Operating System

Computadors – Grau en Ciència i Enginyeria de Dades – 2020-2021 Q2

Facultat d'Informàtica de Barcelona

The Operating System (OS) topic of this course addresses several issues related to process and memory management, as well as basic concepts and functionalities provided by the OS. Additionally, the subsequent topics of this course will also address OS related issues.

This particular Lab session aims at introducing basic tools to let you analyze process management. In particular, approximately the first half of the weekly lab session is focused on Linux, whilst the second half is focused on Windows. As indicated below, this semester the Windows part is not mandatory.

Linux - Processes Management Commands

Linux provides several tools to analyze processes alive in the system. In this course, and in particular in this Lab session, we are going to focus on three of them, namely: "ps", "pstree", and "top".

"ps" (that stands for "process status") shows summarized information about a selected group of processes that exist in the system. Depending on the flags used in the command line, the user can select the processes to analyze and/or the information returned by the tool. By default, "ps" shows information of processes launched by the current user in the current terminal (window).

Exercise 1

*Check the information from the "ps" man page, in particular the flags (**NOTE**: you have to substitute <username> for the user's username that you want to analyze):*

(1) (no flags provided, default output) => only processes bound to the terminal

(2) "-a" => processes bound to any terminal, except session leaders (bash)

(3) "-u" or "u" (they do the same)

*(4) "-u **<username>**"*

(5) "-e" => all processes

(6) "-f" => full information

And (7) "-fL" => … including threads information (LWP)

You can combine some of them. If the output is quite larger than the terminal window size, you can control the output executing the following command line:

```
#> ps <FLAGS> | less
```

Afterwards, create the "answers.txt" file and write down the output obtained from the execution of "ps" with the above mentioned flags. Try to explain, as far as you can, the meaning of every column that appears in the output of the different options.

There is a branch of “ps” command, called “pstree”. This command shows you the process hierarchy of all processes in the system, and how they are related (that is, the relationship among processes). We recommend you to run it with the “-p” flag to show the PIDs.

Exercise 2

Edit the “answers.txt” file to indicate the names of the processes at all levels of the hierarchy shown by the “pstree” command launched by the current interpreter (e.g. bash, tcsh, etc). If there are too many levels of hierarchy and they cannot be represented properly on your terminal, you can increase the window size or reduce the font size. You can directly modify the font size by pressing “Ctrl” and “+” (to increase the font size) or “-“ (to reduce the font size). You can also use the menu of the terminal window to change the font size.

The third command is “top”. It is similar to “ps”, but with an interactive interface. In fact, if you still have the “htop” tool suggested to be used in previous Lab sessions, you can also use it. The “top” command updates the information every 3 seconds, by default. You can configure the delay using the flag “-d <time>” where time is the period of time, in seconds (floating point numbers are also accepted), to update the output. Once the “top” is running you can Quit pressing “q” and you can access the Help menu pressing “h”. You can find more information in the manual of “top”.

Exercise 3

The “ps” command has a “-o” option that allows a full configuration of the values displayed. For example,

```
#> ps -o pid,args        # -o gets a comma-separated list of fields to be displayed
```

shows the pid and the arguments (including the command) being executed by each process listed.

Find the list of field names that you can provide to “-o” in order to obtain the same output as the “top” command. Hints: man ps, and the header of the top command:

PID USER PR NI VIRT RES ~~SHR~~ S %CPU %MEM TIME+ COMMAND

Observe that some of the fields have a different name. Also, observe that the SHR (shared) field of top seems not to be provided by “ps”, so do not search much time for it on the “ps” man page.

Write the “ps -o ...” command that you find to correspond with the output of “top”, in your answers.txt file.

We strongly suggest you to keep a terminal open with “top” running during this session to ease the understanding of what is going on in the different exercises.

Linux - Preliminary Analysis of Process Management

Download the attached file of this lab session from this link (https://docencia.ac.upc.edu/FIB/GCED/COM/documents/Lab/FilesS7.tar.gz) that is very similar to the one you used in Session 4. Read carefully how the taskset command is used in the “launch.sh” shellscript. Then, compile the programs (#> make) and use the taskset command to bind two executables to the same hardware thread (check what you did in Session 4).

The process management in Linux uses a priority-based preemptive policy, similar to the one explained in the Theory lectures. In fact, from the “top” output we want to highlight the meaning of two columns: PR and NI. The former, “PR” stands for “Priority”, that is the priority weigth of every process to be employed by the scheduler of the OS. It is an integer value (ranged from 0 to 39, with a default value of 20 for standard processes). Higher integer values mean lower priority. Lower decimal values mean higher priority. “rt” means the highest priority (i.e. “real-time”). The latter, “NI” stands for “Niceness”. It is an integer value added to the priority base (i.e. 20). The result of adding niceness to the default priority value is the actual priority of the process. According to the scheduling policy, higher priority processes have higher likelihood to use resources than lower priority processes. Regular users, except “root” and similar users, cannot put lower integer values than the default priority. That is, the OS protects process management to guarantee users do not abuse of high priority usage. Thus, common users only can reduce their own processes priorities by increasing the nice value. For example, if you launch a process with “NI=10”, it means that “PR=20+10” and therefore “PR=30”. In other words, this process has lower priority than others. This can be done by the use of “nice” command (check its manual information).

Exercise 4

Launch (using the taskset command) one binary, and then a second instance of the same program by using the “nice” command (that is, “nice ./program &”). Analyze the output of “top”. Edit the “answers.txt” file to indicate the output of “top” from these two processes as well as your findings.

If you prefer so, you can use the “-v” flag on the programs to see the progress they do.

***NOTE**: observe that we have extended the number of iterations performed by the programs, so that they take minutes to end. You can finish the execution of both processes by executing the command line “#>kill -9 <PID>” where PID is the identifier of the process you want to finish.*

Another interesting command is “renice”. It allows to change the niceness value of any of your own processes, while they are already running:

```
#> renice  -n  <niceness>  <pid>    # sets the niceness value of <pid> to <niceness>
```

Non-privileged users can only increase their processes’ niceness value.

Exercise 5

Launch (now you can use the launch.sh script) two of the binaries running on the same hwthread, and use the “renice” command to change the niceness value of one of them, while you look at the “top” output and determine how the percentage of cpu time (%CPU) varies.

Fill the following table and write it into the answers.txt file:

Default %CPU of both processes before using “renice”: _____ and _______

%CPU obtained when one of the processes is reniced to 5: ______ and ______

%CPU obtained when one of the processes is reniced to 10: ______ and ______

%CPU obtained when one of the processes is reniced to 15: ______ and ______

%CPU obtained when one of the processes is reniced to 19: ______ and ______

Linux - Pseudo Filesystem “/proc” and files in “/proc/<PID>”

Linux and other UNIX based systems provide a special folder named “/proc” (it is actually called a pseudo-filesystem), that maintains updated, in real-time, the status of the system. Throughout this section you can access the manual of “proc” to analyze the different files and/or information you will deal with (man 5 proc).

In the “/proc” folder there are several entries that provide information dynamically updated about the current status of the system. It is important to identify the large list of directories that show a number as name. Every numeric folder match to every PID that currently exists in the system. Each of them includes information of the corresponding process. That is, the folder “/proc/<PID>” has a list of files and sub-folders that show information of the process “PID”. We strongly suggest you to dump the contents of this folder by using the full format of the “ls” command. The manual of “proc” shows detailed informatqion of such content represented as “/proc/[pid]/…”. We will analyze different files in subsequent Lab sessions.

Exercise 6

Use the “ps” command to find out the PID of the SHELL you are currently using (e.g. bash, tcsh, …). Then, access to its corresponding “/proc/<PID>” folder and dump the content of the “status” file with the “cat” command. Edit the “answers.txt” file to indicate the current “State” of the process and why do you think the State is that one.

The “/proc/<PID>/status” file shows you detailed summary information of the current status of the process. Among others, there is the PID and PPID, number of threads, memory information (it will be analyzed in further Lab sessions), and number of context switches. Regarding this last information, the OS distinguishes voluntary context switches (the process has called a blocking syscall before it finishes the quantum), and non-voluntary context switches (the OS put it out from the CPU, since the quantum was exhausted).

For the next exercise, copy one of the source codes (for example integers.cc) into a new file “integers-passive.cc”. Replace the main for loop by an infinite “while” loop with a sleep call (that will stall during X seconds the process execution) in every iteration. That is, “while(1) {sleep(1);}”.

Exercise 7

Launch the “integers” and the “integers-pasive” programs to be run in background. And compare the number of context switches shown by each of them. Additionally, compare these numbers to the number of context switches that the SHELL process shows. Edit the “answers.txt” file to briefly explain your findings.

Exercise 8 (OPTIONAL)

With your native Operating System, try to find a tool to obtain similar information than the “ps”, “top”, and “nice” commands, as well as the summary of process information from “/proc/<PID>/status”. Edit the “answers.txt” file to briefly explain what tool you have used it.

Upload the Deliverable

To save the changes you can use the tar command as follows:

```
#tar czvf session7.tar.gz answers.txt
```

Now go to RACO and upload this recently created file to the corresponding session slot.