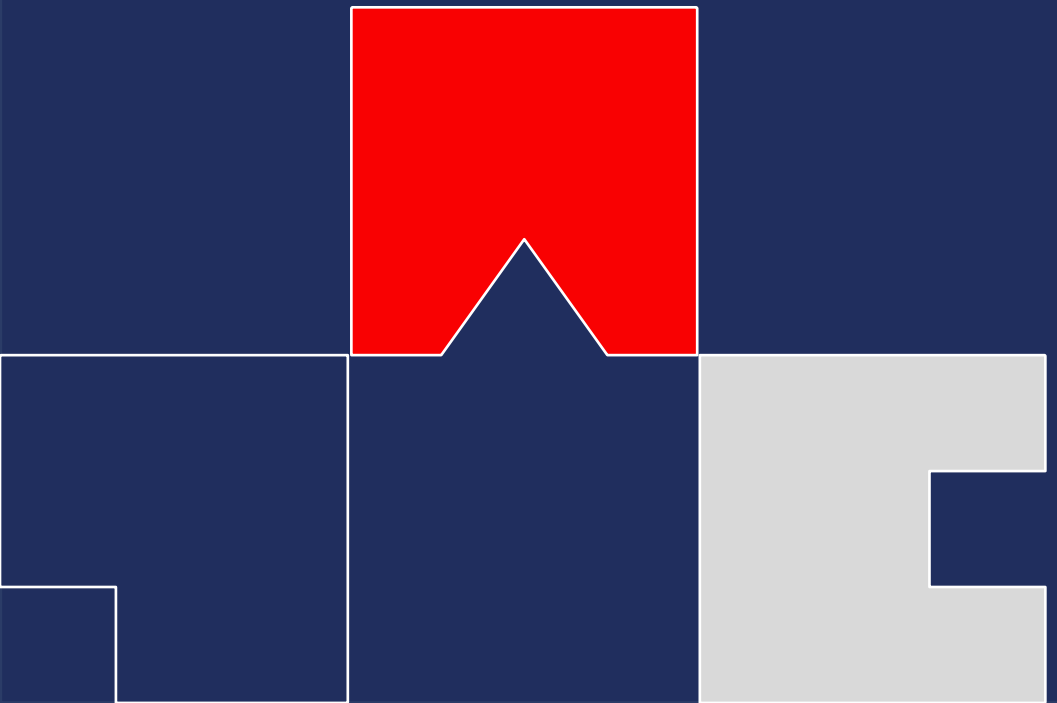
Operating Systems Design
PintOS Part3 : Virtual Memory



CONTENTS

Extra: export the container

- Export container as tar file
  - $ docker export -o [file name].tar [container name]

- Testing your file
  - $ docker import [file name].tar [image name]:[tag name]
  - $ docker run –it –name [container name] [image name]:[tag name] /bin/bash

1

Demand Paging with Swapping

Demand Paging with Swapping

- **Current pintos**
  - All files are loaded into memory at first startup → inefficient because not everything in the file is needed.
  - All physical addresses are fixed for each page at the beginning of process execution → inefficient because it can cause memory overhead and swap-out issues.

  → This inefficiency can be solved by introducing demand paging.
- **Demand paging**
  - Virtual page: Virtual Page number (20 bit) + page offset (12bit)
  - Page frame: physical frame number (20 bit) + page offset (12 bit)
  - Page table:
    - VPN → PFN
    - It is hardware.
  - Swap space: array of page sized blocks

- **Memory mapped file**

- All mappings of a process are implicitly unmapped when the process exits.
- When a mapping is unmapped, the pages are written back to the file.
- Upon releasing the memory mapping, the pages are removed from the process' virtual page list.
- Once created, mapping is valid until it is unmapped regardless of the file is closed or deleted.
- If the two or more processes map the same file, they do not have to see the consistent view.

- **Pinning page**

- Prevent evicting the pages accessed during system call
- Define pinning flag about each physical page.
- On every system call,
  - Find the virtual page and pin the associated physical page.
  - After the system call returns and before the system call handler returns, unpin the pages
- On Swapping handler,
  - Do not select a pinned page as a victim.

2

Stack Growth

Stack Growth

- **Implement expandable stack**
  - In current pintos, stack size is fixed to 4KB.
  - Make the stack expandable.
    - If a process accesses the address that lies outside the stack and that can be handled by expanding the stack, expand the stack.
      - e.g. (access address < stack pointer – 32) Expand stack
  - maximum size of stack is 8MB.

Thank you