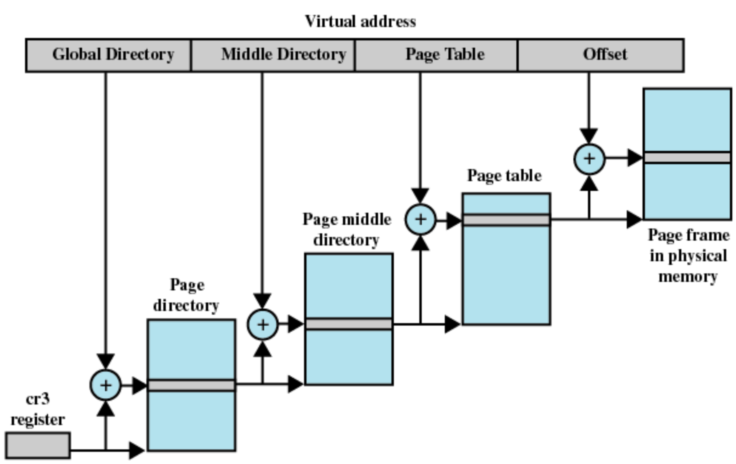

Figure 8.25 Address Translation in Linux Virtual Memory Scheme