Sravya Kuchipudi
Professor Kaisler, Hasanov
CSCI 6917
08 June 2023

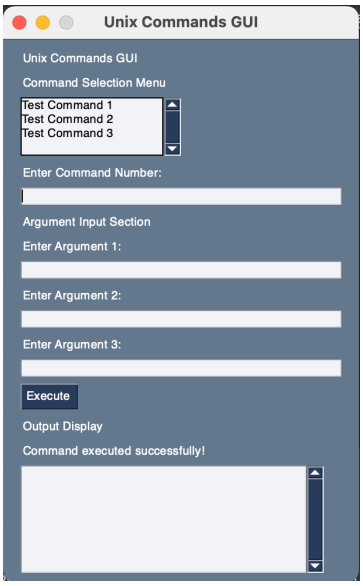
Project Report: Building a GUI for Unix Commands Execution on MacOS

Most of the work done so far for the project has focused on figuring out the best library to work with and then using that library to start building the GUI interface. As I already have proficiency in Python, I decided to research a few different libraries to figure out the best approach. PySimpleGUI ended up being the best choice for the project since it is relatively easy to use which lowers the learning curve. It's a cross-platform library, which means it supports multiple operating systems, including MacOS, which aligns with the project's goal of building a GUI for MacOS, so I can develop a GUI application that works consistently across different platforms without requiring significant modifications. PySimpleGUI includes a range of pre-built widgets, such as buttons, input fields, and dropdown menus, that cover a wide range of user interface needs, which allows for the creation of a visually appealing and functional GUI. Since it is customizable, the existing widgets can be used in several ways to address the needs of this project.

The process of creating the test code involved several steps. First, the PySimpleGUI library is imported, which provides the easy-to-use GUI interface for Python development. Then, the GUI layout is defined using PySimpleGUI's layout syntax. This is described in more detail later on, but the layout is formatted to balance ability with simplicity for novice users who lack comfort with Unix commands.

The next part of the code creates a window using the defined layout, serving as the container for the GUI components. The code enters an event loop, where it waits for user interaction or events to occur. Within the event loop, the code handles specific events such as button clicks or window closures. If the user clicks the "Execute" button, the code retrieves the command number and arguments entered by the user.

To demonstrate the functionality, a placeholder message is currently generated as the output, however, in future implementation, the selected Unix command would be executed, and its output would be captured. Then the output display area in the GUI window shows the captured output, allowing the user to see the results of the executed command.



The layout follows a rather simple format shown in the image above, with clear sections for the command selection menu, argument input, execute button, and output display. The format is clear and intuitive, allowing users to easily identify and understand the various GUI components. Input prompts for the command number and arguments make it clear to users what information is expected, reducing confusion, and ensuring accurate input. The format also includes a designated area for displaying the output, providing users with convenient access to the results of the executed command. Furthermore, the compact design optimizes space usage, resulting in a clean and visually appealing interface without overwhelming the user with excessive clutter. While there may be later changes to either improve functionality or aesthetics, this was the initial choice to cover the different requirements of the input and output.