ELEC 377 Lab 4: Shell Scripting – Description of Solution

Section 003, Group 16 – Thursday Lab

Erhowvosere Otubu, #20293052

Ivan Samardzic, #20296563

Date of Submission: November 14, 2023

Problem Solution

The purpose of Lab 4 is to utilize shell programming to list run processes using the */proc* interface in the Linux file system. The */proc* directory serves as a file-based interface to the kernel, providing variables, counters, and other system information. The script is designed to provide flexibility by allowing users to various flags, such as *-rss*, *-comm*, *-command*, and *-group*, to customize the output. The focus of the script developed for this lab is to generate a comprehensive listing of currently running process numbers, names, and status.

Solution Approach

The following sections will explain in-depth the process of the group's solution.

Phase 1

The script starts by parsing command line arguments using a 'while' loop and the 'shift' command. Various flags, such as *-rss*, *-comm*, *-command*, and *-group*, are recognized, and corresponding variables (*showRSS*, *showComm*, *showCommand*, and *showGroup*) are set to "yes" based on the user input. Error handling is implemented to detect unknown flags and ensure the exclusive use of either *comm* or *command*.

Phase 2

The script iterates through */proc* directories corresponding to running processes (numeric directories from 0-9). It checks if each directory still exists during the loop, addressing the dynamic nature of the */proc* directory.

Phase 3

Within the for loop, the script extracts essential information from the 'status' file of each process. This includes the process ID (*pid*), command name (*cmd*), numeric user ID (*uid*), RSS value (*rss*), and numeric group ID (*gid*). The extraction involves using commands like *grep* and *sed* to isolate the relevant information.

Phase 3b

Numeric user and group IDs are converted to symbolic names (*uid* and *gid*) by referencing information from */etc/passwd* and */etc/group* file. The *awk* command is used to search for the corresponding names.

Phase 4

To address the issue of numerical ordering in the */proc* directories, the script writes the extracted process details to a temporary file (*$tmpFile*). An output only prints a given category if specified by the user. The *printf* command is used to format the output, ensuring neat alignment.

Phase 5

The script prints headers to the terminal based on user-specified flags. It then uses the *sort* command to arrange entries in the temporary file by process ID. Finally, the sorted data is presented on the terminal, and the temporary file is removed.