CS 4420/5420

Fall 2018/2019

Programming Assignment 1

Exploring the Proc Filesystem

Max. Points: 20

Due: 10/3/2018 (11:59pm)

Part III: The "4420_proctool_ps" Tool

Overview:

In the third part of assignment 1, you will implement the tool "4420_proctool_ps". This tool is a re-implementation of the Unix tool "ps" which lists statistics for all the processes in the system. More specifically, the "4420_proctools_ps" tool should produce (a subset of) all the information a call to "ps aux" would produce. The tool has one required command argument and has the following syntax:

Shell> 4420_proctool_ps [-cpu|-mem|-pid|-com]

The tool will produce statistical information about all the process in the system. Specifically, it should produce a table containing the following columns:

PID (u_int)	Command (string)	State (char)	%CPU (float)	%MEM (float)	VSZ(virtual memory size) (u_long; in bytes)	RSS (resident set size) (u_long; in bytes)

In order to calculate the values in the different columns, you will need to access two different proc files:

- "/proc/<pid>/stat"
- "/proc/uptime

Note that the difference between these two proc file locations is that each process has its own "/proc/<pid>/stat" file, while the system as a whole has only one "/proc/uptime" file. Also note that the contents in these files constantly change. In other words, if you read the contents twice, at different points in time, they will reflect the differences in the system between those two time points.

An important resource for you to understand the structure in both files is again the Linux man page for proc (see Part I). It describes the format of both files in detail.

Most of the data needed in the different columns above can be directly read from "/proc/<pid>/stat". Specifically, you can read the "PID", the "Command" string (note that commands are stored in

parentheses, the “State”, the virtual memory size “VSZ”, and the resident set size “RSS”. It turns out that the “%CPU” and “%MEM” field require a little bit extra work.

Let’s start with the “%MEM” field. Since we’re basically re-implementing the Unix “ps” command, we can look at the definition of “%MEM” This field is defined as (see “man proc”) “ratio of the process’s resident set size to the physical memory on the machine, expressed as a percentage”. Since we already know the (RSS) resident set size (see above), all we need to calculate “%MEM” is the availability of physical memory in the machine. There are different ways of doing it, but I suggest you use the “sysconf()” function. This works as follows:

```
long phys_pages = sysconf(_SC_PHYS_PAGES);
long page_size = sysconf(_SC_PAGE_SIZE);

long phys_mem_size = phys_pages * page_size;
```

Next, the RSS is specified in units of numbers of “pages”. A “page” is a fixed size, basic management unit in memory management. Different CPUs may use different page sizes. In order to calculate the actual physical memory footprint of a process in bytes, the value of RSS (as read from “/proc/<pid>/stat”, needs to be multiplied with the (constant) value of the page size. This value can be obtained using the function “getpagesize()” (see man getpagesize) or using “sysconf()” again (as shown above). Now you can calculate the “%MEM” for each process by using the following equation:

(RSS * getpagesize() * 100) / phys_mem_size

The “%CPU” is a little bit more complicated. First note that the Unix commands “ps” and “top” use different ways of calculating “%CPU”. In fact, “top” uses a more precise way of calculating this statistic. Since we re-implement “ps” though, we will use its method. You should be able to verify the correctness of your implementation by comparing your calculations to the ones done by the “ps” tool. Note that its method is not very accurate. For example, it is possible (under certain circumstances) to obtain a %CPU value for a process that exceeds 100%! The Unix tool “ps” defines %CPU (see man ps) as “cpu utilization of the process in “##.#” format. Currently it is the CPU time used divided by the time the process has been running (cputime/realtime ratio), expressed as percentage. It will not add up to 100% unless you are lucky”

Here is how this value can be calculated. First extract the two fields “utime” and “stime” from “/proc/<pid>/stat” (see man -s5 proc). These two field represent the amount of time that a process has been scheduled in user mode (“utime”) and in kernel mode (“stime”), measured in clock ticks. In order to translate these values to seconds, you will need to divide them by system clock speed in HZ (divide by “sysconf(_SC_CLK_TCK)”). In other words,

process_time = (“utime” / sysconf(_SC_CLK_TCK)) + (“stime” / sysconf(_SC_ClK_TCK))

is the total amount of time (in seconds) that a process has received.

Next, you will need to determine the time a process started after (i.e. relative to) system boot. This value, "starttime" is also contained in the file "/proc/<pid>/stat".

Now you will need the total system "uptime", which represents the total amount of "useful" time the CPU has spent since boot. This value is stored in a per-system "/proc/uptime" (see man proc). This file contains two numbers. The one you need is the first one. Note that this time is already given in units of seconds (not clock ticks; you therefore don't need to divide this value).

Now, you finally can calculate a process's %CPU using the following formula:

real_time = "uptime" – ("starttime" / sysconf(_SC_CLK_TCK))

"%CPU" = real_time = process_time * 100 / real_time

```
drews@pu2:~/proctools$ ./4420_proctools_ps -cpu
PID                    Command    State   %CPU          %MEM          VSZ                RSS
8                    rcu_sched    I       0.115612      0.000000      0                  0
1851                      nscd    S       0.079461      0.024743      863248             8116
1                      systemd    S       0.014441      0.045864      47160              15044
1931                irqbalance    S       0.011184      0.000817      19568              268
1044                     snapd    S       0.010977      0.073851      606760             24224
948                 dbus-daemon    S       0.010662      0.028255      132244             9268
8092                        sshd    S       0.010063      0.016792      163692             5508
8093                        bash    S       0.010063      0.015414      14668              5056
7955                        sshd    S       0.007514      0.033511      163692             10992
17972          indicator-keybo    S       0.006381      0.142020      568196             46584
17885                      Xorg    S       0.006019      0.213212      754396             69936
1994                       ntpd    S       0.005836      0.013317      32664              4368
8126                        bash    S       0.005044      0.010805      13132              3544
7963                     systemd    S       0.005028      0.014024      37212              4600
3466                kworker/1:0    I       0.004994      0.000000      0                  0
16                  ksoftirqd/1    S       0.004708      0.000000      0                  0
17920             unity-greeter    S       0.004319      0.214432      1049680            70336
1973                  automount    S       0.003895      0.032670      635612             10716
17906                  systemd    S       0.003877      0.038474      44640              12620
1900                     rwhod    S       0.003757      0.000256      4788               84
1971               sendmail-mta    S       0.003632      0.016109      99384              5284
8247                kworker/1:2    I       0.003180      0.000000      0                  0
17537          accounts-daemon    S       0.002854      0.022316      271000             7320
17451           systemd-logind    S       0.002610      0.023572      25020              7732
20643               kworker/0:2    R       0.001955      0.000000      0                  0
738                 jbd2/sda9-8    S       0.001947      0.000000      0                  0
17971          indicator-datet    S       0.001572      0.040877      449876             13408
17684             rtkit-daemon    S       0.001333      0.004707      175480             1544
18020          unity-settings-    S       0.001216      0.086680      613988             28432
712                kworker/1:1H    I       0.001207      0.000000      0                  0
1995               kworker/5:1H    I       0.001194      0.000000      0                  0
325            systemd-journal    S       0.001151      0.015756      35384              5168
20640               kworker/3:1    I       0.001000      0.000000      0                  0
20645               kworker/2:2    I       0.000955      0.000000      0                  0
7139              kworker/u16:0    I       0.000856      0.000000      0                  0
7425              kworker/u16:2    I       0.000818      0.000000      0                  0
7779              kworker/u16:1    I       0.000698      0.000000      0                  0
2024                       perl    S       0.000638      0.012914      19000              4236
17919               dbus-daemon    S       0.000564      0.010122      35212              3320
17957            dconf-service    S       0.000549      0.014646      178532             4804
17968          indicator-messa    S       0.000509      0.031182      348624             10228
32352               kworker/5:2    I       0.000493      0.000000      0                  0
17985          indicator-sessi    S       0.000425      0.021572      555292             7076
1040                  rsyslogd    S       0.000327      0.024548      342684             8052
17936          at-spi2-registr    S       0.000310      0.019646      206884             6444
67                  khugepaged    S       0.000305      0.000000      0                  0
1412                        sshd    S       0.000283      0.019341      61328              6344
946                         cron    S       0.000264      0.008427      22680              2764
```

Specific Requirements:

- Implement the above described tool "4420_proctools_ps". The tool has one required argument which is the sort order in which the process's entries in the table (columns as described above) are sorted. The following sort orders should be supported:
    - 4420_proctools_ps -cpu
      sort the rows based on "%CPU" (from highest to lowest)

    - 4420_proctools_ps -mem
      sort the rows based on "%MEM" (from highest to lowest)

    - 4420_proctools_ps -pid
      sort the rows based on "PID" (from lowest to highest)

    - 4420_proctools_ps -com
      sort the rows lexicographically by command name
- You have to implement the tool either in C or C++. You need to use the proc filesystem (as explained above) for this project. Do not use any other functions (example: system(), getrusage(), etc.)
- Calling "4420_proctools_ps" witch an invalid argument (or without required argument) should create some error message
- I put a reference implementation of "4420_proctools_ps" in my home directory on the prime machines. You can access it in the folder "/home/drews/proctools".
- The screenshot above shows a sample output of my reference tool.
- You will need to submit the source file(s), header files (if necessary), and you WILL NEED TO SUBMIT A **MAKEFILE**. I will grade the project on "pu2.cs.ohio.edu". I will type "make" to build the executable tool (which, again, should be called "4420_proctools_ps"). If "make" fails (due to an error), the GRADE FOR THE PROJECT WILL BE AN "F"!
- You will need to submit the program from "**p2.cs.ohio.edu**" (NOTE: THIS IS IMPORTANT. THE SUBMISSION WILL NOT WORK FROM ANY OTHER (LAB)MACHINE OR SERVER!). The submission is simple: "cd" into your project folder and type the following command (exactly as is shown here): "**`bash> /home/drews/bin/442submit 3`**". Should you need to re-submit your project (due to changes), you will need to add an override switch "-f" to the submission: "**`bash> /home/drews/bin/442submit -f 3`**".

Deadline:

The project is **due by 10/3/2018 (11.59pm)**. I will **not accept any late submissions**.

Additional Help and Resources:

You can use similar functions as in Parts I and II.