Lab 7
Memory Management

CSCI 4061
Introduction to Operating Systems

Oct 23, 2023

Overview

- Activity
- Virtual Memory
- Changing Memory Content
- PA2 Discussion

Programming Exercise

Activity

- No programming activity
- Answer following questions in a file Answers.md and submit it
  1. Virtual memory page size is same as Physical memory frame size. (True/False)
  2. Two processes cannot have same Virtual memory address. (True/False)
  3. Given a 16-bit virtual address and page size of 4KB. How many bits are required to represent a page?(2/4/10/12)
  4. Given a 16-bit virtual address 0x7A2F with page size 4KB. What is the page offset? (0xF/0x2F/0xA2F/0xA2)

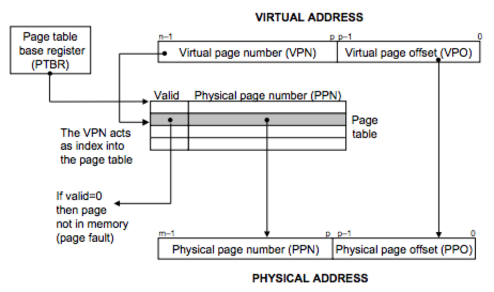
Virtual memory

What is virtual memory?

- Map memory address (virtual) of a program to physical memory
- Address translation hardware, Memory Management Unit (MMU) does the translation
- Gives a false sense of more physical memory than available
- Each process has its own virtual memory (architecture dependent)

Address Translation

Virtual to Physical address


VIRTUAL ADDRESS
Page table base register (PTBR)
n–1
p
p–1
0
Virtual page number (VPN)
Virtual page offset (VPO)
Valid
Physical page number (PPN)
Page table
The VPN acts as index into the page table
If valid=0 then page not in memory (page fault)
m–1
p
p–1
0
Physical page number (PPN)
Physical page offset (PPO)
PHYSICAL ADDRESS

- Virtual memory page size == Physical memory frame size $\Longrightarrow$ The offset can be directly used on knowing base address
- Page table used to map VPN to PPN as shown in image

Example address translation

32-bit address

- Given a 32-bit virtual address and 4 KB page size
- Page offset bit count = 12 bits ($2^{12}B$)
- Page number bit count = 32 - 12 = 20 bits
- Eg: 0x1A2B3C4D - 0x1A2B3 represents page number, 0xC4D represent page offset
- Say the page table entry for 0x1A2B3 maps to a frame with address 0x5E7F8A9B
- Physical address = 0x5E7F8A9B + 0xC4D = 0x5E7F8AF8

Memory Maps

proc

- Pseudo-filesystem that provides an interface to the kernel data strctures
- `man proc`
  - `/proc/[pid]/mem`: Access pages of a process's memory
  - `/proc/[pid]/maps`: Mapped memory regions and their access permissions
- Let's get location of variables in heap from `/proc/[pid]/mem` and try modifying the content
- You need sudo permission, so try it on your machine with Linux installation

Memory Maps

Modify Heap Content

Please do not play around in unknown memory space. It can cause system corruption. Use discretion.

- Check README
- Execute p1.c in Terminal 1
- Find pid of ./p1 using command given in README in Terminal 2
- Execute p2.c in Terminal 3 as shown in README with the identified pid

Virtual Memory demo

Terminal 1

```
$ gcc -o p1 p1.c
$ ./p1
[0] Hello [0x561babb9e2a0]
[1] Hello [0x561babb9e2a0]
[2] Hello [0x561babb9e2a0]
```

Terminal 2

```
$ ps aux | grep ./p1 | grep -v grep
nicks     324257  0.0  0.0   2772   792 pts/1    S+   19:37   0:00 ./p1
```

Terminal 2

```
$ cat /proc/324257/maps
...
561bab9e9000-561bab9ea000 rw-p 00003000 00:27 1376582  /csci4061/Lab7/code/sample/p1
561babb9e000-561babbbf000 rw-p 00000000 00:00 0    [heap] # lower address range
...
7ffe4d60c000-7ffe4d62d000 rw-p 00000000 00:00 0    [stack] # higher address range
...
```

Virtual Memory demo

Terminal 3

```
gcc -o p2 p2.c
sudo ./p2 324257 Hello
```

Terminal 1

```
[166] Hello [0x561babb9e2a0]
[167] Hello [0x561babb9e2a0]
[168] World [0x561babb9e2a0]
[169] World [0x561babb9e2a0]
```

Deliverables

Individual submission: Zip and submit to Gradescope by Oct 24, 11:59pm

1. Answers.md