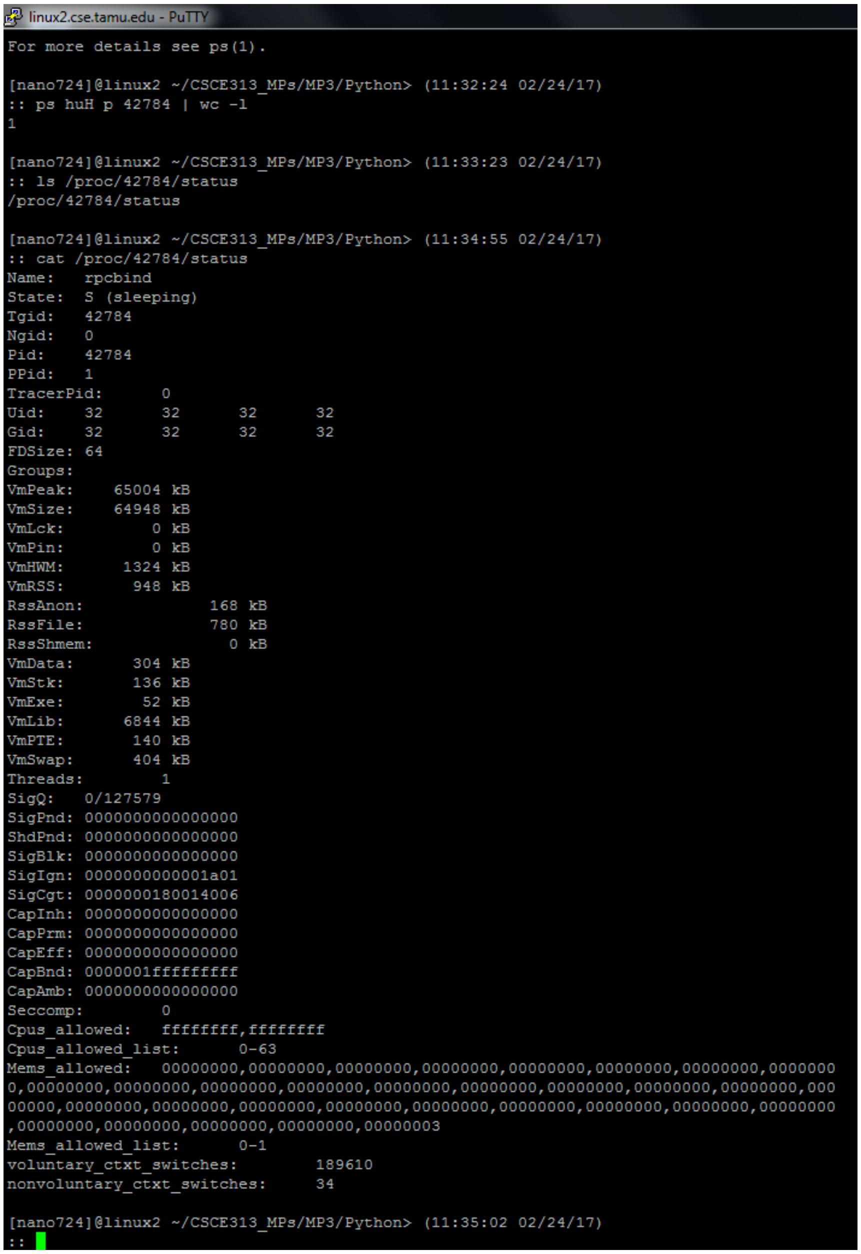
Natalie Cluck, Megan Fischer

February 24, 2017

CSCE 313-504

Machine Problem 3 Report

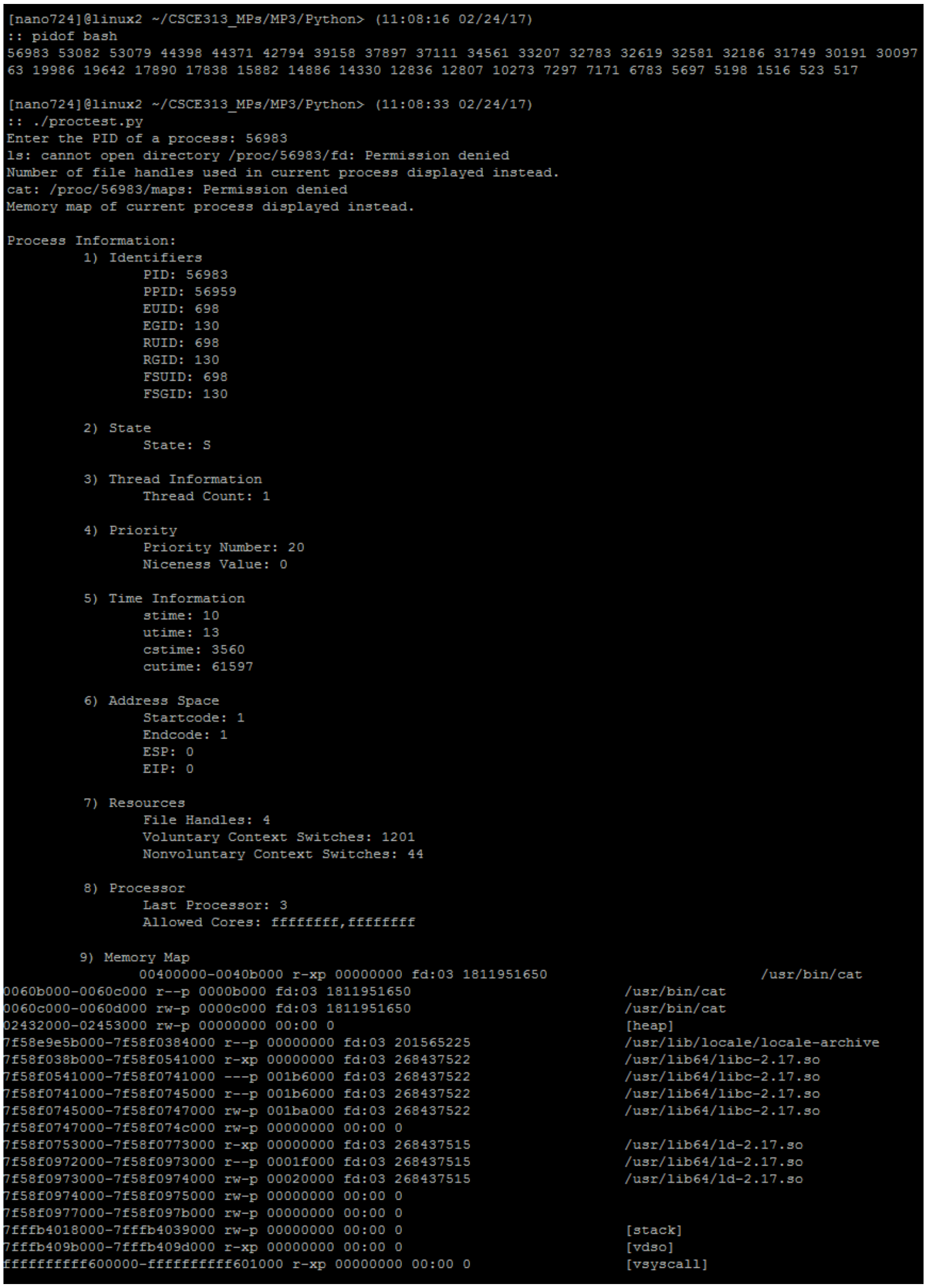
1. For a process run by a user other than yourself, find the following items from Table #1: [Identifiers, State, Thread Information, Priority, Time Information, Resources, and Memory Map]
    a. To find these items, type: “cat /proc/<pid>/status” and it displays all information about that process if it’s run by another user.



```
For more details see ps(1).

[nano724]@linux2 ~/CSCE313_MPs/MP3/Python> (11:32:24 02/24/17)
:: ps huH p 42784 | wc -l
1

[nano724]@linux2 ~/CSCE313_MPs/MP3/Python> (11:33:23 02/24/17)
:: ls /proc/42784/status
/proc/42784/status

[nano724]@linux2 ~/CSCE313_MPs/MP3/Python> (11:34:55 02/24/17)
:: cat /proc/42784/status
Name:   rpcbind
State:  S (sleeping)
Tgid:   42784
Ngid:   0
Pid:    42784
PPid:   1
TracerPid:      0
Uid:    32      32      32      32
Gid:    32      32      32      32
FDSize: 64
Groups:
VmPeak:    65004 kB
VmSize:    64948 kB
VmLck:         0 kB
VmPin:         0 kB
VmHWM:      1324 kB
VmRSS:       948 kB
RssAnon:             168 kB
RssFile:             780 kB
RssShmem:              0 kB
VmData:      304 kB
VmStk:       136 kB
VmExe:        52 kB
VmLib:      6844 kB
VmPTE:       140 kB
VmSwap:      404 kB
Threads:        1
SigQ:   0/127579
SigPnd: 0000000000000000
ShdPnd: 0000000000000000
SigBlk: 0000000000000000
SigIgn: 0000000000001a01
SigCgt: 0000000180014006
CapInh: 0000000000000000
CapPrm: 0000000000000000
CapEff: 0000000000000000
CapBnd: 0000001fffffffff
CapAmb: 0000000000000000
Seccomp:        0
Cpus_allowed:   ffffffff,ffffffff
Cpus_allowed_list:      0-63
Mems_allowed:   00000000,00000000,00000000,00000000,00000000,00000000,00000000,0000000
0,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,000
00000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000
,00000000,00000000,00000000,00000000,00000003
Mems_allowed_list:      0-1
voluntary_ctxt_switches:        189610
nonvoluntary_ctxt_switches:     34

[nano724]@linux2 ~/CSCE313_MPs/MP3/Python> (11:35:02 02/24/17)
::
```

    b.

2. For a process that you have created, retrieve all items enumerated in Table.

    a. 

```
[nano724]@linux2 ~/CSCE313_MPs/MP3/Python> (11:08:16 02/24/17)
:: pidof bash
56983 53082 53079 44398 44371 42794 39158 37897 37111 34561 33207 32783 32619 32581 32186 31749 30191 30097
63 19986 19642 17890 17838 15882 14886 14330 12836 12807 10273 7297 7171 6783 5697 5198 1516 523 517

[nano724]@linux2 ~/CSCE313_MPs/MP3/Python> (11:08:33 02/24/17)
:: ./proctest.py
Enter the PID of a process: 56983
ls: cannot open directory /proc/56983/fd: Permission denied
Number of file handles used in current process displayed instead.
cat: /proc/56983/maps: Permission denied
Memory map of current process displayed instead.

Process Information:
         1) Identifiers
                PID: 56983
                PPID: 56959
                EUID: 698
                EGID: 130
                RUID: 698
                RGID: 130
                FSUID: 698
                FSGID: 130

         2) State
                State: S

         3) Thread Information
                Thread Count: 1

         4) Priority
                Priority Number: 20
                Niceness Value: 0

         5) Time Information
                stime: 10
                utime: 13
                cstime: 3560
                cutime: 61597

         6) Address Space
                Startcode: 1
                Endcode: 1
                ESP: 0
                EIP: 0

         7) Resources
                File Handles: 4
                Voluntary Context Switches: 1201
                Nonvoluntary Context Switches: 44

         8) Processor
                Last Processor: 3
                Allowed Cores: ffffffff,ffffffff

        9) Memory Map
                00400000-0040b000 r-xp 00000000 fd:03 1811951650                         /usr/bin/cat
0060b000-0060c000 r--p 0000b000 fd:03 1811951650                         /usr/bin/cat
0060c000-0060d000 rw-p 0000c000 fd:03 1811951650                         /usr/bin/cat
02432000-02453000 rw-p 00000000 00:00 0                                  [heap]
7f58e9e5b000-7f58f0384000 r--p 00000000 fd:03 201565225                  /usr/lib/locale/locale-archive
7f58f038b000-7f58f0541000 r-xp 00000000 fd:03 268437522                  /usr/lib64/libc-2.17.so
7f58f0541000-7f58f0741000 ---p 001b6000 fd:03 268437522                  /usr/lib64/libc-2.17.so
7f58f0741000-7f58f0745000 r--p 001b6000 fd:03 268437522                  /usr/lib64/libc-2.17.so
7f58f0745000-7f58f0747000 rw-p 001ba000 fd:03 268437522                  /usr/lib64/libc-2.17.so
7f58f0747000-7f58f074c000 rw-p 00000000 00:00 0
7f58f0753000-7f58f0773000 r-xp 00000000 fd:03 268437515                  /usr/lib64/ld-2.17.so
7f58f0972000-7f58f0973000 r--p 0001f000 fd:03 268437515                  /usr/lib64/ld-2.17.so
7f58f0973000-7f58f0974000 rw-p 00020000 fd:03 268437515                  /usr/lib64/ld-2.17.so
7f58f0974000-7f58f0975000 rw-p 00000000 00:00 0
7f58f0977000-7f58f097b000 rw-p 00000000 00:00 0
7fffb4018000-7fffb4039000 rw-p 00000000 00:00 0                          [stack]
7fffb409b000-7fffb409d000 r-xp 00000000 00:00 0                          [vdso]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0                  [vsyscall]
```

3. What are the differences between the real user IDs and effective user IDs, and what is a situation where these will be different?

    a. An effective user ID is used when you may need to temporarily take over another user's identity. With only one user ID, there would be no way of changing back to the original user ID. Real user ID is the actual ID, where the effective user ID is where the OS determines whether certain tasks are allowed for that ID. A situation where these will be different is when a setuid program is executed.

4. Why are most of the files in /proc read only?
    a. They contain a lot of important fundamental information about a process, and if the /proc files were easily writable, it could do serious damage, or it would allow an intruder to more easily corrupt the system. It is mostly read-only for security reasons.
5. Why is the task_struct so important to the kernel and what is it used for?
    a. The task_struct is so important to the kernel because it contains all the information about a particular process. It is the ultimate information that a kernel needs to know about any process. Without the task_struct, the kernel would not have the necessasary data needed to execute a program.