UDACITY

< Return to "C++" in the classroom

Process Monitor

REVIEW

CODE REVIEW 8

HISTORY

Meets Specifications

Awesome work. 🎉✨🤙 You have nailed it. 👍👍👍

You have taken a big step ahead by learning and completing everything in this project. 🚀
I know it takes a really sincere effort to complete a project and you have done a great job. 👑
There is some great hard work here. You have really demonstrated your excellence. 😎

🎉 Awesome Things 🤙

- You have got everything perfect and this project is not simple and easy if you a beginner in C++.
- Every function seems to follow KISS(Keep It Small and Simple) Principle and that is really awesome. It is easy to write some complex code but it is way harder to write simple code that everyone can understand easily.
- You are following the consistent naming scheme and it is really good practice.
- The code is absolutely clean with correct indentation and a well-organized order for the functions.

All the best for your next project.😀 Crush it just like this one. 💪

PS - You can always attach a picture of your error (if there is any) in the future in a folder and then mention it in the notes to the reviewer section while submitting it.
That way we can help you in a better way. Take care of that in the future. Wish for a happy time having the next submission

Did you love the review? Let me know by giving a little bit of your precious time in rating.

Basic Requirements

The program must build an executable system monitor.

✔ Great, The program builds successfully

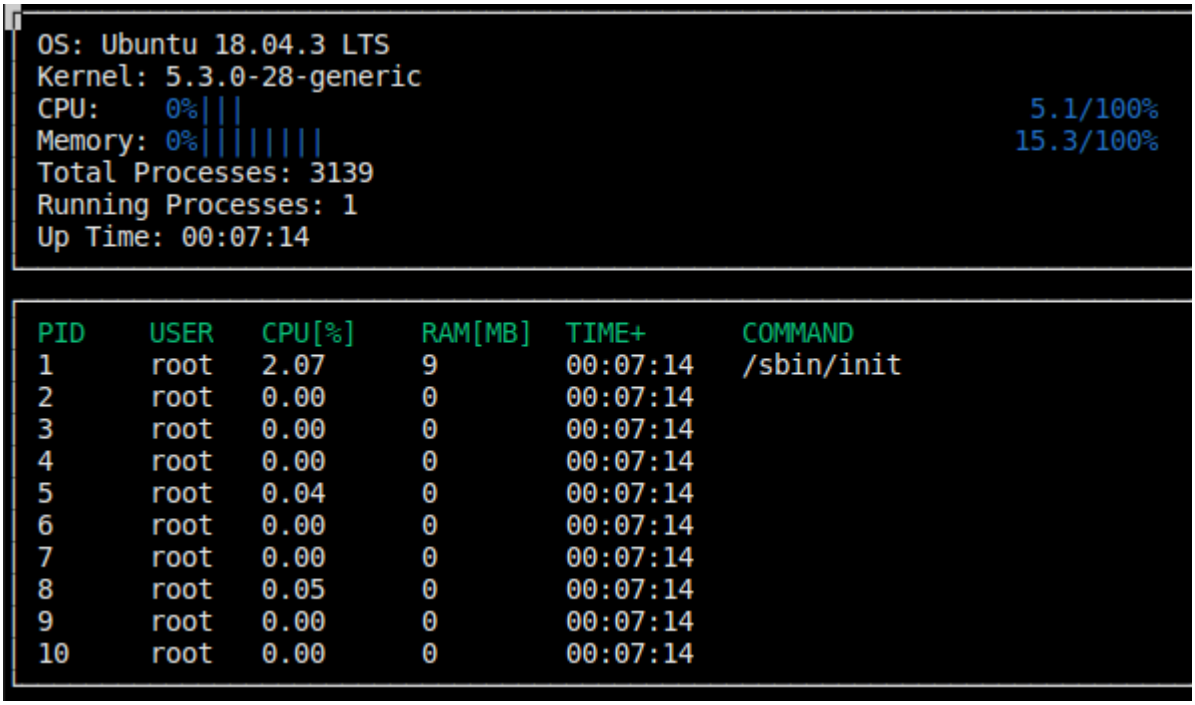
The program must build without generating compiler warnings.

✔ No compiler warnings

```
[ 12%] Building CXX object CMakeFiles/monitor.dir/src/format.cpp.o
[ 25%] Building CXX object CMakeFiles/monitor.dir/src/linux_parser.cpp.o
[ 37%] Building CXX object CMakeFiles/monitor.dir/src/main.cpp.o
[ 50%] Building CXX object CMakeFiles/monitor.dir/src/ncurses_display.cpp.o
[ 62%] Building CXX object CMakeFiles/monitor.dir/src/process.cpp.o
[ 75%] Building CXX object CMakeFiles/monitor.dir/src/processor.cpp.o
[ 87%] Building CXX object CMakeFiles/monitor.dir/src/system.cpp.o
[100%] Linking CXX executable monitor
```

The system monitor must run continuously without error, until the user terminates the program.

✔ The monitor runs without error



The project should be organized into appropriate classes.

✔ Well done! 👏👏
Your project is well organized and meets the professional developer standard.

System Requirements

The system monitor program should list at least the operating system, kernel version, total number of processes, number of running processes, and up time.

✔ Well done! 👏👏
I see all the data like operating system, kernel version, the total number of processes and number of running processes on the terminal easily.

The System class should be composed of at least one other class.

✔ Your project structure meets the rubric specifications since the class LinuxParser has been used to define most of the functions in System class.

Processor Requirements

The system monitor should display the CPU utilization.

✔ Your project displays the CPU utilization of the system.

Process Requirements

The system monitor should display a partial list of processes running on the system.

✔ As mentioned in the previous rubrics your projects display a partial list of processes running on the system.

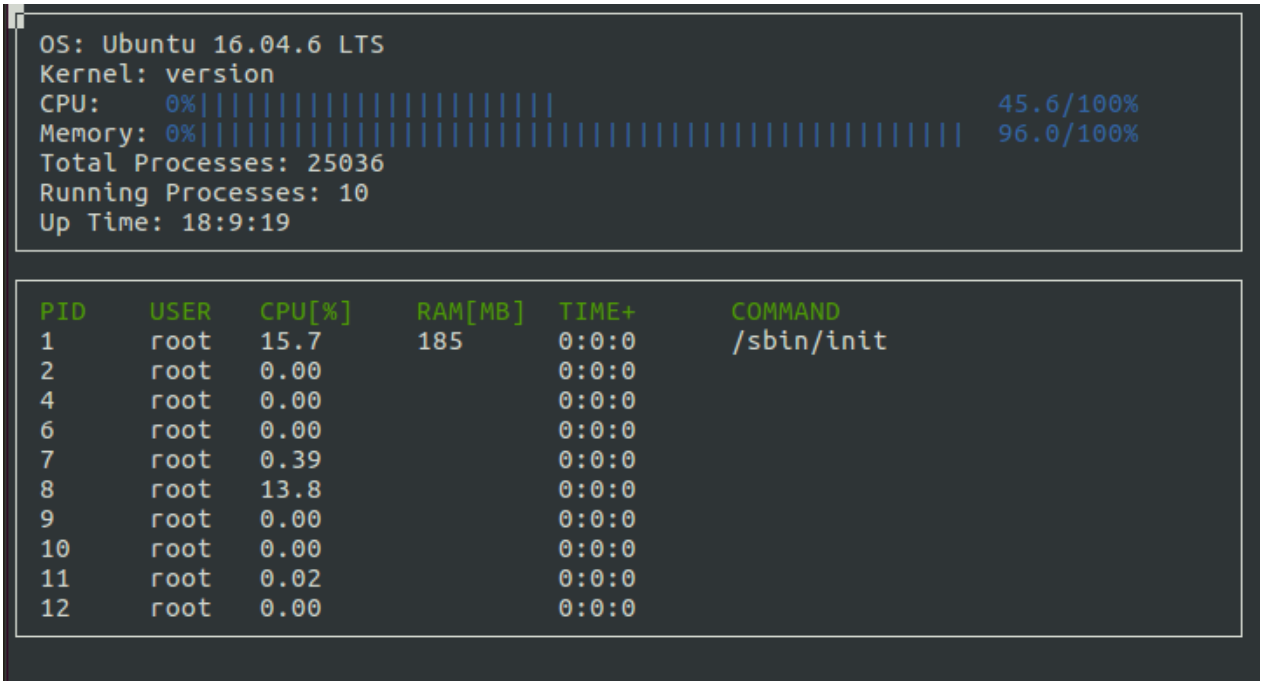
The system monitor should display the PID, user, CPU utilization, memory utilization, up time, and command for each process.

As far as I see your system monitor shows everything on the terminal (PID, user, CPU utilization, memory utilization, uptime, command)

Add To Knowledge

If you would not have sorted the processes according to any condition then they must have been in the increasing order of PID and then you must have seen PID number 2,3 and then all.

You would see something like this:



In that case, you would notice that you are not able to see the command for most of the processes neither the RAM usage. There is nothing wrong with your implementation of any function command or anything.

It so happens that when you select the processes the processes get stored in the vector sorted by PID and so when you take 10 PIDs from that list then first tens which are running lists out.

if you investigate the files associated with these PIDs then you will notice that the files in some of the PIDs are different than the files in some normal processes PIDs.
/proc/[pid]/cmdline
This read-only file holds the complete command line for the
process, unless the process is a zombie.
In the latter case, there is nothing in this file: that is,
a read on this file will return 0 characters. The command-line
arguments appear in this file as a set of strings separated by null
bytes ('\0'), with a further null byte after the last string.

Zombie Process:
A process in Unix or Unix-like operating systems becomes a zombie process when it has completed execution but one or some of its entries are still in the process table. If a process is ended by an "exit" call, all memory associated with it is reallocated to a new process; in this way, the system saves memory. But the process' entry in the process table remains until the parent process acknowledges its execution, after which it is removed. The time between the execution and the acknowledgment of the process is the period when the process is in a zombie state.

When a process dies on Linux, it isn't all removed from memory immediately — its process descriptor stays in memory (the process descriptor only takes a tiny amount of memory - That is why you can see the memory column is showing almost nothing at all).

A zombie process is also known as a defunct process.

PS - Do not worry if you do not understand some of the terms and it is perfectly normal because these terms are taught in a course Operating Systems a that too is a vast course in itself. I just gave you a glimpse so that you know that there is nothing wrong with your implementation and let you know why it is happening.

Once you sort the processes based on some properties (maybe based on the usage of CPU utilization) then vanishes away!

DOWNLOAD PROJECT

8 CODE REVIEW COMMENTS ❯

RETURN TO PATH

Rate this review