Linux System Monitor

Tamara Alhajj: 100948027
Mohamad Yassine: 100966528
April 13th 2018

Table Of Contents

1 Introduction

1.1 Context

To understand problem we are addressing, we must first discuss the role and differences regarding sysfs and procfs, both of which are discussed in the kernel documentation for file systems. The former of the two, sysfs, is a pseudo file system provided by the Linux kernel elaborated on in the kernel documentation [1]. It is primarily used to access information pertaining to the hardware and kernel modules. Sysfs exports information from the kernel's device model to user space through virtual files. Contrastingly, proc file system (procfs) is an older and more chaotic file system compared to sysfs [2]. Procfs also provides a method of communication between kernel space and user space, by acting as an interface to internal data structures in the kernel. Furthermore, procfs presents system information in a hierarchical file structure, which provides some convenience for dynamically accessing process info and the capability to change certain kernel parameters at runtime.

The current system monitor (included in Ubuntu 16.04) provides various information such as current running processes, as well as CPU and memory usage. However, the current system monitor does not display any hardware information; the user is not provided with information on the current running frequencies for the CPU, average temperatures of all cores, nor battery usage information. In addition, the provided monitor does not have logging of the memory usage per each process. Although the system monitor displays running processes, it does not detect high memory usage for each process. In order for a user to get such information they would have to go through the sysfs directory, or run valgrind on a suspected program; neither of which are simple tasks for the average user.

1.2 Problem Statement

The provided tools in Linux (specifically Ubuntu 16.04) do not provide an intuitive and effective methods of viewing information about the system. Users uneducated in the Linux pseudo filesystems such as sysfs or procfs will find difficulty to understand and navigate the filesystems. The many managers provided in the most popular distributions, including Ubuntu 16.04, do not provide the user with hardware information and they do not include memory usage detection methods.

The inspiration of developing SysMon is to provide the end user with a simple and intuitive experience in altering and previewing system information. In addition, Sysmon aims to provide the end user with a simple memory detection monitoring tool. The motivation behind this, is to provide the user with pertinent information in one place. This eliminates the need for the user to directly interact with the kernel, and by extension, the need spend significant time in understanding how to navigate the various directories in the sysfs or procfs. In addition, it excludes the need for the user to install multiple programs or packages to monitor their system. The goal regarding hardware information is to display the information of system components such as CPU information involving frequency, temperature and overall usage are displayed to the user in a comprehensive GUI. The aim is to help and inform the user of specific processes that are rapidly increasing in memory usage, which may be a result of a memory leak.

1.3 Result

We began with a simple CLI to for our early designs, to ensure we could monitor and modify the intendid system components. This step aided us in achieving an all encompassing GUI for our final design. Overall, we have achieved our goals regarding system monitoring and modification. We managed to provide the user with a friendly to use graphical interface (see evaluation on user testing in section 4) while maintaining all of our promised goals and more. The product provides the user with the ability to monitor processes memory usage, in addition to, monitoring hardware information. The user is provided with the option to change the power consumption profile to a ‘powersaver mode’ or a ‘performance mode’. On a another tab, the user has the ability to start a monitor which will probe and log the memory usage of each process individually. However, a couple of the system modification had to be done using specific packages, due to permission issues regarding the modification of specific files.

1.4 Outline

A summary of the contents of each of the remaining sections.

Section 2 conveys the necessary background information on the files used to store running process and hardware information, as well as pre-existing system components that lead to the conception of SysMon. In regards to background information, we then highlight the various disadvantages to the current systems in place. We then move on to section 3 to describe the work we have been able to achieve. This includes a review of our software interface with representative use cases. Following this is a description of the architecture and logic of your software. All claims about SysMon’s success are supported with evidence, shown in section 4.

2 Background Information

The Procfs and Sysfs collect information from the kernel subsystems and organize them in an ordered manner. Each has a specific use. Procfs handles process related information such as memory maps, process identifiers, states including other process information. On the other hand, Sysfs handles system related information such as screen brightness, temperatures, frequencies, fan speeds and other various hardware related items.

Moreover, Sysfs allows the user to alter hardware parameters such as fan speeds (i.e pwm), CPU max frequency and other hardware settings. For example, to change the screen brightness in a Sysfs, you can navigate (using the terminal) to: '/sys/class/backlight/intel_backlight/brightness' and echo a value within the max brightness (found in /sys/class/backlight/max_brightness/) and zero. A simple code snippet below can demonstrate how this could be done :

```
#!/bin/bash

# Get battery percentage

BATTERY_DIRECTORIES=()
while IFS= read -r line; do
    BATTERY_DIRECTORIES+=( "$line" )
done < <( upower -e |grep battery )

for i in "${BATTERY_DIRECTORIES[@]}"
do
           percentage=$(upower -i ${i} | grep "percentage" |awk
{'print $2'} | sed 's/\%//g')
        sum=`expr $sum + $percentage`
done

avg=`echo "$sum / ${#BATTERY_DIRECTORIES[@]}" | bc -l`
printf '%0.2f' "$avg"
```

These tools and features give users a deep insight and visibility to their hardware and devices. However, these tools require an experience user who knows and understands these filesystems to first be able to access them and to modify them safely without damaging their system components. A user that is inexperienced with these file systems will have difficulty accessing them in the first place, and even if they do find them, they have a higher risk of damaging their systems.

Linux also lacks easy to use hardware monitoring tools. Users would typically have to install multiple tools and or traverse the system filesystem (sysfs). Both methods are not clean and not very straight forward. Users will have to jump through multiple hoops to get to one simple piece of information. This is one of the things we wanted to help solve and eliminate. We want to make sure we provide the user with a simple interface to allow them to monitor their system information in a simple manner and concise manner.

Furthermore, another lacking feature we found in Linux was a program or feature that would help and detect high memory usages (potentially leaks). There are many cases where there is a background process that is running and continuously increasing in memory usage. This could lead to global system slow down and possibly an unresponsive system. We wanted to help eliminate this problem and provide this functionality while maintaining the user friendly graphical interface. This is not directly possible method in Linux to achieve this effect, which is our main inspiration of implementing such a feature.

3 Result

The final product result of System Monitor, Sysmon, was to create an easy to use program that simplifies adjusting and previewing system information. In addition, to help in providing additional features that will improve the user experience. In order to clearly and effectively demonstrate the level of success of Sysmon. It is important to show the steps and process the user would have needed to take the traditional way without the use of Sysmon. Consider the following, the user wants to retrieve or alter information regarding his hardware. Things like processor temperatures, setting a governor on his CPU in order to save power, or getting process information to analyse if a process is continuously using more and more memory.

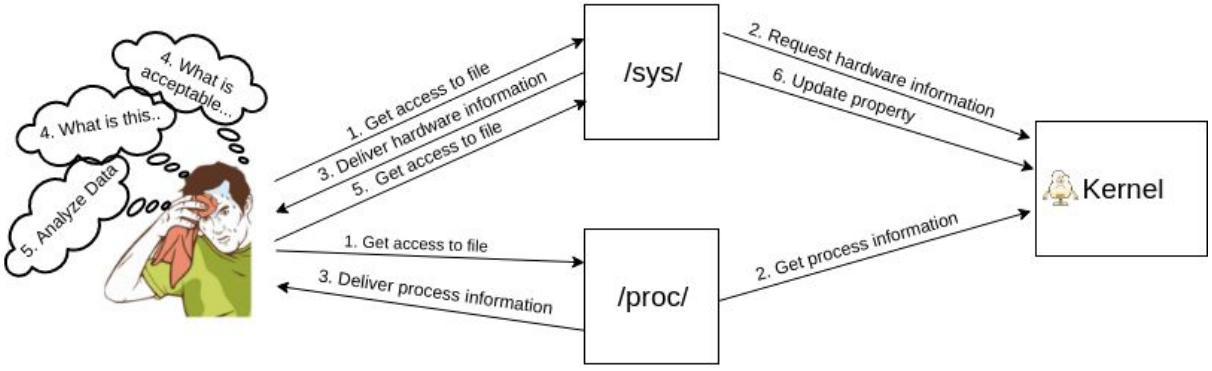

Figure 1.1

As seen in Figure 1.1, the user has to go through many hoops to get to the simple information they need. In step one, the user needs to, using terminal, access

files in /sys/ (sysfs) and files in /proc/ (procfs) directories. In the case of altering hardware information such as the scaling governor for the CPU, in order to prolong the battery life. The user needs to access '/sys/devices/system/cpu/cpu0/cpufreq/scaling_setspeed'. Once accessed, sysfs will request from the kernel information regarding the specific file the user had requested from 'hwmon'. This will immediately return to the user with the information requested (step 3). Once the user has been delivered the information, they would need to alter the contents of the file to change the speed[1] of the processor. The user then submits saves this information to file. Sysfs will then carry these changes to the kernel and update the properties, in this case the CPU clock speed.

On the other hand, if the user wanted to get processes information they would have to traverse /proc/ (procfs). All running or sleeping processes are in directories. These directories contain all contents of the process. Things like process command, current status, memory map, attributes and other various information that is specific for each process. In order for the user to get the memory usage of each process they are required to read through the '/proc/PID/statm' to find the actual memory usage of the process. However, this is not very intuitive. Especially memory usage on Linux is weird. Memory is split between physical memory and virtual memory. Both are *fair* memory for each process. So without prior monitoring the user will have a very difficult time understanding how much memory the process is actually using.

Sysmon provides all the previous functionality, however, at a much simple process. The user only need to interact directly once with the program and the program will continuously manage the information. The following example is the exact same as the previous however using Sysmon.

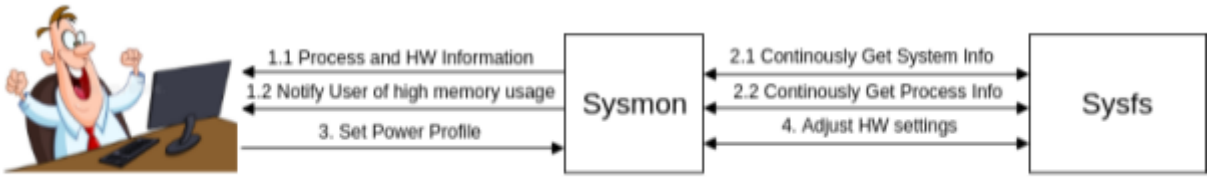

Figure 1.2

Comparing Figure 1.1 and Figure 1.2, the difference is very apparent. The process of interacting with the system is a lot simpler. The user needs to interact with the system once, from the previous example; three times) to set the power profile for the power saving to prolong the battery life. In step 1.1 and 1.2 the user is presented with realtime information about their system. Steps 2.1 and 2.2 run in the background. Sysmon talks with sysfs directly and continuously. The program takes care of representing the information in a visually pleasing way without needing to traverse the filesystem. Furthermore, the user is able to choose based on the

information presented to whether they want power-saving mode or performance. With a simple click of a button the user is able to set the scaling governor of their processor. Of course this functionality can be extended to allow for a slider to adjust the frequency of the processor in terms of percentage. However, that would have mean the user would need to understand what the frequency or clock speed meant. As a result, simple ‘Powersave’ and ‘Performance’ buttons were used.

A significant component of Sysmon is providing the user with a easy to use UI, and that is provided with the GUI. Upon starting the program, the user is presented with the main page showing them relevant information about their system. As seen in the *Figure 2.1*, the program demonstrates relevant information the average temperature, clock speed (frequency), and battery status. In addition to, overall usage of both processor and memory. We felt like most users will not understand or perhaps grasp the bound of each value. As a result, using a visual tool such as a coloured circular progress bar helps and provides the the bounds for each component. For example, the temperature bar will shift more towards the colour red indicating that the temperatures are now “hot”.

In addition, clicking the “Performance Mode” will increase the processor clock speed (frequency) and will set the screen brightness to max, as seen in Figure 2.1. This is to ensure that the system is running at its peak performance. On the other hand, clicking on “Powersave Mode” will reduce the brightness to a third of the max brightness and will reduce the clock speed of the processor, see Figure 2.5. It does so by changing the scaling governor. The new governor is a more conservative increase the clock speed as often. This will reduce the power consumption of the processor, and conveniently aid in elongating the battery life of the machine.

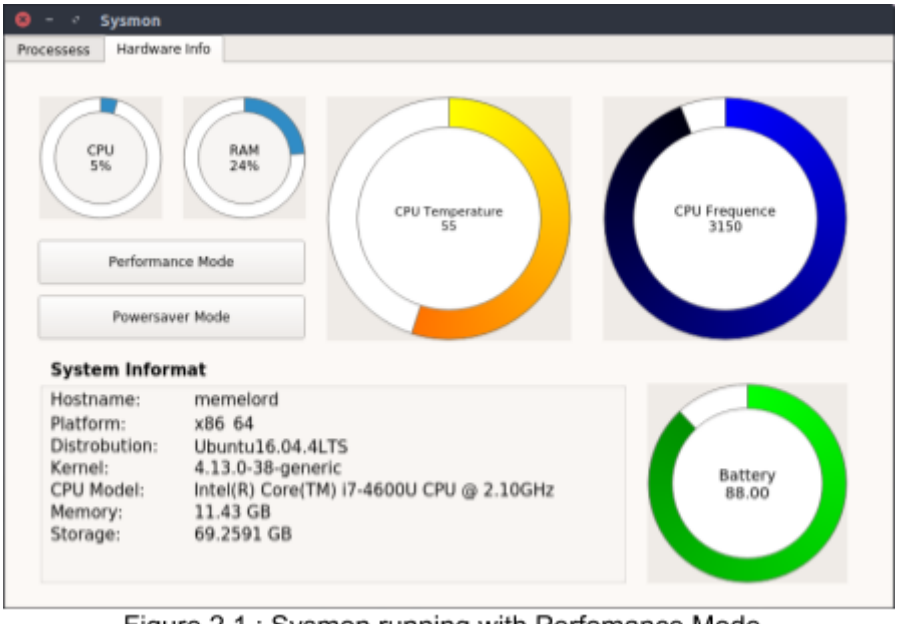

Figure 2.1 : Sysmon running with Perfomance Mode

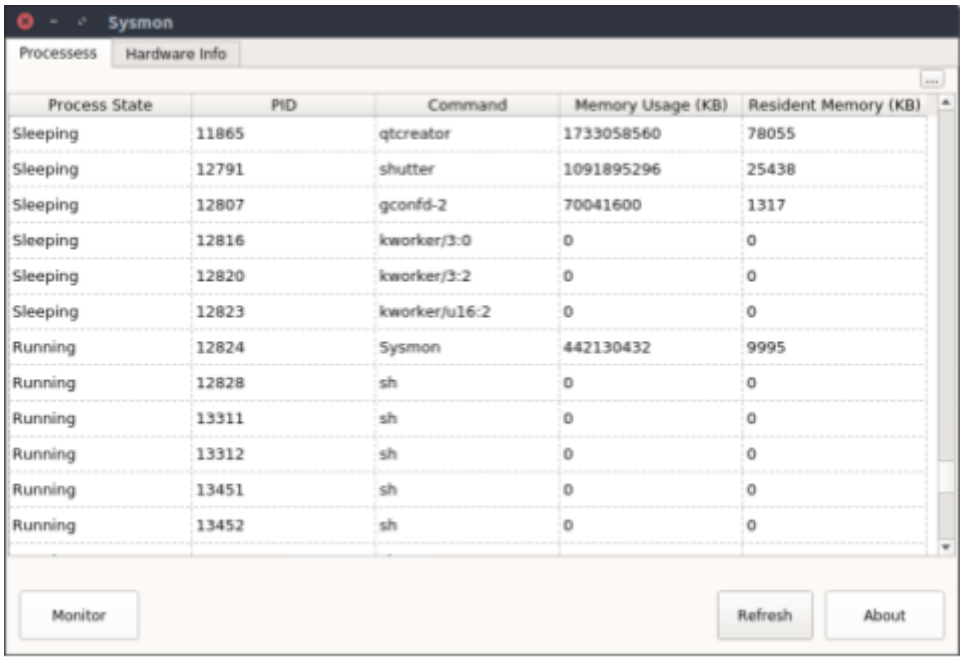

Figure 2-2 : Sysmon showing current running processess

Sysmon is also able to manage processes. Clicking the “Processes” tab will present the user with a list of the currently running processes, seen in Figure 2.2. In addition, the user is also presented with a “Refresh” button that will refresh the processes list and their memory usage. The memory usage includes virtual memory and residual memory. Residual memory is the program memory that is located on physical RAM, whereas virtual memory is the total memory the program is using. This includes memory located in RAM and in swap.

Furthermore, the “Monitor” button will initiate a background process that will continuously check for high memory usage. The program logs the memory usage for each process that is currently running and determine based on a mathematical model if there is a potential high memory usage. Once a memory leak is detected the system will send a system wide notification, as seen in Figure 2.4. The program will not directly kill the process due to uncertainty of the contents of the process.

Figure 2-4 : System Notification of possible memory leak

Figure 2-3 : Graph memory representation

Moreover, if the user double clicks a process they will be presented with a graph about the memory usage over a period of time. The usage is based on the total memory usage of the process and the time at which it was probed.

[1]: Most processors built after 2010 have the ability which allows the user to downclock them without needing to unlock the processor.

4 Evaluation and Quality Assurance

We conducted a small survey after by a testing our software, on a total of 9 people; these people were of various experience using linux systems. The data from this survey can be found in the appendix of this report. The first graphs depicted below show the feedback after using SysMon. The x-axis of all the histograms represent data on a 5 point scale: 1 being poor at all and 5 being great. The y-axis of said histograms are frequency of response (ei. count).

Sysmon User Testing

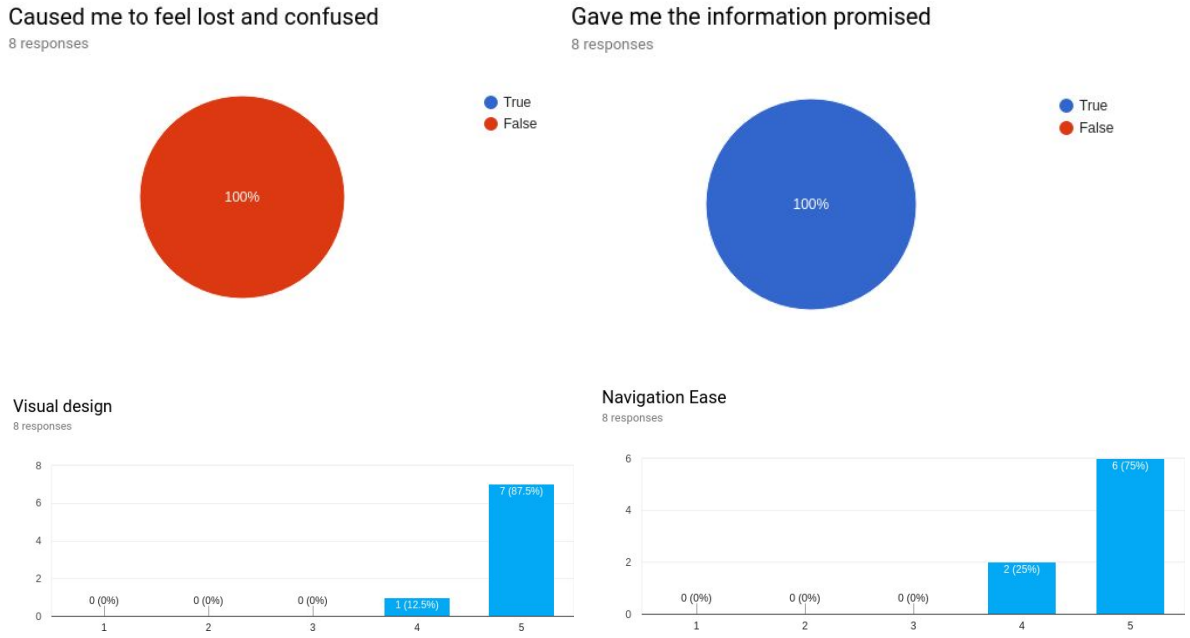

Figure 4.1 SysMon user testing results graphed

It is clear from the graphs that users generally found our software easy to use. We insured that they completed to tasks: interaction with the process tab, and interaction with the hardware tab. Firstly, when we ran a program with a purposeful memory leak, the users were able to navigate to that process in the chart, click to see the usage increase visually, and lastly were shown the warning message for the PID of the memory leak program. Overall this task was completed successfully, but as many users verbalized to us, would have been done faster with a search feature for the PID of the 'leaky program'.

Next, the user was tasked to find CPU temperature, frequency, and memory usage. This was done with speed and ease as all the user had to do was navigate to the hardware tab. We then asked them to decide if they wanted to save battery and

to alter the system components accordingly. This again was done with ease since there is two clear buttons to accommodate. For “powersaving mode”, dims the screen and lowers allowed CPU frequency and “performance mode” does the exact opposite. The users, as shown by the visual design histogram, took well to the visual depiction of this data.

Sysfs User Testing

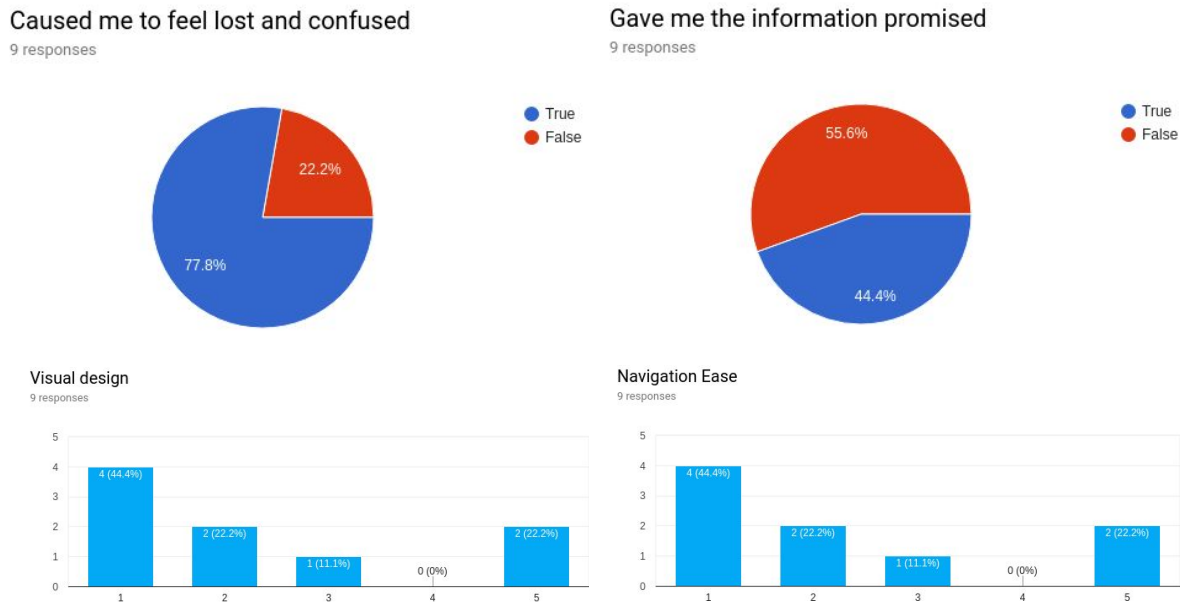

Figure 4.2 Sysfs user testing results graphed

Given the data above it was pretty apparent that most users have difficulty traversing and understanding the system filesystem (sysfs). We asked the users to try and find the information about their system such as temperature and processor clock speed. They were given a terminal and asked to go to sysfs and retrieved the tasked information. However, most users failed to complete the task, while those who did took a very long time. Comparatively, they had an easier time navigating SysMon; this is shown by the graph depicting low scores for navigation ease when using sysfs. Moreover, the visual design scores of SysMon are higher most likely due to sysfs being navigated through the command line. Significantly, some users did find sysfs easy to use however the data shows these users are more experienced with linux; this can be verified in the data linked in the appendix.

System Monitor

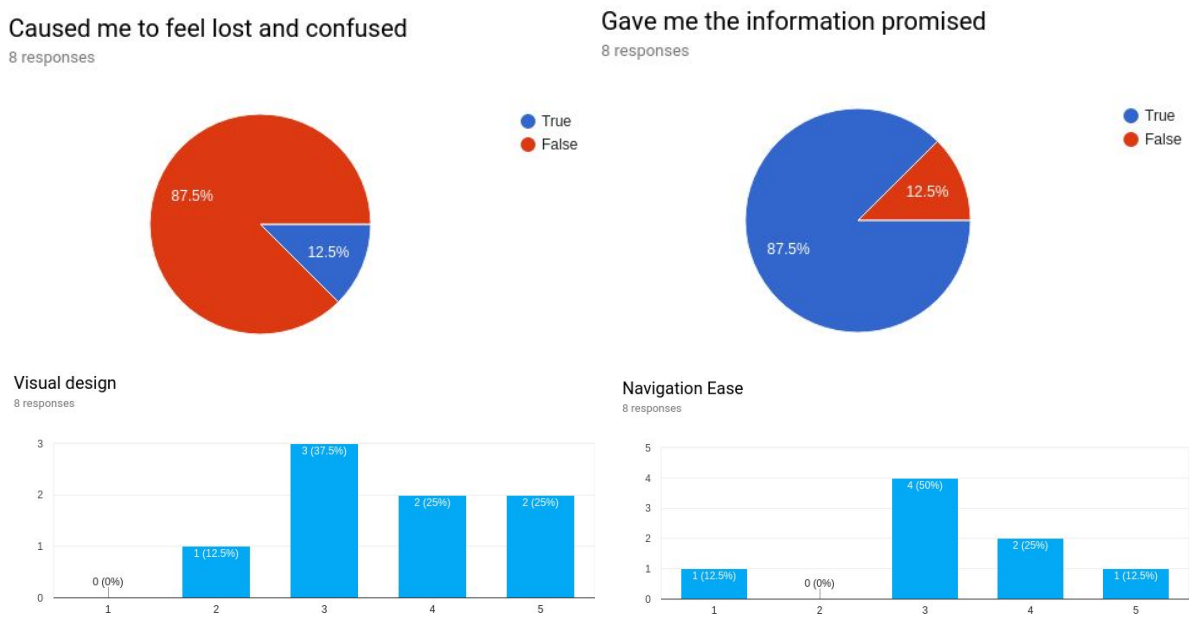

Figure 4.3 System Monitor user testing results graphed

When using the system monitor, the primary task was analyzing the list of running processes to compare the usage to our process tab. The hardware data depicted was still looked at but not focused on since it only includes information on CPU history, memory/swap history and network history. This does not include any graphs on CPU temperature, frequency, nor usage, and also no information on the battery. Thus those tabs on the differing systems are not comparable as they do not share the same goal. To go back to the topic of processes, users somewhat liked the existing system monitor but as the graphed data shows, not as much as the SysMon. Although, users overall were not lost or confused but the linux system monitor, and were able to get the information of the task (promised information), they had more difficulty navigating through the linux system monitor GUI when compared to the stats shown for navigating our SysMon. Overall, users were not disappointed by the system monitor, however the graphs show more positive feedback when using our software to analyze memory usage.

Valgrind Quality Assurance of SysMon

The figure below, 4.4, plainly reveals the there are no memory leaks when running our software for a prolonged period of time. Due to the nature of the program, this was a significant task we wanted to achieve. As a test the program was run and logged memory usage of processes for twenty minutes and showed not potential memory leaks.

Figure 4.4: Valgrind 20 Mins Run - Checking For Memory Leaks

```
$ time valgrind ./Sysmon
==28827== Memcheck, a memory error detector
==28827== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==28827== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==28827== Command: ./Sysmon
==28827==
28827
28852
28852
Constant monitoring has started
==28827==
==28827== HEAP SUMMARY:
==28827==     in use at exit: 0 bytes in 0 blocks
==28827==   total heap usage: 0 allocs, 0 frees, 0 bytes allocated
==28827==
==28827== All heap blocks were freed -- no leaks are possible
==28827==
==28827== For counts of detected and suppressed errors, rerun with: -v
==28827== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)

real    22m47.424s
user    0m26.619s
sys     0m16.462s
```

5 Conclusion

5.1 Summary

Fortunately, we were able to solve the problem we set out to achieve: to provide the end user with a simple and intuitive experience in altering and previewing system and hardware information. Our achievements regarding the GUI display of our hardware and process information are supported by our usability testing linked in the appendix. This sample of students includes people with differing experience in linux, which is logged as user info. Our only flaw is in the robustness of our memory usage monitor. To elaborate, the warning will go in any case of rapidly increasing memory; thus, we can not, and do not, claim that memory leaks will be found. Memory leak analysis could have been implemented in the back end using valgrind, which would add more robust functionality.

5.2 Relevance

Regarding topics from class, our projects makes use of of the procfs, sysfs, and processes topics from class. Respectively, these were used to access and modify data on a user's system, and processes. Our Qt application was used to signal all calls to the terminal as super-user which allows us access the file systems without permission issues. Overall relevance of the content with respect to the

course topic is clear. The topic of this course is operating systems; given we are interacting with the kernel, our project aligns directly with the topic of the course.

5.3 Future Work

To expand on our current work, we could simply add more system information to be logged or altered, found in sysfs. An idea we have considered is fan control to adjust the CPU temperature, which would complement what we currently display for the averaged CPU temperature. Regarding memory usage, it would be highly advantageous to have an option to sort the listing the processes. Moreover, a search by PID would make navigation much easier.

Contributions of Team Members

Tamara Alhajj contributed to the hardware monitor portion of the assignment. This includes retrieving information about core temperatures and averaging them, cpu frequency displayed in MHz, the battery information, and the altering of these system components for powersaver/performance mode. Moreover, Tamara has contributed to the UI design and construction. Lastly the completion of this report was also worked on by Tamara.

Mohamad Yassine contributed to the memory process information and monitoring tab, including the graphs once a processes is selected. He also contributed with additionally functionality to the hardware information tab by adding CPU/RAM usage in percentage, as well as the listed general system information at the bottom. He also constructed the GUI and contributed to this report.

References

[1] https://www.kernel.org/doc/Documentation/filesystems/sysfs.txt
[2] https://www.kernel.org/doc/Documentation/filesystems/procfs.txt
[3] http://man7.org/linux/man-pages/man5/proc.5.html
[4] https://techtalk.interse...memory-part-2-understanding-process-memory/

Appendix

User Testing data organized by Task, for ease of comparison:
(Horizontal line represents loop to the same user as previous)
https://drive.google.com/open?id=1kw62NPMyY8YCHG5R0WzK3hiWuE_0mZPFvvTbXB2lC30
Our survey as used for testing:
https://goo.gl/forms/ptQYxIjxmuzqoifG2