

CS 252: Systems Programming

Fall 2019

Lab 6: Task Manager

Prof. Turkstra

Goals

The goal of this project is to learn about the `/proc` filesystem and work in a group to build a graphical user interface that displays pertinent system information similar to the Windows Task Manager.

The `/proc` Filesystem

The `/proc` filesystem is a pseudo-filesystem that provides information about processes and kernel data structures in a hierarchical file-like structure. This filesystem is automatically created by the system during bootup and relevant kernel variables are updated as needed. As such, `/proc` acts as a simple interface for retrieving system data from user space. To read more about the `/proc` filesystem, visit [this link](#). It will likely come in handy for this project.

Deadline

The deadline for the final submission is **Monday, December 2nd 11:58pm**. Grading will be done in-person during lab sections. You must have made a submission prior to the deadline to avoid the standard late penalties.

1 Requirements

For this project, your group can choose a language or framework of your choice to implement the GUI. If you are having difficulty choosing, there is a list at the end of this handout specifying common languages/frameworks and their respective documentation.

You should create a system monitor that is somewhat similar to e.g. the Windows Task Manager or the Linux System Monitor (mate-system-monitor). Screenshots of the latter are included below.

At a minimum, your system monitor should implement the following features, preferably as tabs or individual windows:

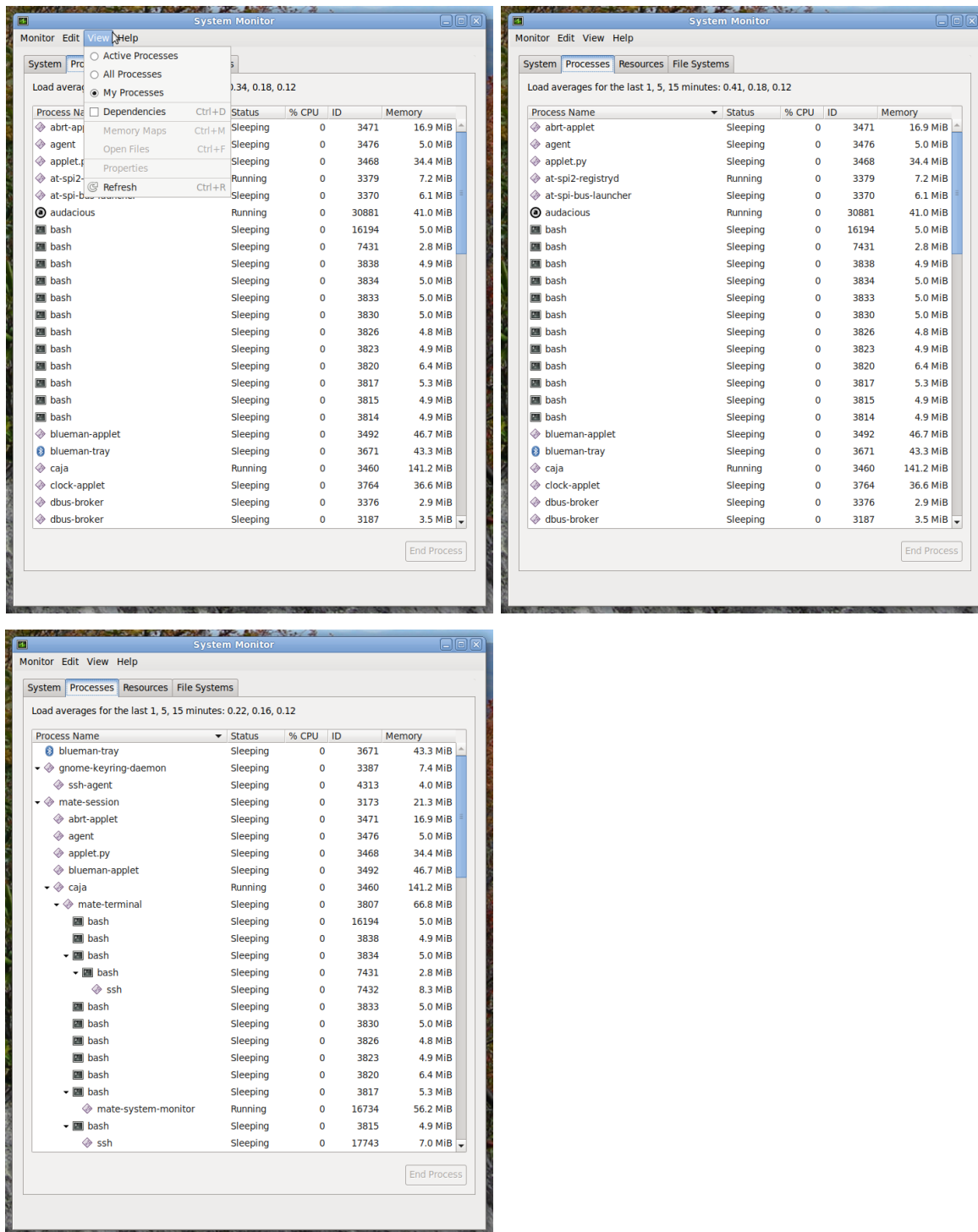
1.1 Basic system information

Should include, at a minimum, the OS release version, kernel version, amount of memory, processor version, and disk storage.



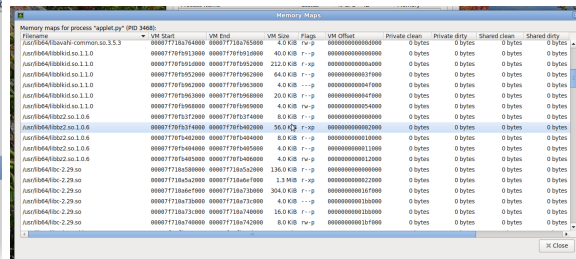
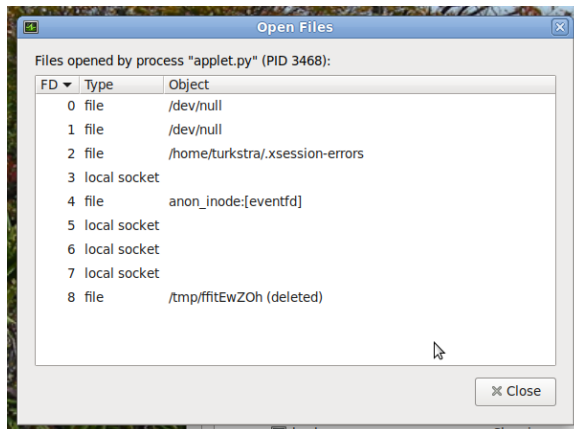
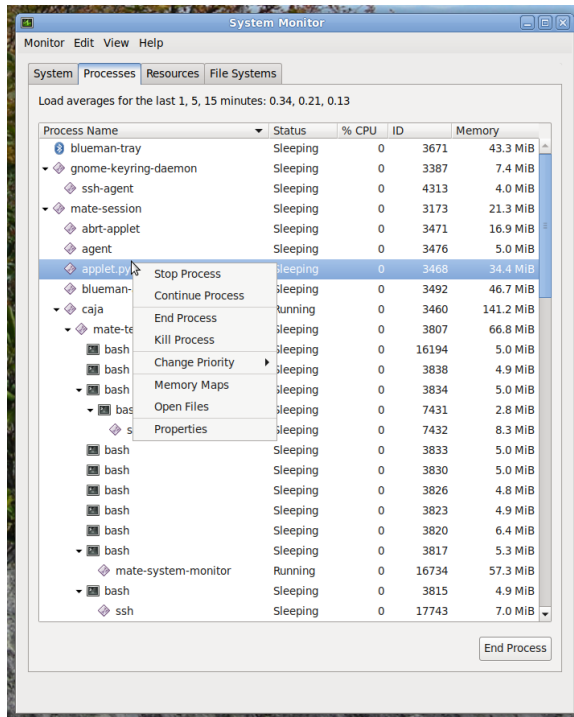
1.2 Process Information

A table with options to: display a list of **all** running processes, display processes only owned by the current user, display the list in tree format. The list should update at a fixed interval or a refresh button should be provided.



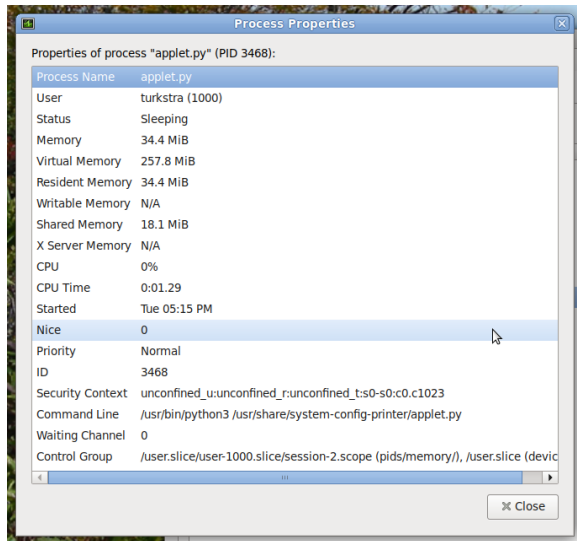
1.2.1 Process-specific Actions

For each process the ability to: stop, continue, kill, list memory maps, list open files



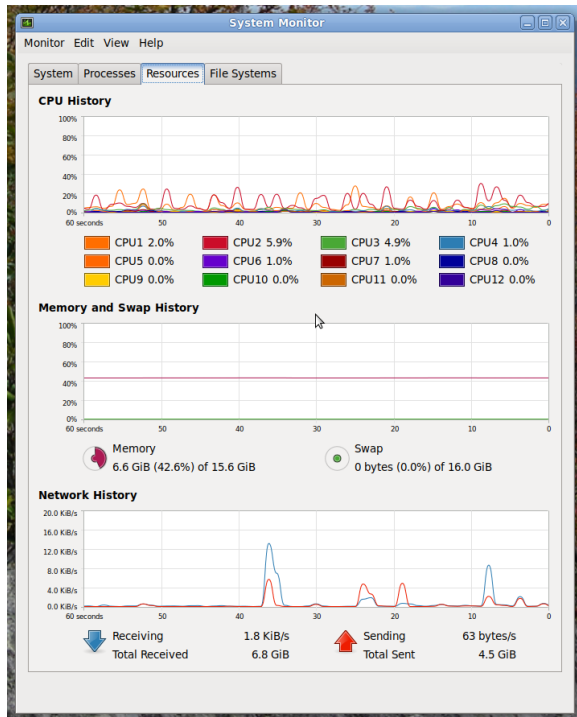
1.2.2 Detailed View

For an individual process, a window that shows: process name, user, state (running, stopped, etc), memory, virtual memory, resident memory, shared memory, CPU time, and date/time started.



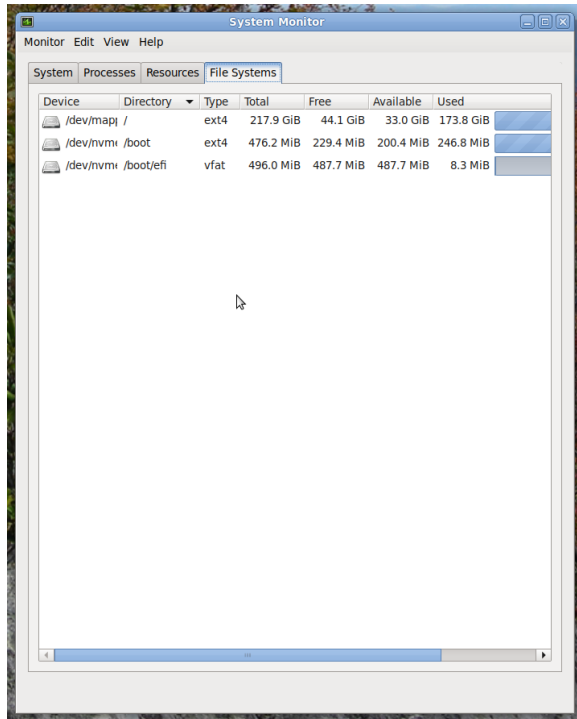
1.3 Graphs

Usage graphs that track: CPU usage, memory and swap usage, and network usage



1.4 File System

Usage for each mount point (e.g. `df`)



The screenshot shows the 'System Monitor' window with the 'File Systems' tab selected. It displays a table of disk usage for three mount points. The table has columns for Device, Directory, Type, Total, Free, Available, and Used. The data is as follows:

Device	Directory	Type	Total	Free	Available	Used
/dev/mapper /	/	ext4	217.9 GiB	44.1 GiB	33.0 GiB	173.8 GiB
/dev/nvme /boot	/boot	ext4	476.2 MiB	229.4 MiB	200.4 MiB	246.8 MiB
/dev/nvme /boot/efi	/boot/efi	vfat	496.0 MiB	487.7 MiB	487.7 MiB	8.3 MiB

2 Grading

2.1 Appearance and Correctness

Five points will be determined by a clean valgrind result.

Five points will be determined by robustness testing (use of multiple features many times)

Another five points will be determined by the overall appearance and intuitiveness of your solution.

2.2 Code Review

In addition to demonstrating your solution, you will participate in a brief code review where you walk your grader through your solution. We expect you to adhere to a code standard. It doesn't have to be the course code standard. We reserve the right to subtract up to five points from your final grade for failure to enforce a code standard.

2.3 Additional Work

The above represents the minimum requirements for your project. We will award up to 15 points extra credit for additional features based on complexity and completeness.

3 Version Control

You are required to use revision control for this project. We suggest you use git. You can set up a repository on GitHub, or you can email Prof. Turkstra and request a private repository. Your request should include ssh keys for each team member.

As part of the grading process, we will look briefly at your commit history. This will determine 5 points of your grade.

Any code without a git repository or history will not be graded.

4 Turn-in

You should submit your code no later than **Monday, December 2 at 11:58pm**. It must compile and run properly on data. You should include a Makefile to build your code. Graders should simply be able to run "make" and then execute the resulting binary.

You should include a README file with a list of team members and technologies used.

To submit your project, execute from your cs252 directory (assuming your code is in the lab6 directory):

```
$ turnin -c cs252 -p lab6 lab6
```

5 Potential Languages or Frameworks

You are required to use either C or C++ for this project. You can use an framework/libraries/etc that you want, however.

If you pull in code from another project or stackoverflow, you **MUST** cite this in a comment above the code!

Teams should not share or inspect code from other teams.

5.1 C

- GTK (Documentation)

5.2 C++

- [Qt \(Documentation\)](#)
- [FLTK \(Documentation\)](#)