CSE-381: Systems 2
Homework #7

Due: Tuesday Nov 27 2018 before 11:59 PM

Email-based help Cutoff: 5:00 PM on Sun, May 6 2018

Maximum Points: 35

Submission Instructions

This homework assignment must be turned-in electronically via Canvas. Ensure your C++ source code is named `MUID_hw7.cpp`, where `MUID` is your Miami University Unique ID. Ensure your program compiles without any warnings or style violations. Ensure you thoroughly test operations of your program as indicated. Once you have tested your implementation, upload the following onto Canvas:

1. The 1 C++ source file developed for this homework.

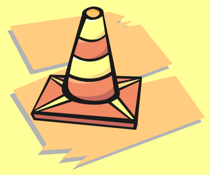
General Note: Upload each file associated with homework individually to Canvas. Do not upload archive file formats such as zip/tar/gz/7zip/rar etc.

Objective

The objective of this homework is to:

- Do some self-research/learning
- Understand the use of `/proc` file system
- Review the use of server sockets for communication
- Extend a C++ program that can be used as a web server
- Develop a web server to run programs & monitor programs

Grading Rubric:



The program submitted for this homework **must pass necessary base case test(s) in order to qualify for earning any score at all**. Programs that do not meet base case requirements will be assigned zero score!
Program that do not compile, have a method longer than 25 lines, or just some skeleton code will be assigned zero score.

- See points set for each command and overall structure, organization, **conciseness**, variable-names, etc. in the next page. **If your methods are not concise points will be deducted**.
- **-1 Points**: for each warning generated by the compiler (warnings are most likely sources of errors in C++ programs)

NOTE: Violating CSE programming style guidelines is a compiler error! Your program should not have any style violations.

Background

The /proc file system is a logical view of internal data that is maintained by the Linux kernel about a process. For a given process the kernel exposes a logical file called `/proc/`**`[pid]`**`/stat` (where **[`pid`]**is the process ID, e.g., `/proc/235/stat`). This `stats` file contains a variety of statistics about the process that can be used to report runtime statistics about the process. In this homework, you will be using the following 3 values as described in the following URL: http://man7.org/linux/man-pages/man5/proc.5.html

Word number in stats file	**Logical name for Value**	**Notes**
14th word	utime (`float`)	Cumulative user time. This value must be divided by divide by `sysconf(_SC_CLK_TCK)` to get value in seconds
15th word	stime (`float`)	Cumulative system time. This value must be divided by divide by `sysconf(_SC_CLK_TCK)` to get value in seconds
23rd word	vsize (`long`)	Current virtual memory size in bytes. This value must be divided by 1000 to get memory size in KB.

Starter code

The starter code for this homework is just the solution for Exercise #9. Invest time to refresh your memory by reviewing the starter code.

Requirements

Suitably extend (how you modify is up to you) the starter source code to perform the following additional functionality:

1. Base case -- Run a program & record runtime statistics [**22 points**]: Run a specified command with suitable arguments and return the output and runtime statistics in the HTML format shown further below. Recording of runtime statistics is:
    a. Outputs are sent line-by-line to the client using chunked encoding
    b. Record and print runtime statistics every second by opening & reading `/proc/[pid]/stat`
    c. Reading and printing statistics each second for the duration of a child process can be accomplished as suggested below:

```
int exitCode = 0;
while (waitpid(pid, &exitCode, WNOHANG) == 0) {
    sleep(1);  // sleep for 1 second.
   // Record current statistics about the process.
}
```

    d. It is best to run the above loop in a separate thread to record the necessary statistics.

    e. The recorded statistics is finally sent to the client as a big chunk at the end of program execution. No special formatting was needed to get the statistics/numbers to be printed as shown in expected outputs.

! The base case is relatively straightforward with plenty of time to finish. Consequently, if the base case does not operate as expected, as per the course policy, the whole program will be assigned zero score.

2. Setup statistics to plot a chart [**8 points**]: If the `genChart` parameter (3rd parameter) to the `serveClient` method is `true`, then the program should also generate the runtime statistics as a JSON array in the following format:

   ```
   [ second, utime+stime, vsize ]
   ```

   See annotated HTML in base case output below for details.

   Note: This output should use exactly the same statistics values from base case. Do not re-read the stats file to generate the statistics in JSON output.

3. Code quality [**5 points**]: Five points in this homework are reserved for good code quality, effective code reuse, and documentation (in the form of comments). **These 5 points would be the hardest to earn in this homework**.

Functional testing

The supplied starter code already includes a simple test harness that will also be used for testing and grading purposes. Several test inputs and expected output files are also supplied to streamline your testing as shown below. The `true/false` flag (*i.e.*, last command-line argument) is used to enable/disable generation of JSON format used to plot charts.

```
$ ./hw7 base_case1_inputs.txt my_base_case1_output.txt false
$ diff my_base_case1_output.txt base_case1_expected_outputs.txt
```

Testing for graph case:

```
$ ./hw7 base_case1_inputs.txt my_graph_case1_output.txt true
$ diff my_graph_case1_output.txt graph_case1_expected_outputs.txt
```

Needless to add, the above `diff` commands will not produce any output if the outputs are identical.

! Due to non-deterministic scheduling and multiple users on the machine, the CPU time and virtual memory use could be a bit different in some cases. Minor differences in values are acceptable.

Ensure you test with input files and corresponding expected outputs prior to submission.

Browser testing

You may test operation of your webserver running on `os1.csi.miamioh.edu`. A partial screenshot is shown further below for your reference (you may have to use a different `port` number instead of `4000`).

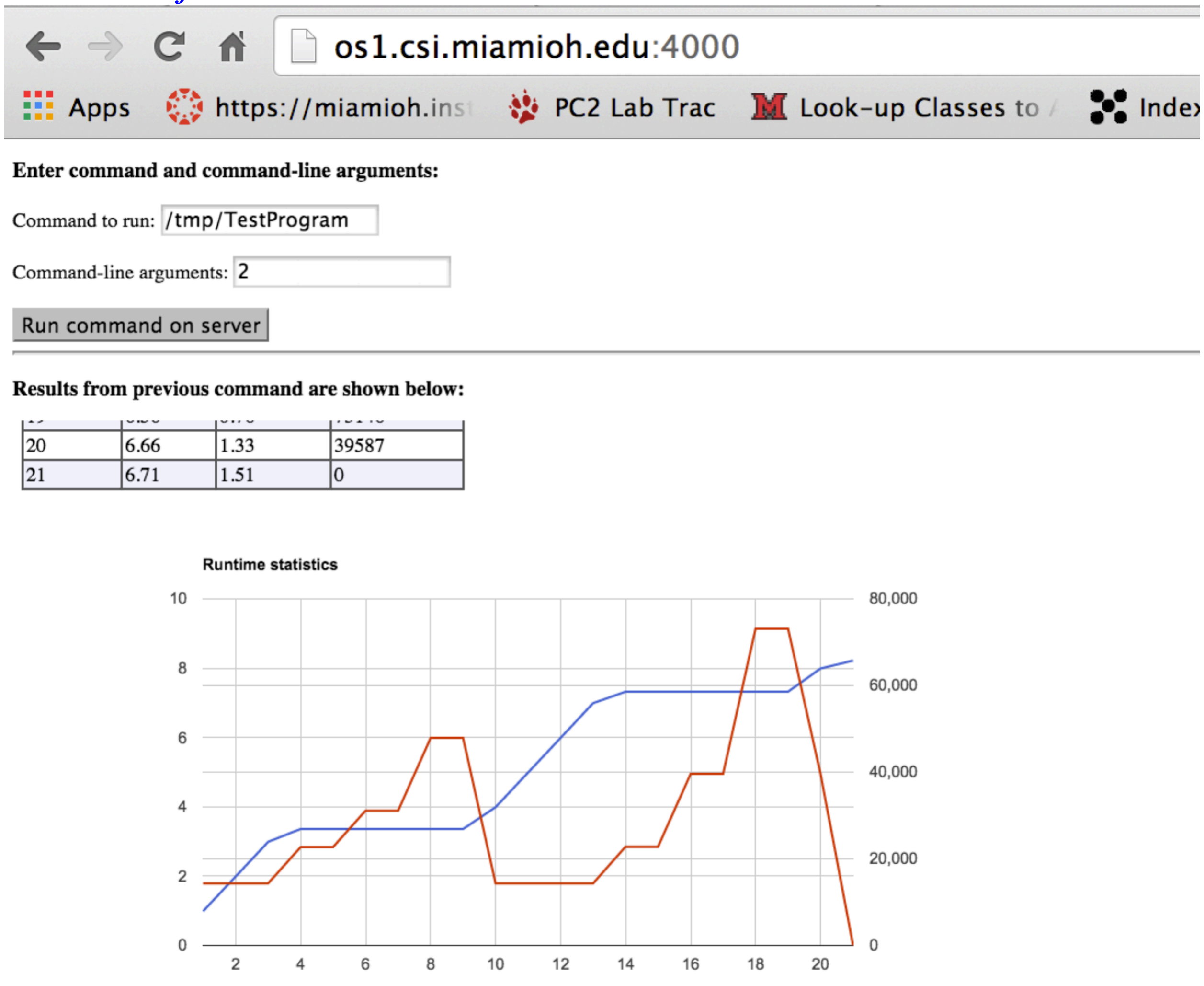
Submit to Canvas

This homework assignment must be turned-in electronically via Canvas. Ensure your C++ source files are named appropriately. Ensure your program compiles (without any warnings or style errors) successfully. Ensure you have tested operations of your program as indicated. Once you have tested your implementation, upload the following onto Canvas:

- The 1 C++ source file you developed for this homework

Upload all the necessary C++ source files to onto Canvas independently. Do not submit zip/7zip/tar/gzip files. Upload each file independently.

Screenshot of browser test:



Enter command and command-line arguments:

Command to run: /tmp/TestProgram

Command-line arguments: 2

Results from previous command are shown below:

20	6.66	1.33	39587
21	6.71	1.51	0

Base case #2 Annotated output HTML

Parts of the HTML highlighted in gray are fixed/constant strings that can be hardcoded in your program. Values highlighted in red are chunk sizes in hexadecimal notation.

```
HTTP/1.1 200 OK
Content-Type: text/html
Transfer-Encoding: chunked
Connection: Close

156
<html>
  <head>
    <script type='text/javascript'
src='https://www.gstatic.com/charts/loader.js'></script>
    <script type='text/javascript' src='/draw_chart.js'></script>
    <link rel='stylesheet' type='text/css' href='/mystyle.css'>
  </head>

  <body>
    <h3>Output from program</h3>
    <textarea style='width: 700px; height: 200px'>

1b
Using CPU for computation.

1d
Using memory for operations.

1b
Using CPU for computation.

1d
Using memory for operations.

10

Exit code: 0

742
     </textarea>
     <h2>Runtime statistics</h2>
     <table>
       <tr><th>Time (sec)</th><th>User time</th><th>System time</th><th>Memory
(KB)</th></tr>
       <tr><td>1</td><td>0.99</td><td>0</td><td>14336</td></tr>
       <tr><td>2</td><td>1.99</td><td>0</td><td>14336</td></tr>
       <tr><td>3</td><td>2.99</td><td>0</td><td>14336</td></tr>
       <tr><td>4</td><td>3.36</td><td>0</td><td>22728</td></tr>
       <tr><td>5</td><td>3.36</td><td>0</td><td>22728</td></tr>
       <tr><td>6</td><td>3.36</td><td>0</td><td>31117</td></tr>
       <tr><td>7</td><td>3.36</td><td>0</td><td>31117</td></tr>
       <tr><td>8</td><td>3.36</td><td>0</td><td>47894</td></tr>
       <tr><td>9</td><td>3.36</td><td>0</td><td>47894</td></tr>
```

```
        <tr><td>10</td><td>3.45</td><td>0.54</td><td>14336</td></tr>
        <tr><td>11</td><td>4.25</td><td>0.74</td><td>14336</td></tr>
        <tr><td>12</td><td>5.25</td><td>0.74</td><td>14336</td></tr>
        <tr><td>13</td><td>6.25</td><td>0.74</td><td>14336</td></tr>
        <tr><td>14</td><td>6.56</td><td>0.74</td><td>22810</td></tr>
        <tr><td>15</td><td>6.56</td><td>0.74</td><td>22810</td></tr>
        <tr><td>16</td><td>6.56</td><td>0.74</td><td>39587</td></tr>
        <tr><td>17</td><td>6.56</td><td>0.74</td><td>39587</td></tr>
        <tr><td>18</td><td>6.56</td><td>0.74</td><td>73146</td></tr>
        <tr><td>19</td><td>6.56</td><td>0.74</td><td>73146</td></tr>
        <tr><td>20</td><td>6.69</td><td>1.29</td><td>39587</td></tr>
        <tr><td>21</td><td>6.72</td><td>1.51</td><td>0</td></tr>
      </table>
      <div id='chart' style='width: 900px; height: 500px'></div>
  </body>
  <script type='text/javascript'>
    function getChartData() {
      return google.visualization.arrayToDataTable(
        [
          ['Time (sec)', 'CPU Usage', 'Memory Usage'],

        ]
      );
    }
  </script>
</html>

0
```