Operating System: Virtual Memory

Alex Chi

Update: April 13, 2020

Contents

1 Background

- not all data need to be loaded into memory
- use virtual memory

2 Virtual Memory

- separation of user logical memory from physical memory

3 Demand Paging

- don't bring page into memory until access

3.1 Q&A

1. Why less I/O?
   - We just need to load part of the code into memory.
2. Is it possible that module for handling page fault isn't in memory?
   - Nope. It will always stay in memory.

4 Page Management

4.1 Page Table Entry

- what does a page table entry look like?
  - example: RISC-V Sv39, physical page + valid, user, write, read, execute, reserved bits
- handle page fault
  - trap
  - OS handle
  - bring page to physical memory
  - reset page table
  - restart instruction

4.2 Page Replacement Algorithm

- use dirty bit

4.3 FIFO Page Replacement

- worst case: 0 1 2 3 0 1 2 3 (with 3 cache)

4.4 Optimal Page Replacement

- replace page that won't be used for longest time
- must know reference string in advance

4.5 Least Recently Used

- use past knowledge rather than future
- replace page that has not been used in most amount of time
- implementation: linked list

4.6 LRU Approximation

- reference bit (do not specify order)
- second-change algorithm

4.7 Counting Algorithms

- least frequently used
- most frequently used

4.8 Demanding Paging

- refer to example in slides