Memory Management

Disclaimer: some slides are adopted from book authors’ slides with permission
Recap: Two Level Paging

Virtual address

1\textsuperscript{st} level \hspace{1cm} 2\textsuperscript{nd} level \hspace{1cm} offset

Base ptr

1\textsuperscript{st} level Page table

2\textsuperscript{nd} level Page

Physical address

Frame # \hspace{1cm} Offset
Quiz

• What is the page table size for a process that only uses 8MB memory?
  – Common: 32bit address space, 4KB page size
  – Case 1) 1-level page table
    • Assume each page table entry is 4 bytes
  
  – Case 2) two-level page table
    • Assume first 10 bits are used as the index of the first-level page table, next 10 bits are used as the index of the second-level page table. In both-levels, single page table entry size is 4 bytes
Quiz

• What is the page table size for a process that only uses 8MB memory?
  – Common: 32bit address space, 4KB page size
  – Case 1) 1-level page table
    • Assume each page table entry is 4 bytes
    • Answer: $2^{20} \times 4$ byte = 4MB

  – Case 2) two-level page table
    • Assume first 10 bits are used as the index of the first-level page table, next 10 bits are used as the index of the second-level page table. In both levels, single page table entry size is 4 bytes
    • Answer: $2^{10} \times 4 + 2 \times (2^{10} \times 4) = 4$KB + 8KB = 12KB
Recap: Demand Paging

- Idea: instead of keeping the entire memory pages in memory all the time, keep only part of them on a on-demand basis
Recap: Page Fault Handling

1. Load M
2. Trap
3. Page is on backing store
4. Bring in missing page
5. Reset page table
6. Restart instruction
Recap: Starting Up a Process

![Diagram showing memory regions: Code, Data, Heap, Stack, with unmapped pages indicating areas not currently in use.]

- **Stack**
- **Heap**
- **Data**
- **Code**
Recap: Starting Up a Process

Over time, more pages are mapped as needed
Anonymous Page

• An executable file contains code (binary)
  – So we can read from the executable file

• What about heap?
  – No backing storage (unless it is swapped out later)
  – Simply map a new free page (anonymous page) into the address space
Multiple instances of the same program

- E.g., 10 bash shells
Concepts to Learn

• Page replacement/swapping
• Thrashing
Memory Size Limit?

- Demand paging → illusion of infinite memory

![Diagram showing memory allocation and paging]

- Process A: 4GB (stack, heap, data, text)
- Process B: 4GB (stack, heap, data, text)
- Process C: 4GB (stack, heap, data, text)

Physical Memory: 500GB

Disk: 500GB

TLB, MMU, Page Table
Illusion of Infinite Memory

• Demanding paging
  – Allows more memory to be allocated than the size of physical memory
  – Uses memory as cache of disk

• What to do when memory is full?
  – On a page fault, there’s no free page frame
  – Someone (page) must go (be evicted)
Recap: Page Fault

• On a page fault
  – Step 1: allocate a free page frame
  – Step 2: bring the stored page on disk (if necessary)
  – Step 3: update the PTE (mapping and valid bit)
  – Step 4: restart the instruction
Page Replacement Procedure

• On a page fault
  – Step 1: allocate a free page frame
    • If there’s a free frame, use it
    • If there’s no free frame, choose a **victim frame** and evict it to disk (if necessary) \(\rightarrow\) **swap-out**
  – Step 2: bring the stored page on disk (if necessary)
  – Step 3: update the PTE (mapping and valid bit)
  – Step 4: restart the instruction
Page Replacement Procedure
Page Replacement Policy

• Which page (a.k.a. victim page) to go?
  – What if the evicted page is needed soon?
    • A page fault occurs, and the page will be re-loaded
  – Important decision for performance reason
    • The cost of choosing wrong page is very high: disk accesses
Page Replacement Policies

• FIFO (First In, First Out)
  – Evict the oldest page first.
  – Pros: fair
  – Cons: can throw out frequently used pages

• Optimal
  – Evict the page that will not be used for the longest period
  – Pros: optimal
  – Cons: you need to know the future
Page Replacement Policies

• Random
  – Randomly choose a page
  – Pros: simple. TLB commonly uses this method
  – Cons: unpredictable

• LRU (Least Recently Used)
  – Look at the past history, choose the one that has not been used for the longest period
  – Pros: good performance
  – Cons: complex, requires h/w support