Operating System Design & Implementation

TA: 周昆霖 吳崧銘

Lab 8: Memory Killer

Objective:

In this Lab, you will learn

- Memory Management structure
- Memory Management structure modifications
- Kill the processes according to a pre-defined memory usage criteria (kernel threads) and some critical threads excluded, for example Xorg)

Experiment:

- 8-1: Dump the memory usage of all processes, sorted in size of memory used
- 8-2: Kill the processes in different criteria of memory usage

Exp. 8-1 Dump the memory usage of all process (including kernel threads)

- Compile and run the program (malloc.c).
- You are requested to list the memory usage of all processes.
  1. Obtain memory usage of all processes
  2. Sort the memory usage and list the processes in ascending order

- Implement a kernel thread in kernel module

  memoryKiller.c

  <hint>

  i. Implement static int memoryKiller(void * data){

     daemonize("memoryKiller");

     Step1 : traverse all the process

     Step2 : get mm_struct from task_struct

     Step3 : get the resident memory

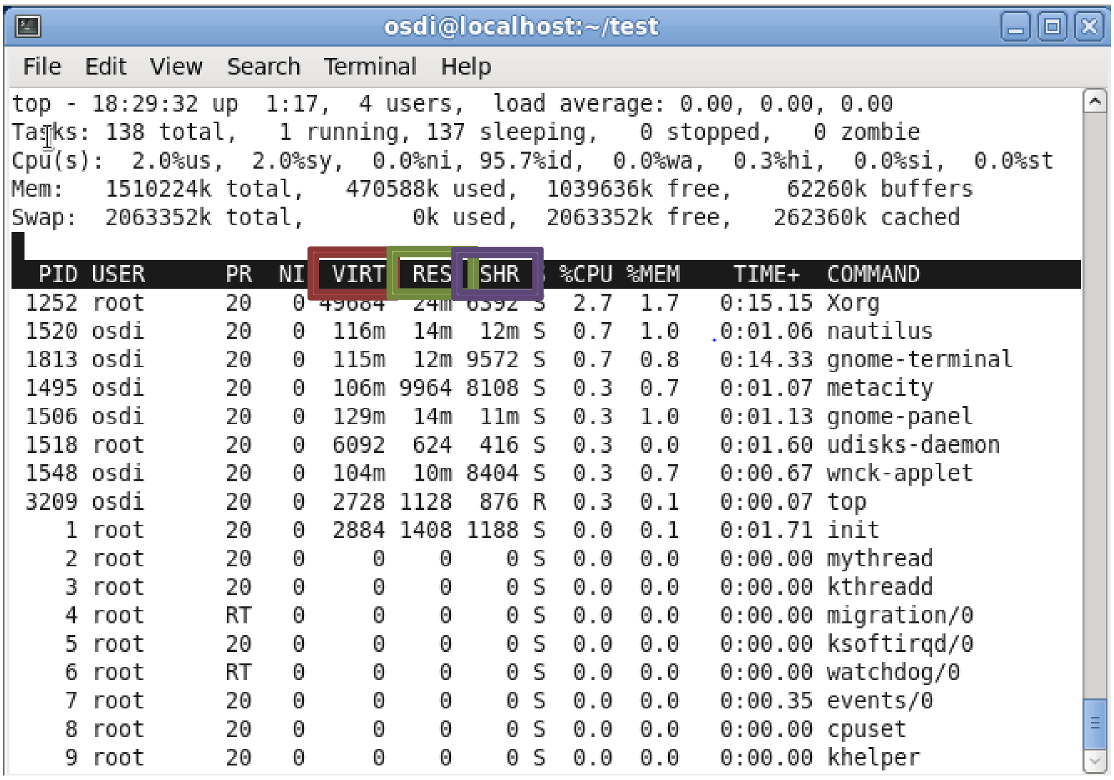
     Step4 : Sorted in size of memory used for all processes.

     Step5 : Kill the process which has the largest resident memory form all processes.

     }

  ii. Q: What is the resident memory??

  iii. A: The resident memory which means that the operating system is not permitted to swap them out to a storage device; they will always remain in memory



```
top - 18:29:32 up  1:17,  4 users,  load average: 0.00, 0.00, 0.00
Tasks: 138 total,   1 running, 137 sleeping,   0 stopped,   0 zombie
Cpu(s):  2.0%us,  2.0%sy,  0.0%ni, 95.7%id,  0.0%wa,  0.3%hi,  0.0%si,  0.0%st
Mem:   1510224k total,   470588k used,  1039636k free,    62260k buffers
Swap:  2063352k total,        0k used,  2063352k free,   262360k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 1252 root      20   0 49684  24m 6392 S  2.7  1.7   0:15.15 Xorg
 1520 osdi      20   0  116m  14m  12m S  0.7  1.0   0:01.06 nautilus
 1813 osdi      20   0  115m  12m 9572 S  0.7  0.8   0:14.33 gnome-terminal
 1495 osdi      20   0  106m 9964 8108 S  0.3  0.7   0:01.07 metacity
 1506 osdi      20   0  129m  14m  11m S  0.3  1.0   0:01.13 gnome-panel
 1518 root      20   0  6092  624  416 S  0.3  0.0   0:01.60 udisks-daemon
 1548 osdi      20   0  104m  10m 8404 S  0.3  0.7   0:00.67 wnck-applet
 3209 osdi      20   0  2728 1128  876 R  0.3  0.1   0:00.07 top
    1 root      20   0  2884 1408 1188 S  0.0  0.1   0:01.71 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.00 mythread
    3 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kthreadd
    4 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    5 root      20   0     0    0    0 S  0.0  0.0   0:00.00 ksoftirqd/0
    6 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 watchdog/0
    7 root      20   0     0    0    0 S  0.0  0.0   0:00.35 events/0
    8 root      20   0     0    0    0 S  0.0  0.0   0:00.00 cpuset
    9 root      20   0     0    0    0 S  0.0  0.0   0:00.00 khelper
```

iv. Virtual image (in red color) :The total virtual memory used by a process . Total memory here includes CODE , DATA, share memory and swap out.

v. Share memory(in purple color): This memory share to other process.

vi. You can get total virtual memory form mm_struct->total_vm, and get share memory form mm_struct->shared_vm.

vii. The resident memory include two part :

   1. Number of page frames allocated to the Process but map to file => mm_struct->file_rss
   2. Number of page frames assigned to anonymous memory mappings
      => mm_struct->anon_rss

viii. Use macro get_mm_rss to get resident memory
The macro id defined from
/usr/src/kernels/linux-2.6.32.60/linux/sched.h

Exp. 8-2 Kill the processes according to the following conditions

- You are requested to send signals to processes, with the largest resident memory and some critical threads such as kernel threads and Xorg are excluded.

i. Kill the process by sys_tkill

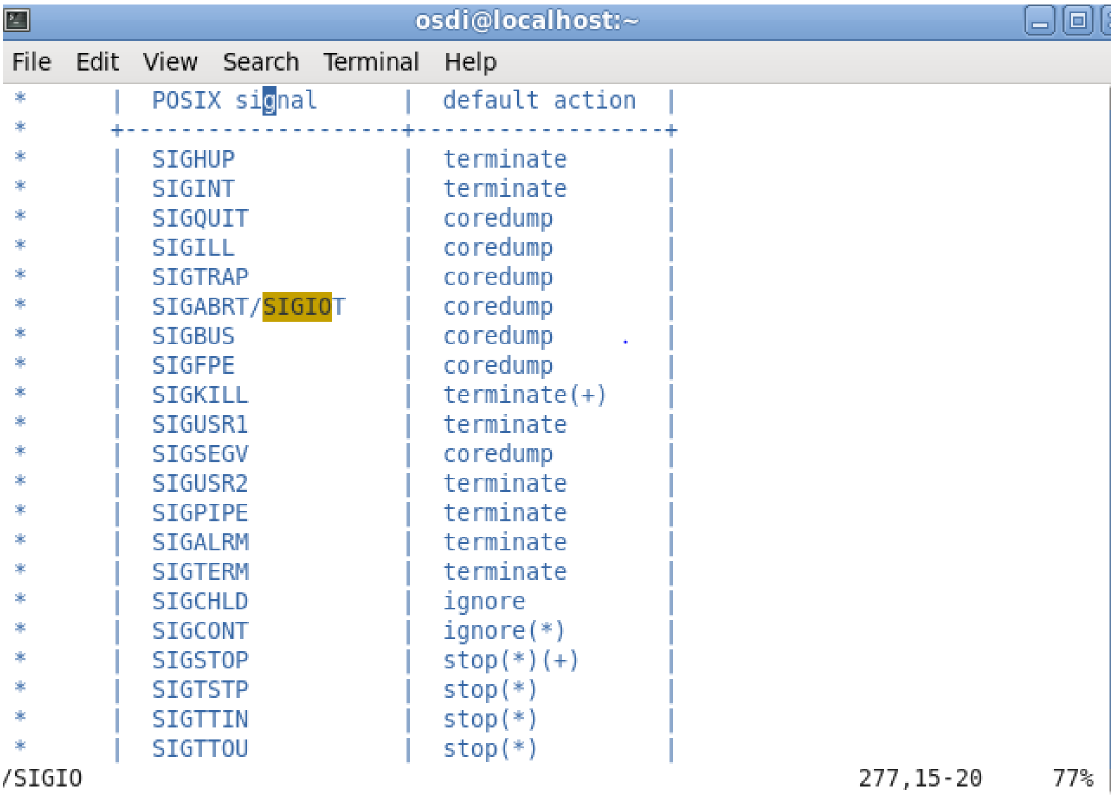
<hint>

Step1 :get sys_call_table address
   unsigned long *sys_call_table = (unsigned long
   *)address of sys_cal_table;
Step2 : get sys_tkill address
   sys_tkill = sys_call_table[__NR_tkill];
Step3 send kill signal to process
   sys_tkill(pid , sign)

1. Be sure to add asmlinkage long(*sys_tkill)(int pid , int sig) to the top line of memoryKiller.c

2. In step1 you can use command : grep sys_call_table /boot/System.map to get the address of sys_call_table

3. In step3 please refer to /usr/src/kernels/linux-2.6.32.60/include/linux/signal.h line 279 to get a signal as sys_tkill argument



```
 *      |  POSIX signal      |  default action  |
 *      +--------------------+------------------+
 *      |  SIGHUP            |  terminate       |
 *      |  SIGINT            |  terminate       |
 *      |  SIGQUIT           |  coredump        |
 *      |  SIGILL            |  coredump        |
 *      |  SIGTRAP           |  coredump        |
 *      |  SIGABRT/SIGIOT    |  coredump        |
 *      |  SIGBUS            |  coredump     .  |
 *      |  SIGFPE            |  coredump        |
 *      |  SIGKILL           |  terminate(+)    |
 *      |  SIGUSR1           |  terminate       |
 *      |  SIGSEGV           |  coredump        |
 *      |  SIGUSR2           |  terminate       |
 *      |  SIGPIPE           |  terminate       |
 *      |  SIGALRM           |  terminate       |
 *      |  SIGTERM           |  terminate       |
 *      |  SIGCHLD           |  ignore          |
 *      |  SIGCONT           |  ignore(*)       |
 *      |  SIGSTOP           |  stop(*)(+)      |
 *      |  SIGTSTP           |  stop(*)         |
 *      |  SIGTTIN           |  stop(*)         |
 *      |  SIGTTOU           |  stop(*)         |
/SIGIO                                                       277,15-20    77%
```