Operating Systems–II: CS3523
January 2021
Programming Assignment 6: Linux Virtual Memory
Last date for submission: 30th April 2021, midnight

Setup:
  a. Install Virtualization software (virt-manager on Linux, VirtualBox on Windows/MAC) on your laptop.
  b. Create a virtual machine with 1 GiB RAM, 8GB virtual HDD (dynamic vdi), 1 vCPU, 1 network card. Make sure that during installation you partition the virtual-disk into 7GB for / fs and 1GB for swap. If you forgot to create a swap partition, shutdown to add another virtual disk of 1GB for swap. You may also create swap file in / fs. Refer this link and this
  c. Install Ubuntu 18/20 server edition 64-bit on it (no GUI). Do not install Ubuntu desktop edition, as it requires a lot of disk space and does not help in learning linux systems programming.
  d. You should be able to login via **ssh** to the VM. Make sure your VM's IP is reachable from your host (laptop).
  e. **Run your programs inside the VM setup.**

If you don't use VM setup, your laptop may hang, crash, and go bad.

Problem Statement 1:
Background:
  ● Linux maintains separate virtual address space for each process' userspace.

Goal: The goal of this assignment is to see anatomy of virtual address space of processes.

Details: On the virtual machine, run the following experiments and create a report with observations.

Experiment 1:
  a. As a normal user, run two processes of the "**sample.c**" program shared in this assignment in two different ssh sessions.
  b. While the processes are still running, collect **pmap** of both the processes from a third ssh session and save them.
      i. Identify code/data/stack/heap segments in terms of starting and ending addresses and their sizes
  c. While the processes are still running, from the third ssh session, analyze the processes' address spaces using "**pagemap**" utility shared in this assignment.
      i. Convert the virtual address of each variable and main() function into physical address, and make a note of Page Frame No.
      ii. Notice the Page Frame Number of the libc's text segment. It should be the same for both the processes.

d. Capture the outputs from (a), (b), (c) (only necessary lines), in a separate .txt report clearly marking outputs of each process. You may also modify sample.c to print its pid on the console. In the report, mention what role ASLR (Address Space Layout Randomization) played, length of virtual addresses and physical address, mapping of each of virtual addresses used in your process to physical addresses and Page Frame Nos.

Experiment 2: Modify the "sample" program to "sample2.c" include the following:
i) one uninitialized global int variable named global_unint,
ii) one constant global int variable named constant_global_max with value 999,
iii) one constant local int variable named constant_local_max with value 99,
iv) one int variable on heap using malloc;

a) As a normal user, run one instance (process) of the "**sample2.c**" program and collect **pmap** of it while it is still running from another ssh session.
    i) Identify code/data/stack/heap segments in terms of starting and ending addresses and their sizes.
b) While the process is still running analyze the process' address spaces using "**pagemap**" utility shared in this assignment from another ssh session.
    i) Convert the virtual address of each variable and main() function into physical address, and make a note of Page Frame No. Mention which segment is used for allocating memory space for each of the variable and main() function.
c) Capture the outputs from (a), (b), in a separate .txt report clearly marking outputs of the process. In the report, mention the mapping of each of the virtual addresses used in your process to physical addresses and corresponding Page Frame Nos.

Problem Statement 2:
Background:
● Linux implements swapping to swap devices or files. Swapping activity starts when there is memory pressure.

Goal: The goal of this assignment is to observe swapping activity during memory pressure.

Details: Write a C/C++ program "memleak.cpp" that iteratively does "sleep(1), malloc(100MB) and write something (minimum one byte per each page, don't need to fill all bytes in each page) to the memory" in an infinite loop. There should not be any call to free() in the program. Upon malloc failure, the program should print the number of iterations succeeded, and bail out of the loop and wait for the user's input. Create a .txt report with observations. Your program may be killed by **oom killer** before printing iteration number. But that is ok. In such case, print iteration number in every iteration

Experiment 1:
   a. As a sudo or root, switch off swap using "swapoff <device_or_file>" [to find device/file name use "swapon -s" command]
   b. As a normal user, check free memory and swap using "free -mh"
   c. As a normal user, run "vmstat -t 1" in one terminal.
   d. As a normal user, run the memleak program in another terminal.
   e. Capture the outputs from (a), (b), (c) (d) (only necessary lines), in a separate .txt report. Notice the number of iterations it succeeded.

Experiment 2:
   a. As a sudo or root, switch on swap using "swapon <device_or_file>". [to find device/file name use "swapon -s" command]
   b. As a normal user, check free memory and swap using "free -mh"
   c. As a normal user, run "vmstat -t 1" in one terminal.
   d. As a normal user, run the memleak program in another terminal.
   e. Capture the outputs from (a), (b), (c) (d) (only necessary lines) in a separate .txt report. Notice the number of iterations it succeeded.

Submission Instructions:
   1. Place the following in a folder named: Assgn-linuxvmm-<RollNo>
      a. report.txt from problem1,
      b. report.txt from problem2.
   2. Zip the folder.
   3. Upload the zip file.

If you do n't follow these guidelines, then your assignment will not be evaluated. Upload all these in the Google classroom by the specified deadline.

Evaluation:
   - Problem1
      ○ Setting up VM: 10 marks.
      ○ Report (outputs and observations) : 40 marks
   - Problem2
      ○ Code, Swapping experiments without and with swap device: 30 marks.
      ○ Report (outputs and observations) : 20 marks
   - If there are missing outputs marks will be deducted from the respective part.

Any queries in the problem statement to be posted over GC on/before April 25th.

Late Policy:

10% cut in marks for each late day