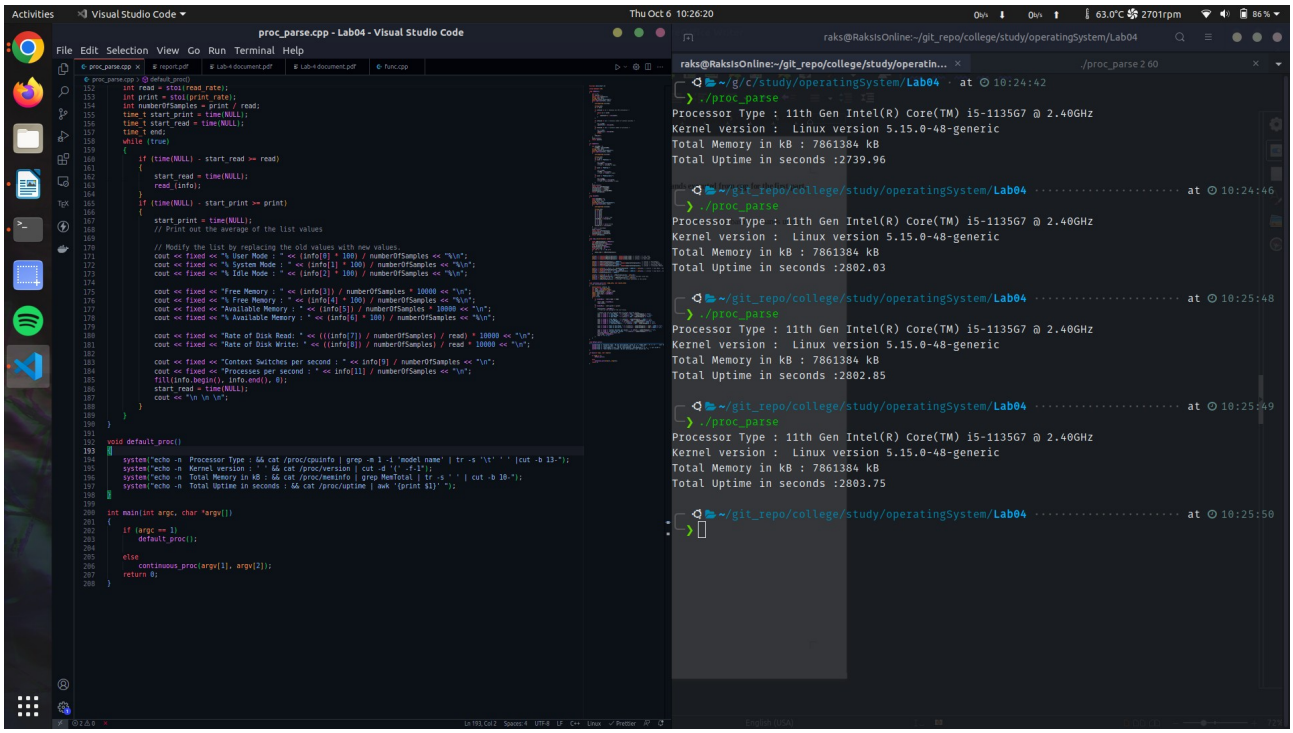
The following program proc_parse uses bash commands executed from cpp for the first part, and in the next part, it uses file handling .

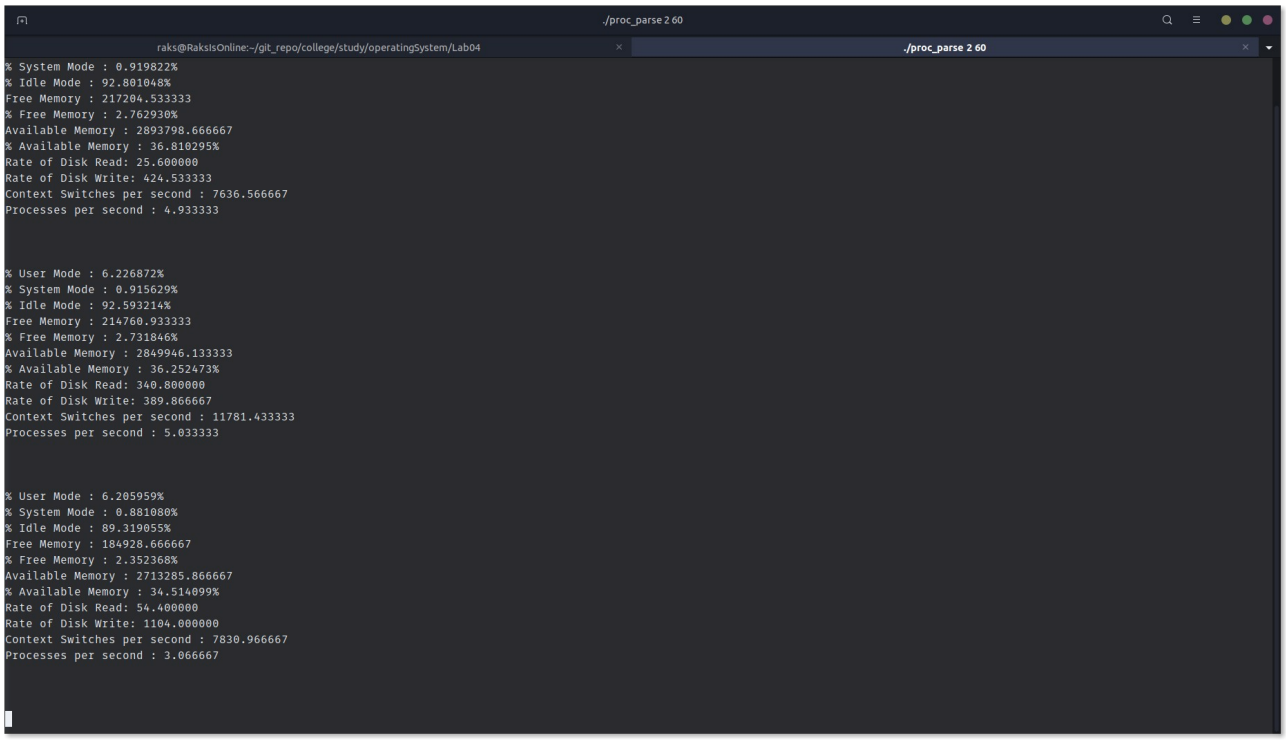
1st Version:
If no parameters are passed while execution, the program executes certain bash commands using the *system(<args>)* function.

Here's the output as well as the code snippet to display the same.



The bash commands used are – ***cat, awk, tr,*** and ***grep.***

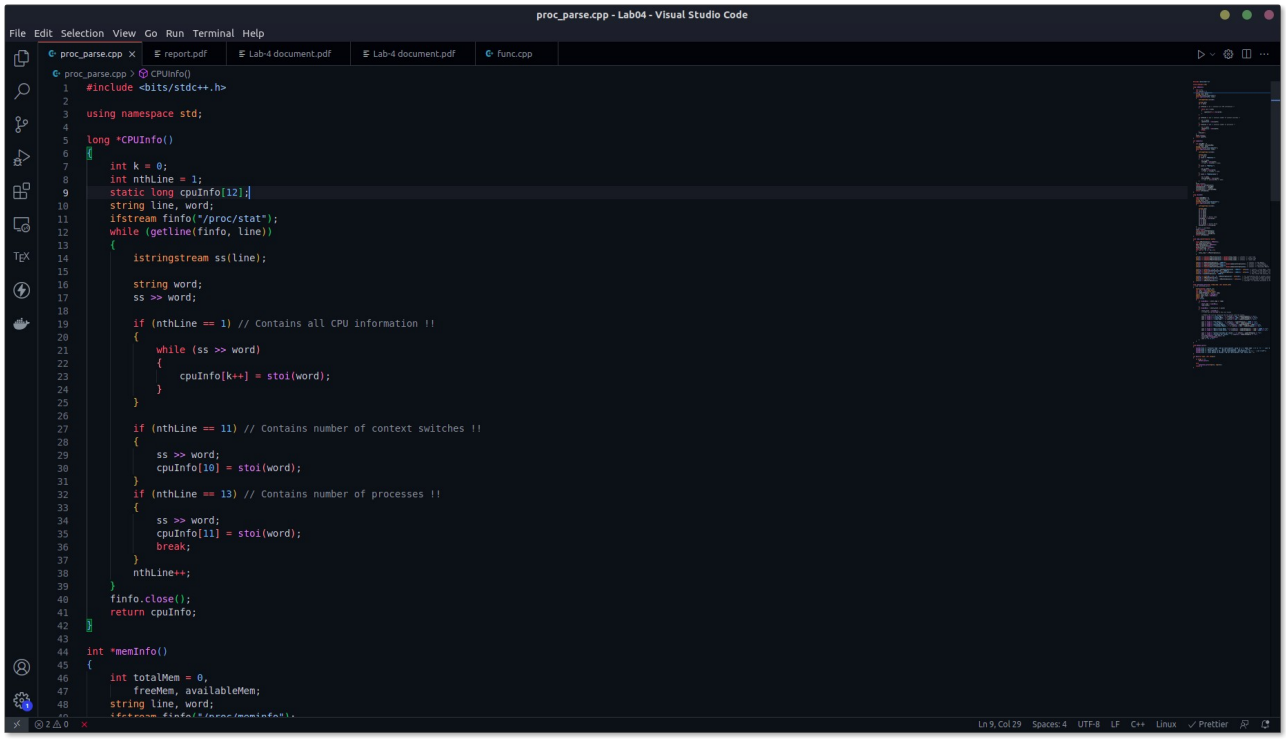
2nd Version:
As far as the second version of the code is concerned, the output are as follows :

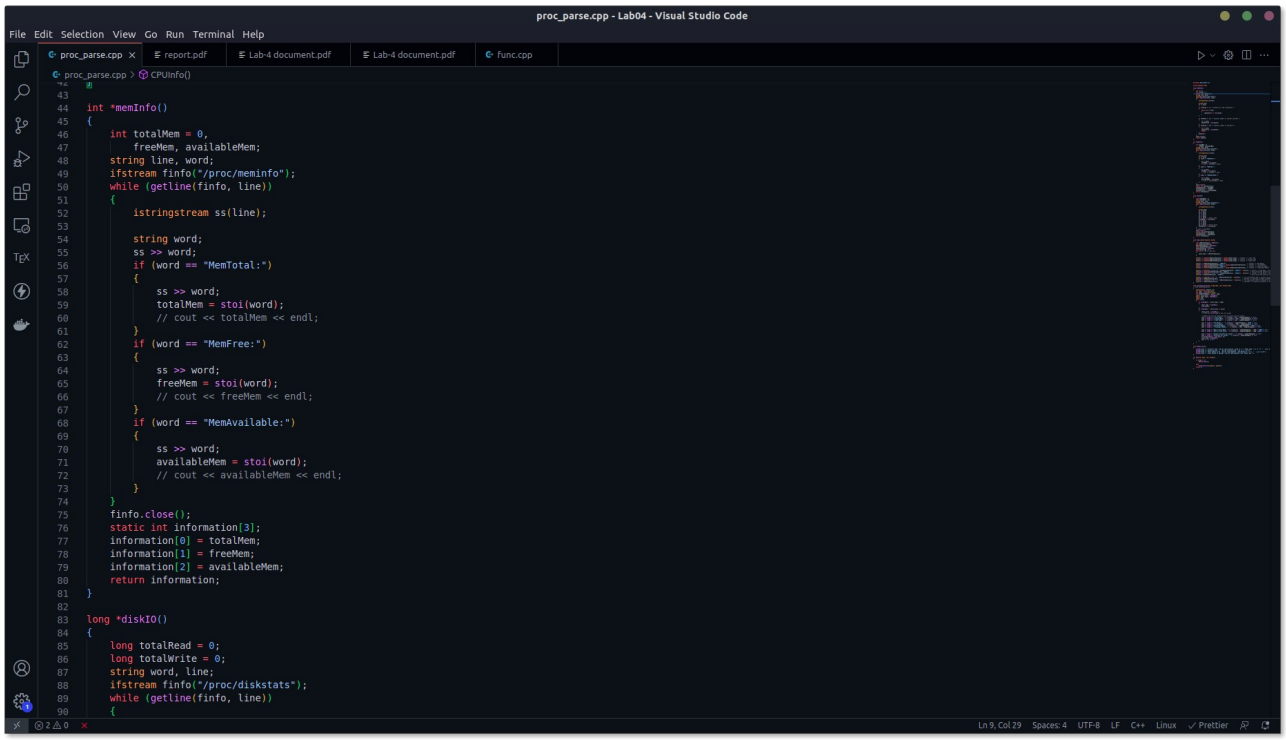


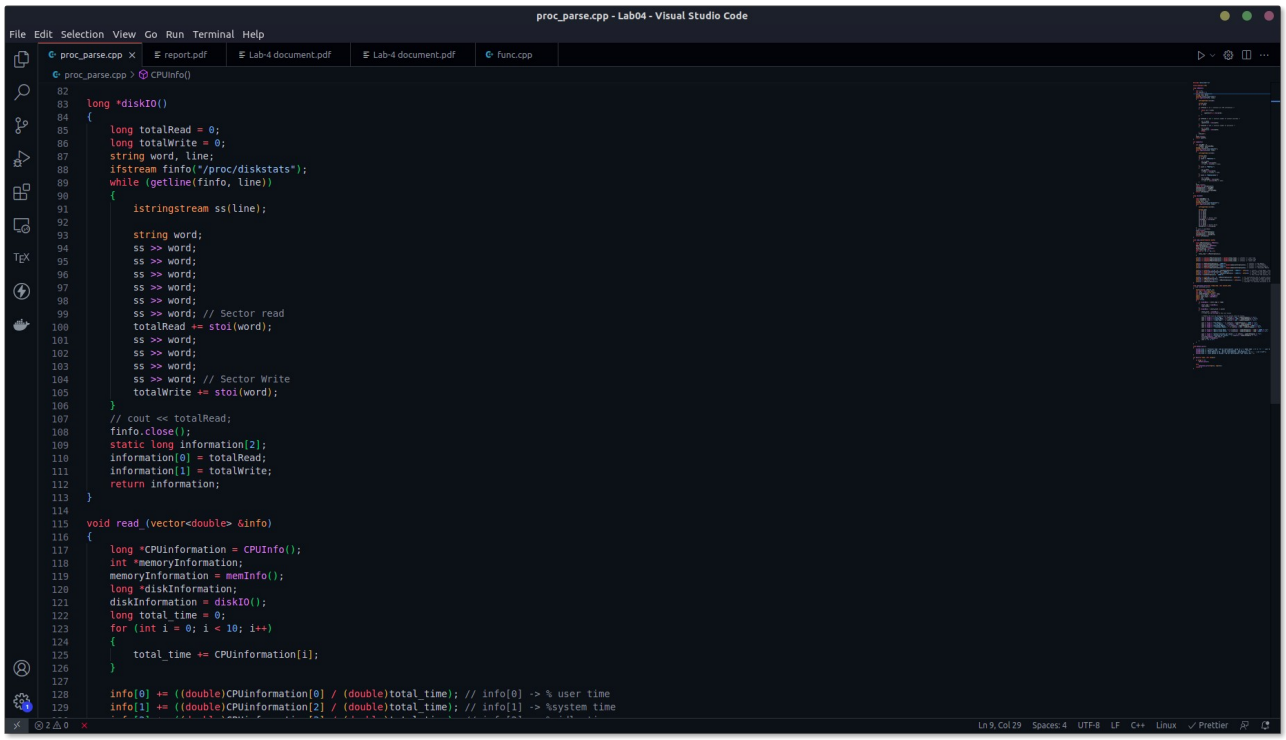
For this part of the code, command line arguments have been used to determine the print and read rate. After that, various */proc* files are read via *ifstream()* to get values of the above. Percentages are calculated, rate is calculated through the read rate and number of samples, etc. Since few values were provided with regards to boot-time, previous values were stored to calculate number of new processes/context switches produced.

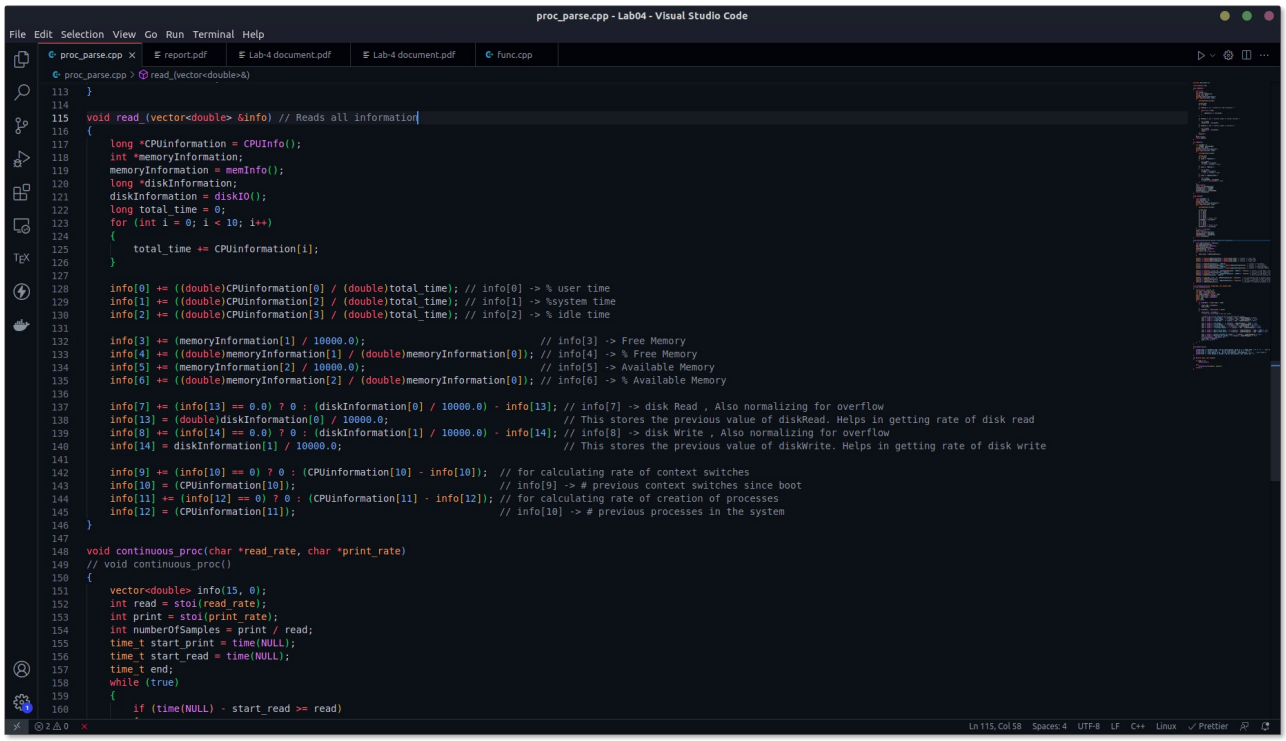
Time was calculated using *time()* function to keep a track of the read rate and print rate. Number of Samples is basically the ratio of print rate and read rate.

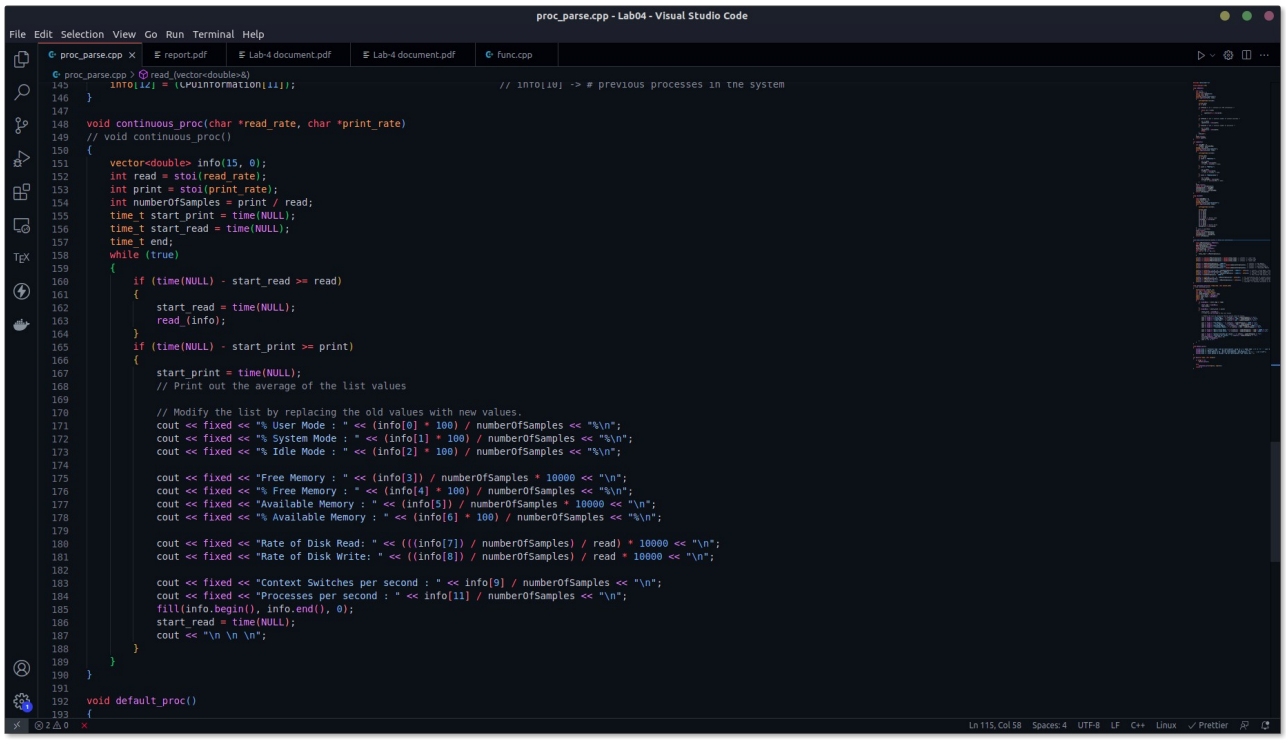
The code is as follows :

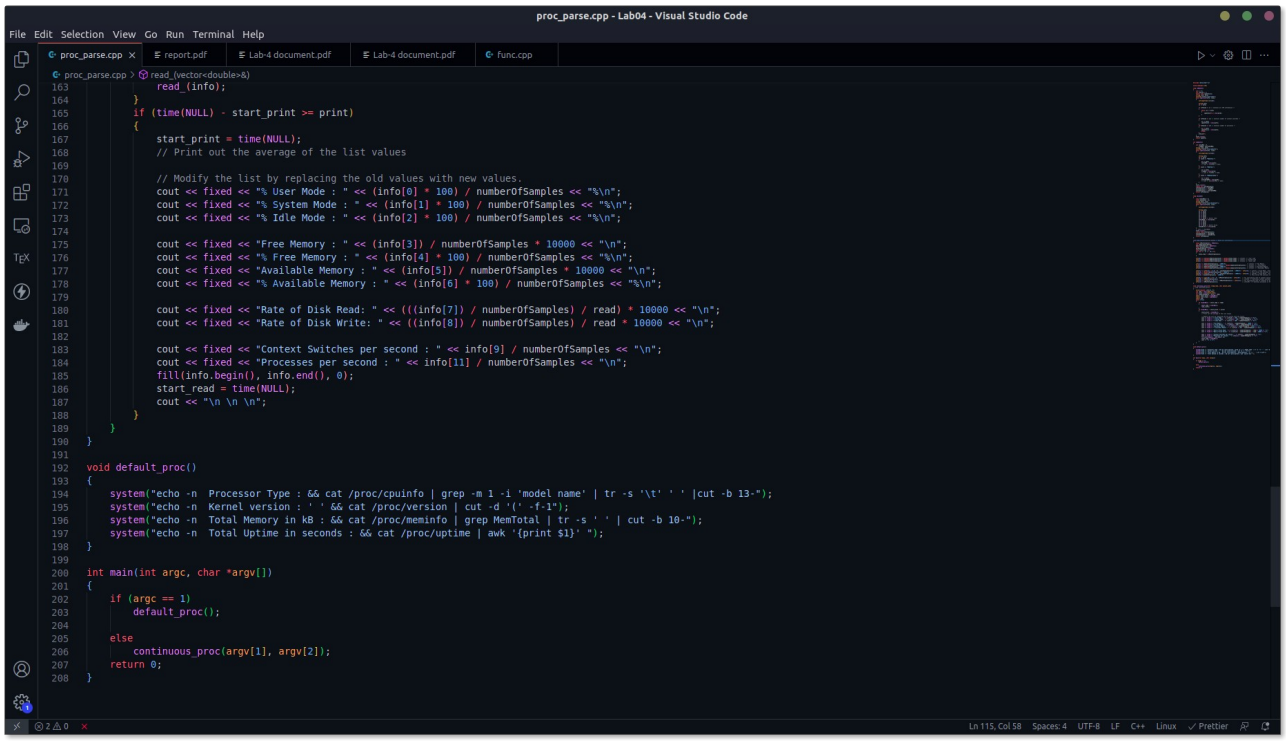





proc_parse.cpp - Lab04 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
proc_parse.cpp
report.pdf
Lab-4 document.pdf
Lab-4 document.pdf
func.cpp
proc_parse.cpp > CPUinfo()
long *diskIO()
{
long totalRead = 0;
long totalWrite = 0;
string word, line;
ifstream finfo("/proc/diskstats");
while (getline(finfo, line))
{
istringstream ss(line);
string word;
ss >> word;
ss >> word;
ss >> word;
ss >> word;
ss >> word;
ss >> word; // Sector read
totalRead += stoi(word);
ss >> word;
ss >> word;
ss >> word;
ss >> word; // Sector Write
totalWrite += stoi(word);
}
// cout << totalRead;
finfo.close();
static long information[2];
information[0] = totalRead;
information[1] = totalWrite;
return information;
}
void read_(vector<double> &info)
{
long *CPUinformation = CPUInfo();
int *memoryInformation;
memoryInformation = memInfo();
long *diskInformation;
diskInformation = diskIO();
long total_time = 0;
for (int i = 0; i < 10; i++)
{
total_time += CPUinformation[i];
}
info[0] += ((double)CPUinformation[0] / (double)total_time); // info[0] -> % user time
info[1] += ((double)CPUinformation[2] / (double)total_time); // info[1] -> %system time
Ln 9, Col 29
Spaces: 4
UTF-8
LF
C++
Linux
Prettier


proc_parse.cpp - Lab04 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
proc_parse.cpp
report.pdf
Lab-4 document.pdf
Lab-4 document.pdf
func.cpp
proc_parse.cpp > read_(vector<double>&)
}
void read_(vector<double> &info) // Reads all information
{
long *CPUinformation = CPUInfo();
int *memoryInformation;
memoryInformation = memInfo();
long *diskInformation;
diskInformation = diskIO();
long total_time = 0;
for (int i = 0; i < 10; i++)
{
total_time += CPUinformation[i];
}
info[0] += ((double)CPUinformation[0] / (double)total_time); // info[0] -> % user time
info[1] += ((double)CPUinformation[2] / (double)total_time); // info[1] -> %system time
info[2] += ((double)CPUinformation[3] / (double)total_time); // info[2] -> % idle time
info[3] += (memoryInformation[1] / 10000.0); // info[3] -> Free Memory
info[4] += ((double)memoryInformation[1] / (double)memoryInformation[0]); // info[4] -> % Free Memory
info[5] += (memoryInformation[2] / 10000.0); // info[5] -> Available Memory
info[6] += ((double)memoryInformation[2] / (double)memoryInformation[0]); // info[6] -> % Available Memory
info[7] += (info[13] == 0.0) ? 0 : (diskInformation[0] / 10000.0) - info[13]; // info[7] -> disk Read , Also normalizing for overflow
info[13] = (double)diskInformation[0] / 10000.0; // This stores the previous value of diskRead. Helps in getting rate of disk read
info[8] += (info[14] == 0.0) ? 0 : (diskInformation[1] / 10000.0) - info[14]; // info[8] -> disk Write , Also normalizing for overflow
info[14] = diskInformation[1] / 10000.0; // This stores the previous value of diskWrite. Helps in getting rate of disk write
info[9] += (info[10] == 0) ? 0 : (CPUinformation[10] - info[10]); // for calculating rate of context switches
info[10] = (CPUinformation[10]); // info[9] -> # previous context switches since boot
info[11] += (info[12] == 0) ? 0 : (CPUinformation[11] - info[12]); // for calculating rate of creation of processes
info[12] = (CPUinformation[11]); // info[10] -> # previous processes in the system
}
void continuous_proc(char *read_rate, char *print_rate)
// void continuous_proc()
{
vector<double> info(15, 0);
int read = stoi(read_rate);
int print = stoi(print_rate);
int numberOfSamples = print / read;
time_t start_print = time(NULL);
time_t start_read = time(NULL);
time_t end;
while (true)
{
if (time(NULL) - start_read >= read)
Ln 115, Col 58
Spaces: 4
UTF-8
LF
C++
Linux
Prettier


proc_parse.cpp - Lab04 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
proc_parse.cpp
report.pdf
Lab-4 document.pdf
Lab-4 document.pdf
func.cpp
proc_parse.cpp > read_(vector<double>&)
info[12] = (CPUInformation[11]);
// info[10] -> # previous processes in the system
}
void continuous_proc(char *read_rate, char *print_rate)
// void continuous_proc()
{
vector<double> info(15, 0);
int read = stoi(read_rate);
int print = stoi(print_rate);
int numberOfSamples = print / read;
time_t start_print = time(NULL);
time_t start_read = time(NULL);
time_t end;
while (true)
{
if (time(NULL) - start_read >= read)
{
start_read = time(NULL);
read_(info);
}
if (time(NULL) - start_print >= print)
{
start_print = time(NULL);
// Print out the average of the list values
// Modify the list by replacing the old values with new values.
cout << fixed << "% User Mode : " << (info[0] * 100) / numberOfSamples << "%\n";
cout << fixed << "% System Mode : " << (info[1] * 100) / numberOfSamples << "%\n";
cout << fixed << "% Idle Mode : " << (info[2] * 100) / numberOfSamples << "%\n";
cout << fixed << "Free Memory : " << (info[3]) / numberOfSamples * 10000 << "\n";
cout << fixed << "% Free Memory : " << (info[4] * 100) / numberOfSamples << "%\n";
cout << fixed << "Available Memory : " << (info[5]) / numberOfSamples * 10000 << "\n";
cout << fixed << "% Available Memory : " << (info[6] * 100) / numberOfSamples << "%\n";
cout << fixed << "Rate of Disk Read: " << (((info[7]) / numberOfSamples) / read) * 10000 << "\n";
cout << fixed << "Rate of Disk Write: " << ((info[8]) / numberOfSamples) / read * 10000 << "\n";
cout << fixed << "Context Switches per second : " << info[9] / numberOfSamples << "\n";
cout << fixed << "Processes per second : " << info[11] / numberOfSamples << "\n";
fill(info.begin(), info.end(), 0);
start_read = time(NULL);
cout << "\n \n \n";
}
}
}
void default_proc()
{
Ln 115, Col 58 Spaces: 4 UTF-8 LF C++ Linux Prettier


proc_parse.cpp - Lab04 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
proc_parse.cpp
report.pdf
Lab-4 document.pdf
Lab-4 document.pdf
func.cpp
proc_parse.cpp > read_(vector<double>&)
read_(info);
}
if (time(NULL) - start_print >= print)
{
start_print = time(NULL);
// Print out the average of the list values
// Modify the list by replacing the old values with new values.
cout << fixed << "% User Mode : " << (info[0] * 100) / numberOfSamples << "%\n";
cout << fixed << "% System Mode : " << (info[1] * 100) / numberOfSamples << "%\n";
cout << fixed << "% Idle Mode : " << (info[2] * 100) / numberOfSamples << "%\n";
cout << fixed << "Free Memory : " << (info[3]) / numberOfSamples * 10000 << "\n";
cout << fixed << "% Free Memory : " << (info[4] * 100) / numberOfSamples << "%\n";
cout << fixed << "Available Memory : " << (info[5]) / numberOfSamples * 10000 << "\n";
cout << fixed << "% Available Memory : " << (info[6] * 100) / numberOfSamples << "%\n";
cout << fixed << "Rate of Disk Read: " << (((info[7]) / numberOfSamples) / read) * 10000 << "\n";
cout << fixed << "Rate of Disk Write: " << ((info[8]) / numberOfSamples) / read * 10000 << "\n";
cout << fixed << "Context Switches per second : " << info[9] / numberOfSamples << "\n";
cout << fixed << "Processes per second : " << info[11] / numberOfSamples << "\n";
fill(info.begin(), info.end(), 0);
start_read = time(NULL);
cout << "\n \n \n";
}
}
}
void default_proc()
{
system("echo -n Processor Type : && cat /proc/cpuinfo | grep -m 1 -i 'model name' | tr -s '\t' ' ' |cut -b 13-");
system("echo -n Kernel version : ' ' && cat /proc/version | cut -d '(' -f-1");
system("echo -n Total Memory in kB : && cat /proc/meminfo | grep MemTotal | tr -s ' ' | cut -b 10-");
system("echo -n Total Uptime in seconds : && cat /proc/uptime | awk '{print $1}' ");
}
int main(int argc, char *argv[])
{
if (argc == 1)
default_proc();
else
continuous_proc(argv[1], argv[2]);
return 0;
}
Ln 115, Col 58 Spaces: 4 UTF-8 LF C++ Linux Prettier