Sam Dinkelman

CIS 360

02/03/2019

Lab Report 2

Introduction:

The purpose of this assignment was to write a script that would look at some values of the processes in the /proc directory in linux. The requirements were that it use at least two command line arguments, a help option with syntax and also that it provide at least three different values.

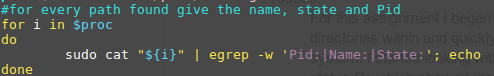
Process:

For this assignment I began by looking through the /proc directory and around at the directories within and quickly realized that most of the directories were processes on the system. I started looking within them and found that the most interesting file was the status file which showed the process name, state and pid. I decided that for this assignment I would try to display information based on this file here.

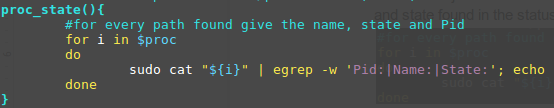
So I began by getting through the directories to the status file, in my bash script I used a variable and stored the output of an ls command into that variable called proc. I used the -d command to make sure it did not recurse through directories and the \*[[:digit:]]\* specifier in the path makes sure that ls only looks at directories that begin with a digit.



I also created a function that contained the following for loop which goes through the variable we just made with all of the paths and cats the file while looking at the pid, name and state found in the status file.

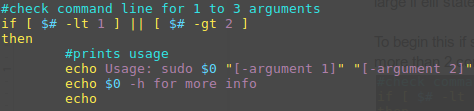


Here is that function in full:

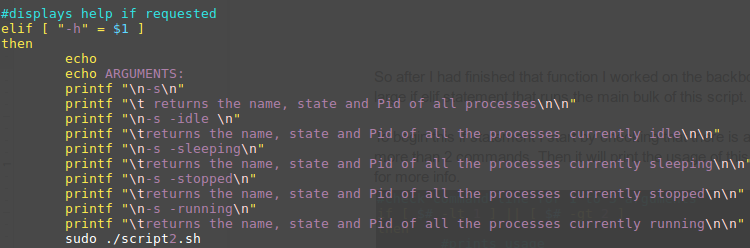


So after I had finished that function I worked on the backbone of this script which is the large if elif statement that runs the main bulk of this script.

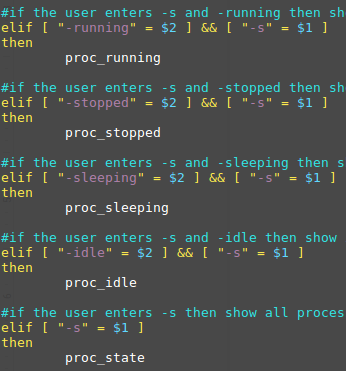
To begin this if statement I start by checking that there is at least one command and not more than 2 commands. Then it will print the usage of this script and a user can enter -h for more info.



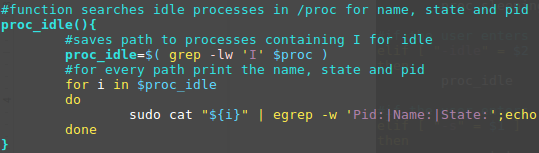
Then next I have my help prompts which print information about how to use the script and its options and gives a short description of each option. After the help is displayed the script is called again so it prints the usage again.



Then I begin checking user input against the strings that pertain to each option listed in the help info. All of these are essentially the same it just checks the second string of user input first and if that’s found then it will check the first string of user input and if they both match it will call the function associated with it.

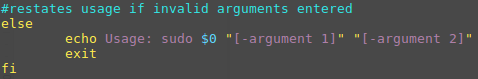


Next I will show the functions I created which are really just functions that first find the paths to the files that have matching strings of I, S, T or R and saves those paths into a variable. Then I run a for loop that goes through each of the paths saved in the variable and cats the name, pid and state of each of the processes that are found with that state.



The rest of the functions look exactly the same as this one besides a different variable name and string to search for.

The last few lines of this script simply state the usage of the script again and will print if no strings are found matching the required arguments or there are too many arguments and then exit.



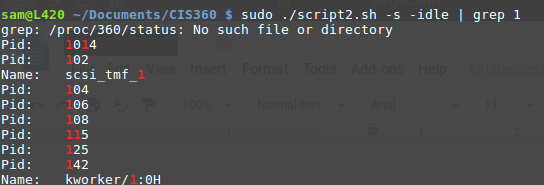
Testing:

In testing my script I first thought I should try and enter more arguments than the script would like and here are the results from that.

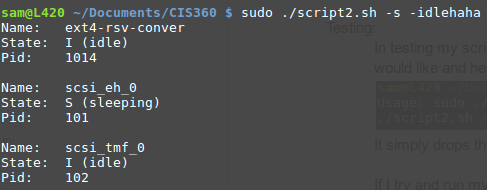


It simply drops through to the else statement at the end which prints usage again.

If I try and run my script as it should be run but then pipe it to grep the output to search for a specific process it also works and here are those results.



One problem I did find with my script was that if you entered two arguments but didn’t enter exactly the right string for the second argument then it would default to running the -s command and print all the processes. This is likely because I have the -s command as the last elif statement so it will default to it since it didn’t see any of the other strings in the second argument.



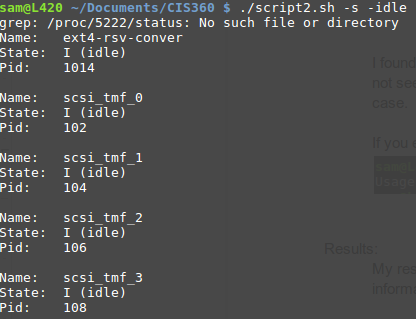
I found that this error will work for any string entered as the second argument since it will not see a second string that matches any of the elif cases and will drop through to the -s case.

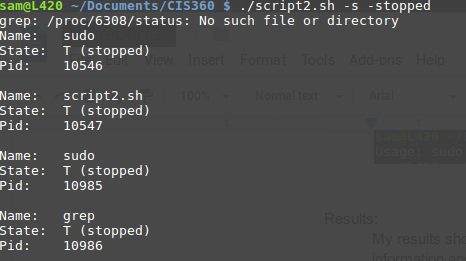
If you enter two arguments it can’t recognize it will print out the usage again.



Results:

My results shown below demonstrate that this script can give the user lots of different information and organize that information easily with simple commands. This script could also be expanded to support a wide variety of cases. The several that I have programmed for are idle, stopped, sleeping, and running which were the only ones I found on my particular distro.





Conclusions:

In conclusion this was a very enjoyable assignment I really felt like I learned a lot more about writing scripts that behave like programs in bash. I felt like I understood more the use of functions and variables in bash. I also feel like I got a good understanding of the /proc directory and it’s usage not only in my little script but in other scripts and programs.

References and acknowledgements:

I again used ryan’s tutorials for this assignment and I also used various postings on stack overflow mostly about different options of grep and cat that helped me with this assignment. All the work of this assignment was my own.