UDACITY

< Return to Classroom

Process Monitor

REVIEW

CODE REVIEW

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. 😎

Let's dig deeper into your project. ⏳

🎉 Awesome Things 🏆

- You have not given up and completed everything mentioned by the previous reviewer 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. e.g.

```
// TODO: Read and return the system uptime
long LinuxParser::UpTime(){
```

```
long LinuxParser::UpTime(){
string line, uptime, idletime;
long output;
vector<string> split_str;

// std::ifstream filestream(kMeminfoFilename); // open file
std::ifstream filestream("/proc/uptime"); // open file

if (filestream.is_open()) {
  // if file open is ok, do while.
  while(std::getline(filestream, line)){
    split_str = split(line, ' ');

    // get key & value
    if (split_str.size() == 2){
      uptime   = split_str[0];
      idletime = split_str[1];
      output   = (long)std::stoi(uptime);
      return output;
    }
  }
}

return 0;
}
```

This is a very clean solution.

- You are following the consistent naming scheme and it is really good practice.
- The code is absolutely clean with correct indentation and well-organized order for the functions.

Suggestions

- You should remove TODO comments and some unnecessary comments once you have completed the task. You should clean them up because it can make your code a lot cleaner.
- If you can see in the output, then you can see that RAM column represents value in MB but they do not seem right. do they? It's not your fault but it happens because Course Lesson mentions using "VmSize" for memory usage. But that is not right because Virtual Memory can be more than Physical Memory. So if you are curious about how to get the correct values for memory usage then you can use resident memory("VmRss"). If you explore the Htop source code, you will find that("M_RESIDENT" field). https://github.com/hishamhm/htop/tree/master/linux

Go Further

These resources can help you step up your game:

- https://google.github.io/styleguide/cppguide.html
- https://developers.google.com/edu/c++/
- https://techdevguide.withgoogle.com/resources/programming_languages/c-plus-plus/?programming_languages=c-plus-plus
- https://gist.github.com/lefticus/10191322
- https://chromium.googlesource.com/chromium/src/+/master/styleguide/c++/c++.md

Tools for C++:

- https://visualstudio.microsoft.com/vs/features/cplusplus/
- https://dev.to/dwd/tools-for-c-development
- https://blog.education-ecosystem.com/thetenbestccproductivitytoolspluginsandlibraries/

- https://codecondo.com/top-10-ide-for-c-and-cplusplus-for-programmers/
- http://www.stroustrup.com/compilers.html

Go deeper into the Project:

- https://github.com/hishamhm/htop

- https://github.com/GNOME/gnome-system-monitor
- https://github.com/brndnmtthws/conky
- https://github.com/topics/process-monitor?l=c%2B%2B&o=desc&s=forks
- https://docs.microsoft.com/en-us/windows/win32/psapi/process-status-helper

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

Basic Requirements

The program must build an executable system monitor.

The program must build without generating compiler warnings.

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

The project should be organized into appropriate classes.

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.

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

Processor Requirements

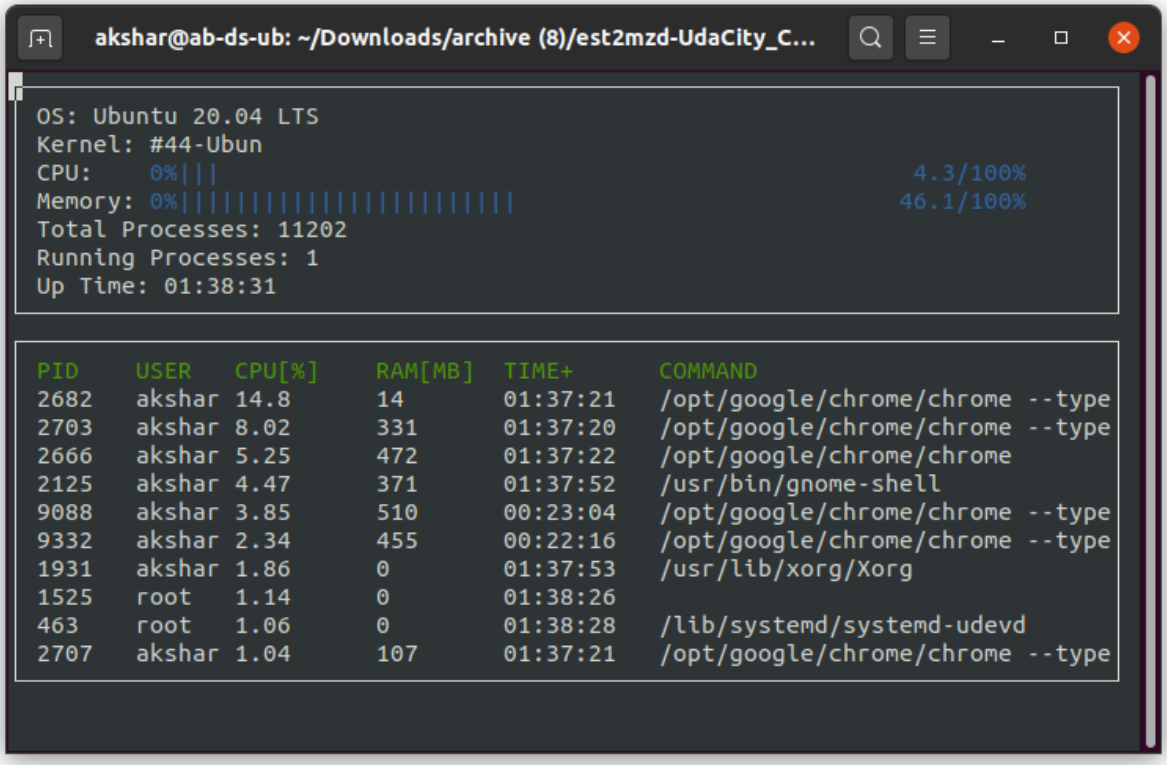
The system monitor should display the CPU utilization.

Process Requirements

The system monitor should 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.

It compiles and runs flawlessly.



DOWNLOAD PROJECT

RETURN TO PATH