Project Selection Report

I selected the Linux System Monitor project. I chose this project because I often need to use design and simulation tools that are heavy on system resources, so keeping track of system usage is very important to me. I would like to create a monitor with a simple graphic interface to quickly convey this usage information to me.

My system monitor is a terminal-based tool that provides real-time system usage statistics, including CPU, memory, and disk usage in a visual bar format. These bars will use symbols like ▪ and ▫ to represent the percentage of usage. For example, if CPU usage is at 40%, it might display ▪▪▪▪▫▫▫▫▫▫ to indicate that usage visually. This tool will gather data using built-in Unix utilities, then filter the information using commands such as grep and awk. This filtered information is then converted to graphs to be outputted. The graphs should periodically refresh to provide real-time data.

The tools built into Unix for system monitoring are very powerful and informative, but they generally provide too much information at once in a way that is overwhelming at a quick glance. This results in time being spent deciphering the information being given to the user when the main focus should be on the work the user is doing.

My proposed solution solves this by doing this deciphering for the user. By turning the walls of information into small graphs, the user can focus more on their work while still having quick access to usage information at a glance. This solution should require very little computing power and would be easy to run in a terminal in the background.

Project Timetable

Week	Situation	Task	Action	Result
Week 1 (Apr 10-17)	Need basic system information parsing from Linux sources	Collect and parse data for CPU, memory, and disk usage	Read from */proc/stat*, */proc/meminfo*, and use *df* to fetch disk usage; parse values with *awk*/*grep*	Successfully extract raw metrics needed for usage calculations
Week 2 (Apr 17-24)	Need to display values visually in terminal	Build visual ASCII bar function	Create script that translates percentage to a bar	Working bar script; outputs correct visual representation
Week 3 (Apr 24-May 1)	Need real-time refresh and improved UX	Add terminal refresh & layout formatting	Use *clear* to update display; loop with sleep and while loop	Script refreshes every few seconds with a clean, readable layout

Unix Project Report

The biggest hurdle I faced was extracting the usage data in a form that can be converted to the graphical bars. Bash requires that all data be an integer for math to be done, so all of the measurements had to be parsed as such. This was further complicated by the need to use different tools–*top*, */proc/meminfo/*, and *df*–to access the different resource usages, so each access method needed to be bespoke for each resource. A more minor hurdle was creating an efficient loop that could create the graphical bars from the usage data.

The *awk* and *grep* commands proved invaluable for the data access. Each data point could be retrieved using *grep* to grab the line with the data and *awk* to choose the part of the line that had the actual value. For the CPU data, I had to use the *printf* function inside of *awk* to explicitly cast the CPU usage (100-*idle*), which is in the form *XX.X*, to an integer. For memory usage, total and available memory had to be grabbed with *grep* and *awk* as separate variables to calculate used memory. For disk usage, *grep* and *awk* were used again to obtain *Use%* of total system disk space, but *sed* had to be used to remove the *%* symbol to store the value as an integer. To make an efficient loop, rather than create a *count* variable to use in a *while* loop, I found that *for i in (seq 1 $filled)* can be used. This creates a list from 1 to our usage variable, which then can be looped through.

If I were to expand on this project, I could add several quality-of-life features to the display. This might include colors for faster understanding of the information, such as a red tint for high usage. I could also monitor total system usage of individual programs, or on the flipside, usage of individual CPU cores or disk partitions. Finally, I might also log data to a *CSV* file to see and process usage over time.

Final Project Report

For the final version of the system monitor, I was able to implement several of the features discussed in the previous report. First, each line of text was given a different title color. This makes distinguishing the data points easier, as the "[Resource] Usage:" labels now stand out from each other. On the subject of coloring, I also added a color warning system. As long as the usage of each resource is below 50%, they are output in green; however, if the usage of a resource passes 50%, the value and bar for that particular resource will be output in red. This provides an immediate warning to the user that they need to pay attention to that resource's usage. Next, I added functionality to obtain the name of the process consuming the most CPU time. This is output underneath the individual usages. Finally, I added logging to a file, named "Monitor-Log.csv". This includes the current date and time, the usage of each resource, and the process with the highest consumption.