Virtual memory

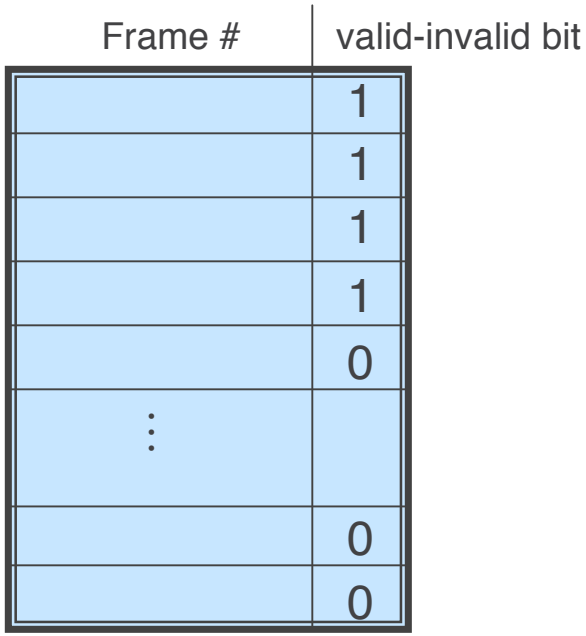
- separation of user logical memory from physical memory
  - Only part of the program needs to be in memory for execution.
  - Logical address space can therefore be much larger than physical address space.
  - Allows address spaces to be shared by several processes.
  - Allows for more efficient process creation

Demand Paging

- Bring a page into memory only when it is needed.
  - Less I/O needed
  - Less memory needed
  - Faster response
  - More users
- Page is needed ⇒ reference to it
  - invalid reference ⇒ abort
  - not-in-memory ⇒ bring to memory

Valid-Invalid Bit

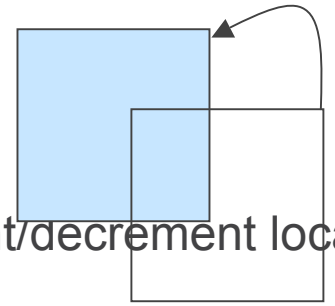
- With each page table entry a valid–invalid bit is associated
  (1 ⇒ in-memory, 0 ⇒ not-in-memory)
- Initially valid–invalid but is set to 0 on all entries.
- Example of a page table snapshot.



- During address translation, if valid–invalid bit in page table entry is 0 ⇒ page fault.

Page Fault

- If there is ever a reference to a page, first reference will trap to
  OS ⇒ page fault
- OS looks at another table to decide:
  - Invalid reference ⇒ abort.
  - Just not in memory.
- Get empty frame.
- Swap page into frame.
- Reset tables, validation bit = 1.
- Restart instruction: Least Recently Used
  - block move



  - auto increment/decrement location

What happens if there is no free frame?

- Page replacement – find some page in memory, but not really in use, swap it out
  - performance – want an algorithm which will result in minimum number of page faults.
- Same page may be brought into memory several times.
- Page Fault Rate $0 \leq p \leq 1.0$
  - if $p = 0$ no page faults
  - if $p = 1$, every reference is a fault

Process Creation

- Virtual memory allows other benefits during process creation:
  - Copy-on-Write
    - Copy-on-Write (COW) allows both parent and child processes to initially *share* the same pages in memory.
    - If either process modifies a shared page, only then is the page copied. COW allows more efficient process creation as only modified pages are copied
  - Memory-Mapped Files
    - Memory-mapped file I/O allows file I/O to be treated as routine memory access by mapping a disk block to a page in memory
    - Simplifies file access by treating file I/O through memory rather than read() write() system calls
    - Also allows several processes to map the same file allowing the pages in memory to be shared

Page Replacement

- Prevent over-allocation of memory by modifying page-fault service routine to include page replacement.
- Use *modify* (*dirty*) *bit* to reduce overhead of page transfers – only modified pages are written to disk.
- Page replacement completes separation between logical memory and physical memory – large virtual memory can be provided on a smaller physical memory.

Basic Page Replacement

- Find the location of the desired page on disk.

- Find a free frame:
  - If there is a free frame, use it.
  - If there is no free frame, use a page replacement algorithm to select a *victim* frame.

- Read the desired page into the (newly) free frame. Update the page and frame tables.

- Restart the process.

Page Replacement Algorithms

- Want lowest page-fault rate.
- Evaluate algorithm by running it on a particular string of memory references (reference string) and computing the number of page faults on that string.
- In all our examples, the reference string is

  1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5.

First-In-First-Out (FIFO) Algorithm

- Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- 3 frames (3 pages can be in memory at a time per process)

1	1	4	5	
2	2	1	3	9 page faults
3	3	2	4	

- 4 frames

1	1	5	4	
2	2	1	5	10 page faults
3	3	2		
4	4	3		

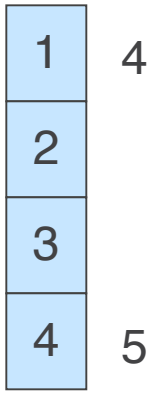
- FIFO Replacement – Belady's Anomaly
  - more frames ⇒ less page faults

FIFO Illustrating Belady's Anomaly

Optimal Algorithm

- Replace page that will not be used for longest period of time.
- 4 frames example

1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

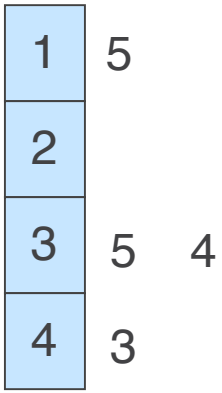


6 page faults

- How do you know this?
- Used for measuring how well your algorithm performs.

Least Recently Used (LRU) Algorithm

- Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5



- Counter implementation
  - Every page entry has a counter; every time page is referenced through this entry, copy the clock into the counter.
  - When a page needs to be changed, look at the counters to determine which are to change.

LRU Algorithm (Cont.)

- Stack implementation – keep a stack of page numbers in a double link form:
  - Page referenced:
    - move it to the top
    - requires 6 pointers to be changed
  - No search for replacement

LRU Approximation Algorithms

- Reference bit
  - With each page associate a bit, initially = 0
  - When page is referenced bit set to 1.
  - Replace the one which is 0 (if one exists). We do not know the order, however.
- Second chance
  - Need reference bit.
  - Clock replacement.
  - If page to be replaced (in clock order) has reference bit = 1. then:
    - set reference bit 0.
    - leave page in memory.
    - replace next page (in clock order), subject to same rules.

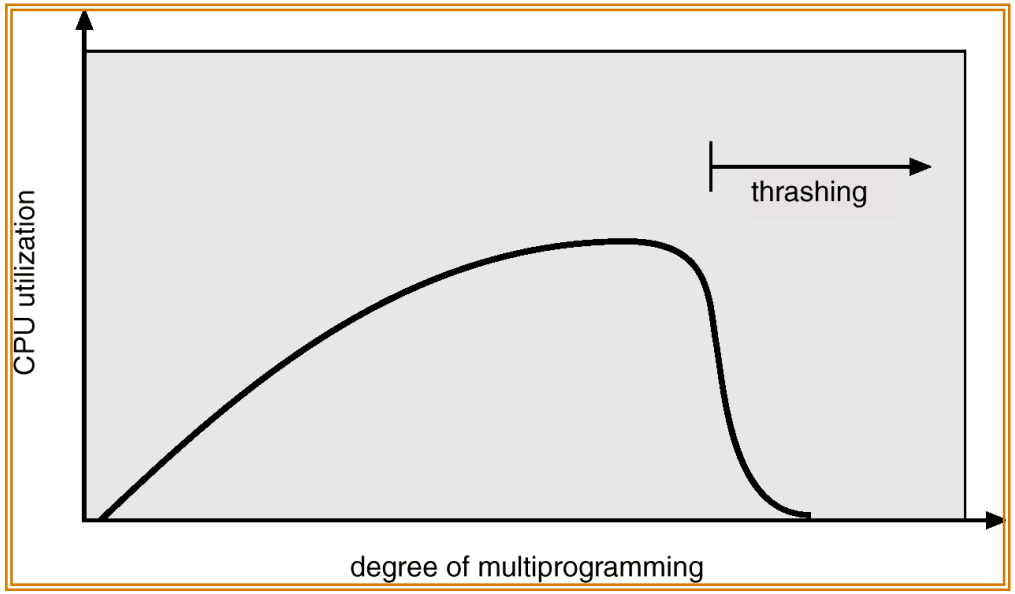
Counting Algorithms

- Keep a counter of the number of references that have been made to each page.
- LFU Algorithm: replaces page with smallest count.
- MFU Algorithm: based on the argument that the page with the smallest count was probably just brought in and has yet to be used.

Thrashing

- If a process does not have “enough” pages, the page-fault rate is very high. This leads to:
  - low CPU utilization.
  - operating system thinks that it needs to increase the degree of multiprogramming.
  - another process added to the system.
- **Thrashing** ≡ a process is busy swapping pages in and out.

Thrashing



- Why does paging work?
  Locality model
  - Process migrates from one locality to another.
  - Localities may overlap.
- Why does thrashing occur?
  Σ size of locality > total memory size

Demand Segmentation

- Used when insufficient hardware to implement demand paging.
- OS/2 allocates memory in segments, which it keeps track of through segment descriptors
- Segment descriptor contains a valid bit to indicate whether the segment is currently in memory.
  - If segment is in main memory, access continues,
  - If not in memory, segment fault.

Outline

- Chapter 18: Protection
- Chapter 19: Security

Protection

- Protect computer resources from being accessed by processes that should not have access
  - Access right: Operations allowed on an object
  - Domain: Set of all access rights
- UNIX: domain is userid, setuid bit in file switches domains
- Multics: rings, tasks can get access based on entry points
- Access Matrix defines protection: rows represent domains & columns represent objects
  - Global table
  - Access list for objects: easier to program
  - Capability list for domains/users:
  - Hybrid: lock-key mechanism
- Revocation of rights:
  - Immediate vs delayed, selective vs general, partial vs total, temporary vs permanent

Revocation

- *Access List* – Delete access rights from access list.
  - Simple
  - Immediate
- *Capability List* – Scheme required to locate capability in the system before capability can be revoked.
  - Reacquisition
  - Back-pointers
  - Indirection
  - Keys

Compiler/language based mechanism

- Compiler based enforcement
  - Specification of protection in a programming language allows the high-level description of policies for the allocation and use of resources
- Java VM
  - Multiple threads within a single JVM have different access rights
- A class is assigned a protection domain when it is loaded by the JVM. The protection domain indicates what operations the class can (and cannot) perform
  - Protection enforced using stack inspection

Security

- Security problem: protection from unauthorized access, malicious modification or destruction
- User authentication:
  - Passwords
    - Encrypted passwords
      - Encrypted form should be secret because attacker can check offline
    - One-time passwords
    - Biometrics
- Threats:
  - Trojan horse
  - Trap door/stack and buffer overflow
  - Worms/viruses
  - Denial of service
  - Intrusion and detection

Risk analysis

- Important to understand threat and perform risk analysis
  - No system is “secure”, systems usually trade security for performance, ease of use etc.
  - If information is worth **x** and it costs **y** to break into system and if (**x** < **y**), then not worth encryption
  - Wasteful to build a system that is more secure than is necessary
    - Ssh in CSE dept – good
  - Palm pilots may not require powerful encryption systems if they are expected to be physically secure

Security classification

- U.S. Department of Defense outlines four divisions of computer security: A, B, C, and D
  - D – Minimal security
  - C – Provides discretionary protection through auditing. Divided into C1 and C2. C1 identifies cooperating users with the same level of protection. C2 allows user-level access control
  - B – All the properties of C, however each object may have unique sensitivity labels. Divided into B1, B2, and B3
  - A – Uses formal design and verification techniques to ensure security
- Windows NT: Configurable security from D to C2
- SuSE Linux Enterprise Server 8 on IBM eServer xSeries - Evaluation Assurance Level 2+ certification (EAL2)
- http://www.radium.ncsc.mil/tpep/epl/historical.html

Security Attacks

- Social engineering attacks
  - Preys on people gullibility (good nature), hardest to defend
    - E.g. I once got an unlisted number from a telephone operator because I sounded desperate (I was, but that was not the point)
    - E.g. Anna kour*va virus, Nigerian email scam, MS update scam
    - E.g. If I walk in with coupla heavy looking boxes into the elevator to go to Fitz 3rd floor (at night) would you let me in? You can get into “secure” companies by looking like you “belong” there
- Denial of service attacks
  - Network flooding, Distributed DOS, holding resources, viruses

Common technology - firewalls

- Firewalls are used to restrict the kinds of network traffic in/out of companies
  - Application level proxies
  - Packet level firewalls
- Does not prevent end-to-end security violations
  - People sometimes email list of internal computer users outside firewall to scrupulous “researchers”
  - Emails viruses exploit certain vulnerabilities in VBS to get around firewalls

Intrusion detection

- Detect attempts to intrude into computer systems.
- Detection methods:
  - Auditing and logging
  - Tripwire (UNIX software that checks if certain files and directories have been altered – I.e. password files)
- System call monitoring