East West University Bangladesh
Computing Science and Engineering Department

CSE-325: Operating System, Memory Management- Lab 9

Objectives:

- Work with Linux memory commands.
- Understand Process Address Space-code, data, heap and stack segments.
- Learn etext, end, sbrk external variables.

top: with the help of man top command, get a clear idea about

VIRT: virtual image (kb). The total amount of virtual memory used by the process. It includes all code, data and shared libraries plus pages that have been swapped out. VIRT = SWAP + RES

SWAP: Swapped size (kb). The swapped out portion of a process's total virtual memory image.

RES: Resident size (kb). (ps -aly). The non-swapped physical memory a task has used. RES = CODE + DATA

CODE: Code size (kb). The amount of physical memory devoted to executable code, also known as the 'text resident set' size or TRS.

DATA: Data+Stack size (kb). The amount of physical memory devoted to other than executable code, also known as the 'data resident set' size or DRS.

SHR: Shared Mem size (kb). The amount of shared memory used by a task. It simply reflects memory that could be potentially shared with other processes.

nFLT: Page Fault count. The number of major page faults that have occurred for a task. A page fault occurs when a process attempts to read from or write to a virtual page that is not currently present in its address space. A major page fault is when backing storage access (such as a disk) is involved in making that page available.

nDRT: Dirty Pages count. The number of pages that have been modified since they were last written to disk. Dirty pages must be written to disk before the corresponding physical memory location can be used for some other virtual page.

You can also use ps command to monitor the status of a process
ps -o user,vsz,rss,pmem,fname -e | more

user: The user who started the process running.
vsz: The total size of the process in (virtual) memory in kilobytes as a decimal integer.
rss: The resident set size of the process, in kilobytes as a decimal integer.
pmem: The ratio of the process's resident set size to the physical memory on the machine, expressed as a percentage.
fname: The first few character's of the process' name.

Question: why the resident set is smaller than the process size?
Operating system doesn't include shared libraries (size) in the resident set, because the libraries are loaded into memory only once.

df command to see the Swap Usage:

command: df /tmp

The size program shows the sizes of the text, data and bss sections in a program.
Command: which ls
Output:/bin/ls
Command: size /bin/ls
Output: Total size of the process

```
gcc –Wall size.c
size a.out
The output should be something like
text   data   bss   dec    hex    filename
1164    504    16   1684   694     a.out
```

If you exam its size with ls -l, you should get something similar to the following line

```
–rwxr–xr–x  shamim shamim      8377    April 3 13:32 a.out


strip a.out

size a.out

The output should be something like
text   data   bss   dec    hex    filename
1164    504    16   1684   694     a.out

ls–l a.out
```

Tasks I:

1. What does the 8377 mean?
2. What do the 1164, 504, 16, and 1684 mean?
3. What does the change occur after strip call?
4. Check the following program and find the size by `size a.out`

```
#include <stdio.h>

int main(void)
{
    return 0;
}
```

5. Let's add one global variable in the program, and check the size of bss.

```
#include <stdio.h>

int global; /* Uninitialized variable stored in bss*/

int main(void)
{
    return 0;
}
```

6. Now add one static variable and check the bss.

```
#include <stdio.h>

int global; /* Uninitialized variable stored in bss*/

int main(void)
{
    static int i; /* Uninitialized static variable stored in bss */
    return 0;
}
```

7. Let us initialize the static variable check the size of Data Segment (DS)

```
#include <stdio.h>

int global; /* Uninitialized variable stored in bss*/

int main(void)
{
    static int i = 100; /* Initialized static variable stored in DS*/
    return 0;
}
```

8. Let us initialize the global variable and check the size of Data Segment (DS)

```
#include <stdio.h>

int global = 10; /* initialized global variable stored in DS*/

int main(void)
{
    static int i = 100; /* Initialized static variable stored in DS*/
    return 0;
}
```

ldd command to see the shared libraries:
shared libraries are sub routines, which are used as common routines to many programs. The printf() function is used by nearly all C programs, and so load it once into memory, and share it among all C processes.

Command: ldd /bin/ls
Output: Show all shared libraries used by ls

Page size: Memories are broken into some chunks, each are 8192 bytes or 4096 bytes chunk.
Command: **getconf PAGESIZE** or **getconf PAGE_SIZE**.

use **limit / ulimit – s** command to see the default stack size. You can also change the stack size using limit.

Memory is like a huge array with (say) 0xffffffff elements (32 bits system) and 0xffffffffffff elements (64 bits system).

Question:

- Why does 64 bits system not support 2^64 bytes virtual address space?
- Why does 64 bits system support 2^48 bytes virtual address space? (x86-64 CPUs support 48 address pin)

Tasks II:

1. Run lab0.c program and trace the output? Answer: why does it show **“segmentation violation”** message.
2. Run lab1.c program and trace the output? Explain each message.
3. Run lab2.c program and trace the output? Explain how the local parameters work.
4. Run lab3.c program and trace the output? Explain how stack works.