Project 3-2: Virtual Memory

Prof. Jin-Soo Kim (jinsookim@skku.edu)
Computer Systems Laboratory
Sungkyunkwan University
http://csl.skku.edu

SKK UNIVERSITY

Project 3-1 Review

- **Do not use palloc at mmap system call**
  - All page allocations for user space (except first stack page) should be done by page fault
- **Supplemental page table**
  - Pointing mmap/ swap entry at constant time
  - Freeing entries which are invalid in original pte

Project 3

- **Project 3-1 (50%)**
  - Project 2 (30%)
  - Project 3-1 (70%)
- **Project 3-2 (50%)**
  - Project 2 + Code/Data Lazy Loading (30%)
  - Test case (70%)
- **Bonus point**
  - Test time (tick, page-xxxxx)
  - Swap count (write, read)

Project 3 Overview

- **Requirements -> Paging**
  - Code data loading
  - Page replacement (Swapping)
  - Dynamic stack growth
  - Memory mapped files

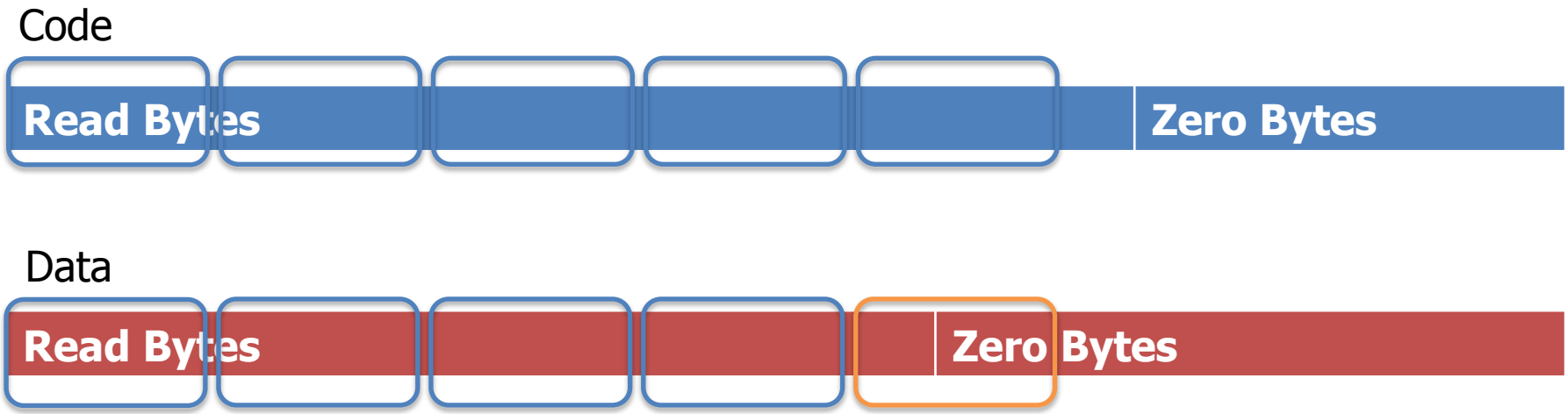
Code/data Lazy Loading (1)

- **Why?**
  - An executable file holds code and data images
  - A process will not need all the pages immediately
- **How to?**
  - Use the executable file as the backing store
  - Handling page faults
  - Similar to mmap

Code/data Lazy Loading (2)

- **Loading code/data from the executable file**
  - In `load_segment()` @ `userprog/process.c`
  - `Segments in each loop`
  - `Read bytes, zero bytes in each segment`

Code/data Lazy Loading (3)



Page replacement (1)

- **Why?**
  - You may run out of the physical memory
- **How to? (swap)**
  - Find a victim page in the physical memory
  - Swap out the victim page to the swap disk
  - Extend your supplemental page table to indicate the victim virtual addr has been swapped out
  - When the virtual addr is accessed later, swap in the page from the swap disk to the physical memory

Page replacement (2)

Segment	Convert to Dirty	Backing Storage	write to backing storage	Need Swap?
Code	No	Yes	Not permitted	No (Always Clean)
Data	Yes	Yes	Not permitted	Yes
Stack	Yes	No		Yes
Mmap	Yes	Yes	Permitted	No (writeback to backing storage)

Page replacement (3)

- **User pool**
  - Indicate an allocation of physical page in user pool
  - code, data, stack, mmap
  - PAL_USER
- **Kernel pool**
  - page table
  - struct thread
  - …….

Page replacement (4)

- **Swap disk**
  - Use the following command to create an 4 MB swap disk in the `vm/build` directory

```
$ pintos-mkdisk swap.dsk 4
```

  - Alternatively, you can tell Pintos to use a temporary 4-MB swap disk for a single run with `--swap-size=4`
    - Used during "`make check`"
  - A swap disk consists of swap slots
    - A swap slot is a continuous, page-size region of disk space on the swap disk

Page replacement (5)

- **Accessing swap disk**
  - Use the disk interface in `devices/block.h`
    - A size of a disk sector is 512 bytes
    - You can read or write one sector at a time

```
struct block *block_get_role (enum block_type);
block_sector_t block_size (struct block *);
void block_read (struct block *, block_sector_t, void *);
void block_write (struct block *, block_sector_t,
                  const void *);
```

Page replacement (6)

- **Page replacement policy**
  - You should implement a global page replacement algorithm that approximates LRU
  - Get/Clear Accessed and Dirty bits in the PTE
    - `pagedir_is_dirty()`, `pagedir_set_dirty()`
    - `pagedir_is_accessed()`, `pagedir_set_accessed()`
- **Frame table**
  - Which physical page should be evicted out?
  - Array or list will be OK

Submission

- **Document**
  - Data structure of supplemental page table, swap/mmap entry, frame table
  - Flow chart when a page fault occurs

Submission

- **Source Code**
  - Dec 10, 11:59PM
  - [group_number]_project3-2.tar.gz
  - Tar and gzip your Pintos source codes
    - after make clean
  - Upload it at sys.skku.edu
  - threads, vm, userprog, filesys directories, Makefile
  - 100s timeout per test
  - Oral test (Dec 11~)