

[◀ Return to Classroom](#)

# Process Monitor

## REVIEW

## CODE REVIEW 5

## HISTORY

### ▼ CppND-System-Monitor/src/linux\_parser.cpp 2

```
1 #include <dirent.h>
2 #include <unistd.h>
3 #include <string>
4 #include <vector>
5
6 #include "linux_parser.h"
7 #include "my_utility.h"
8
9 using std::stof;
10 using std::string;
11 using std::to_string;
12 using std::vector;
13
14 // DONE: An example of how to read data from the filesystem
15 string LinuxParser::OperatingSystem() {
16     string line;
17     string key;
18     string value;
19     std::ifstream filestream(kOSPath);
20     if (filestream.is_open()) {
21         while (std::getline(filestream, line)) {
22             std::replace(line.begin(), line.end(), ' ', '_');
23             std::replace(line.begin(), line.end(), '=', ' ');
24             std::replace(line.begin(), line.end(), '"', ' ');
25             std::istringstream linestream(line);
26             while (linestream >> key >> value) {
27                 if (key == "PRETTY_NAME") {
28                     std::replace(value.begin(), value.end(), '_', ' ');
29                     return value;
30                 }
31             }
32         }
33     }
34     return value;
35 }
36
37
38 // DONE: An example of how to read data from the filesystem
```

```

39 string LinuxParser::Kernel() {
40     string os, kernel;
41     string line, output;
42     vector<string> split_str;
43     std::ifstream stream(kProcDirectory + kVersionFilename);
44     if (stream.is_open()) {
45         std::getline(stream, line);
46         //-----
47         if (1){
48             std::istringstream linestream(line);
49             linestream >> os >> kernel;
50             output = kernel;
51         }else{
52             split_str = split(line, ' ');
53             output = (split_str[8] + split_str[9]);
54         }

```

## REQUIRED

Your kernel version is not able to show in the output. You need to fix this part.

```

55     //-----
56 }
57 return output;
58 }
59
60
61 // BONUS: Update this to use std::filesystem
62 vector<int> LinuxParser::Pids() {
63     vector<int> pids;
64     DIR* directory = opendir(kProcDirectory.c_str());
65     struct dirent* file;
66     while ((file = readdir(directory)) != nullptr) {
67         // Is this a directory?
68         if (file->d_type == DT_DIR) {
69             // Is every character of the name a digit?
70             string filename(file->d_name);
71             if (std::all_of(filename.begin(), filename.end(), isdigit)) {
72                 int pid = stoi(filename);
73                 pids.push_back(pid);
74             }
75         }
76     }
77     closedir(directory);
78     return pids;
79 }
80
81
82 // TODO: Read and return the system memory utilization
83 float LinuxParser::MemoryUtilization() {
84     string line;
85     vector<string> split_str;
86     float mem_total, mem_available, mem_used;
87     float mem_utilization;
88
89     // std::ifstream filestream(kMeminfoFilename); // open file
90     std::ifstream filestream("/proc/meminfo"); // open file
91
92     if (filestream.is_open()) {
93         // if file open is ok, do while.
94         while(std::getline(filestream, line)){
95             int pos = line.find(":");
96
97             if (pos != (int)string::npos){
98                 // delete space from line
99                 line = del_space(line);
100
101                 // split line with ":"
102                 split_str = split(line, ':');
103

```

```

104         // get key & value
105         if (split_str.size() == 2) {
106             if (split_str[0] == "MemTotal") {
107                 mem_total = std::stof(trim_rear(split_str[1], 2));
108             }
109             if (split_str[0] == "MemAvailable") {
110                 mem_available = std::stof(trim_rear(split_str[1], 2));
111                 mem_used = mem_total - mem_available;
112                 mem_utilization = (float)(mem_used / mem_available);
113             }
114             return mem_utilization;
115         }
116     }
117 }
118 }
119 }
120 return 0.0;
121 }
122
123
124 // TODO: Read and return the system uptime
125 long LinuxParser::UpTime() {
126     string line, uptime, idletime;
127     long output;
128     vector<string> split_str;
129
130     // std::ifstream filestream(kMeminfoFilename); // open file
131     std::ifstream filestream("/proc/uptime"); // open file
132
133     if (filestream.is_open()) {
134         // if file open is ok, do while.
135         while (std::getline(filestream, line)) {
136             split_str = split(line, ' ');
137
138             // get key & value
139             if (split_str.size() == 2) {
140                 uptime = split_str[0];
141                 idletime = split_str[1];
142                 output = (long)std::stoi(uptime);
143                 return output;
144             }
145         }
146     }
147
148     return 0;
149 }
150
151
152
153 // TODO: Read and return the number of jiffies for the system
154 long LinuxParser::Jiffies() {
155     return LinuxParser::UpTime() * sysconf(_SC_CLK_TCK);
156 }
157
158 // TODO: Read and return the number of active jiffies for a PID
159 // REMOVE: [[maybe_unused]] once you define the function
160 long LinuxParser::ActiveJiffies(int pid) {
161     vector<string> split_str;
162     string line;
163     string file_path = "/proc/" + std::to_string(pid) + "/stat";
164     long utime, stime, cutime, cstime, active_jiffies;
165     std::ifstream filestream(file_path);
166
167     if (filestream.is_open()) {
168         while (std::getline(filestream, line)) {
169             split_str = split(line, ' ');
170
171             if (split_str.size() >= 17) {
172                 utime = std::stol(split_str[13]);
173                 stime = std::stol(split_str[14]);
174                 cutime = std::stol(split_str[15]);
175                 cstime = std::stol(split_str[16]);

```

```

176         active_jiffies = utime + stime + cutime + cstime;
177
178         return active_jiffies;
179     }
180 }
181 }
182
183 return 0;
184 }
185
186 // TODO: Read and return the number of active jiffies for the system
187 long LinuxParser::ActiveJiffies() {
188     vector<string> split_str;
189     string line;
190     long active_jiffies = 0;
191     std::ifstream filestream("/proc/stat");
192
193     if(filestream.is_open()){
194         while(std::getline(filestream, line)){
195             split_str = split(line, ' ');
196             if (split_str.size() == 11){
197
198                 if (split_str[0] == "cpu"){
199                     for(int i=1; i<=10; i++){
200                         // sum of all values without idle & iowait
201                         if ( (i!=4) && (i!=5)){
202                             active_jiffies += std::stol(split_str[i]);
203                         }
204                     }
205                     return active_jiffies;
206                 }
207             }
208         }
209     }
210     return 0;
211 }
212
213 // TODO: Read and return the number of idle jiffies for the system
214 long LinuxParser::IdleJiffies() {
215     vector<string> split_str;
216     string line;
217     long active_jiffies = 0;
218     std::ifstream filestream("/proc/stat");
219
220     if(filestream.is_open()){
221         while(std::getline(filestream, line)){
222             split_str = split(line, ' ');
223             if (split_str.size() == 11){
224
225                 if (split_str[0] == "cpu"){
226                     for(int i=1; i<=10; i++){
227                         // sum of 2 values which are idle & iowait
228                         if ( (i==4) || (i==5)){
229                             active_jiffies += std::stol(split_str[i]);
230                         }
231                     }
232                     return active_jiffies;
233                 }
234             }
235         }
236     }
237     return 0;
238 }
239
240
241
242
243 // TODO: Read and return CPU utilization
244 float LinuxParser::CpuUtilization() {
245     string line;
246     vector<string> split_str;
247     float user, nice, system, idle, iowait, irq, softirq, steal;

```

```

248 //float guest, guest_nice;
249 float sum_idle, sum_non_idle, total;
250 float cpu_util;
251
252 std::ifstream filestream("/proc/stat");
253
254 if (filestream.is_open()){
255     while(std::getline(filestream, line)){
256         split_str = split(line, ' ');
257         if (split_str[0] == "cpu"){
258             user = std::stof(split_str[1]);
259             nice = std::stof(split_str[2]);
260             system = std::stof(split_str[3]);
261             idle = std::stof(split_str[4]);
262             iowait = std::stof(split_str[5]);
263             irq = std::stof(split_str[6]);
264             softirq = std::stof(split_str[7]);
265             steal = std::stof(split_str[8]);
266             //guest = std::stof(split_str[9]);
267             //guest_nice = std::stof(split_str[10]);
268             //
269             sum_idle = idle + iowait;
270             sum_non_idle = user + nice + system + irq + softirq + steal;
271             total = sum_idle + sum_non_idle;
272             cpu_util = (total - sum_idle) / total;
273
274             return cpu_util;
275         }
276     }
277 }
278 return 0.0;
279 }
280
281 // TODO: Read and return the total number of processes
282 int LinuxParser::TotalProcesses() {
283     vector<string> split_str;
284     string line;
285     std::ifstream filestream("/proc/stat");
286
287     if(filestream.is_open()){
288         while(std::getline(filestream, line)){
289             int pos = line.find("processes ");
290             if (pos != (int)string::npos){
291                 split_str = split(line, ' ');
292                 return std::stoi(split_str[1]);
293             }
294         }
295     }
296     return 0;
297 }
298
299 // TODO: Read and return the number of running processes
300 int LinuxParser::RunningProcesses() {
301     vector<string> split_str;
302     string line;
303     std::ifstream filestream("/proc/stat");
304
305     if(filestream.is_open()){
306         while(std::getline(filestream, line)){
307             int pos = line.find("procs_running ");
308             if (pos != (int)string::npos){
309                 split_str = split(line, ' ');
310                 return std::stoi(split_str[1]);
311             }
312         }
313     }
314     return 0;
315 }
316
317
318 // TODO: Read and return the command associated with a process
319 // REMOVE: [[maybe_unused]] once you define the function

```

```

320 string LinuxParser::Command(int pid) {
321     string line;
322     string file_path = "/proc/" + std::to_string(pid) + "/cmdline";
323     std::ifstream filestream(file_path);
324     if (std::getline(filestream, line)){
325         return line;
326     }
327     return " ";
328 }
329
330 // TODO: Read and return the memory used by a process
331 // REMOVE: [[maybe_unused]] once you define the function
332 float LinuxParser::Ram(int pid) {
333     string line;
334     vector<string> split_str;
335     float sum_rss = 0.0;
336     string file_path = "/proc/" + std::to_string(pid) + "/smaps";
337     std::ifstream filestream(file_path);
338
339     if (filestream.is_open()) {
340         while(std::getline(filestream, line)){
341             line = del_space(line);
342             split_str = split(line, ':');
343             if (split_str[0] == "Rss"){
344                 sum_rss += std::stof( trim_rear(split_str[1], 2) );
345             }
346         }
347         return sum_rss;
348     }
349     return 0.0;
350 }
351
352 // TODO: Read and return the user ID associated with a process
353 // REMOVE: [[maybe_unused]] once you define the function
354 int LinuxParser::Uid(int pid) {
355     vector<string> split_str;
356     string line;
357     string file_path = "/proc/" + std::to_string(pid) + "/status";
358     std::ifstream filestream(file_path);
359
360     if(filestream.is_open()){
361         while(std::getline(filestream, line)){
362             // split_str = split(line, ' ');
363             char str_tab = '\t';
364             split_str = split(line, str_tab);
365
366             if(split_str.size() >= 2){
367                 if (split_str[0] == "Uid:"){
368                     return std::stoi(split_str[1]);
369                 }
370             }
371         }
372     }
373
374     return 0;
375 }
376
377 // TODO: Read and return the user associated with a process
378 // REMOVE: [[maybe_unused]] once you define the function
379 string LinuxParser::User(int pid) {
380     vector<string> split_str;
381     string line;
382     string uid = std::to_string(LinuxParser::Uid(pid));
383     string file_path = "/etc/passwd";
384     std::ifstream filestream(file_path);
385
386     if(filestream.is_open()){
387         while(std::getline(filestream, line)){
388             // cout << "line = " << line << "\n";
389             split_str = split(line, ':');
390
391             if (split_str.size() >= 3){

```

```

392         if (split_str[2] == uid) {
393             return split_str[0];
394         }
395     }
396 }
397 }
398
399 return "";
400 }
401
402 // TODO: Read and return the uptime of a process
403 // REMOVE: [[maybe_unused]] once you define the function
404 long LinuxParser::UpTime(int pid) {

```

#### SUGGESTION

Uptime was showing static output for processes. It should be updated timely as known from its function name.

Expected Output:

```

OS: Ubuntu 19.10
Kernel: 5.3.0-26-generic
CPU:    0%|||||
Memory: 0%|||||
Total Processes: 6338
Running Processes: 1
Up Time: 0:14:39

```

PID	USER	CPU[%]	RAM[MB]	TIME+	COMMAND
2591	workspa	3.86	3287	0:12:56	/usr/lib
1557	workspa	6.37	3139	0:14:7	/usr/bin
3156	workspa	0.56	2556	0:14:39	/usr/lib
1319	workspa	0.00	2540	0:14:9	/usr/bin
2653	workspa	0.79	2524	0:14:39	/usr/lib
2706	workspa	0.00	2396	0:12:55	/usr/lib
4090	workspa	0.00	2379	0:14:39	/usr/lib
3286	workspa	5.02	2092	0:10:17	/home/wor
3272	workspa	11.5	1672	0:10:21	/usr/bin
700	root	0.34	1513	0:14:31	/usr/lib

```

405     vector<string> split_str;
406     string line;
407     string file_path = "/proc/" + std::to_string(pid) + "/stat";
408     std::ifstream filestream(file_path);
409     if (filestream.is_open()) {
410         while (std::getline(filestream, line)) {
411             split_str = split(line, ' ');
412             if (split_str.size() >= 22) {
413                 return std::stol(split_str[21]);
414             }
415         }
416     }
417     return 0;
418 }

```

- CppND-System-Monitor/src/system.cpp 1
- CppND-System-Monitor/src/process.cpp 1
- CppND-System-Monitor/src/format.cpp 1
- CppND-System-Monitor/src/test07.cpp
- CppND-System-Monitor/src/test06.cpp
- CppND-System-Monitor/src/test05.cpp
- CppND-System-Monitor/src/test04.cpp
- CppND-System-Monitor/src/test03.cpp
- CppND-System-Monitor/src/test02.cpp
- CppND-System-Monitor/src/test01.cpp
- CppND-System-Monitor/src/test00.cpp
- CppND-System-Monitor/src/processor.cpp
- CppND-System-Monitor/src/ncurses\_display.cpp
- CppND-System-Monitor/src/my\_utility.h
- CppND-System-Monitor/src/my\_utility.cpp
- CppND-System-Monitor/src/main.cpp
- CppND-System-Monitor/include/system.h
- CppND-System-Monitor/include/processor.h
- CppND-System-Monitor/include/process.h
- CppND-System-Monitor/include/ncurses\_display.h
- CppND-System-Monitor/include/linux\_parser.h
- CppND-System-Monitor/include/format.h
- CppND-System-Monitor/README.md
- CppND-System-Monitor/Makefile



Learn the [best practices for revising and resubmitting your project](#).

RETURN TO PATH

---