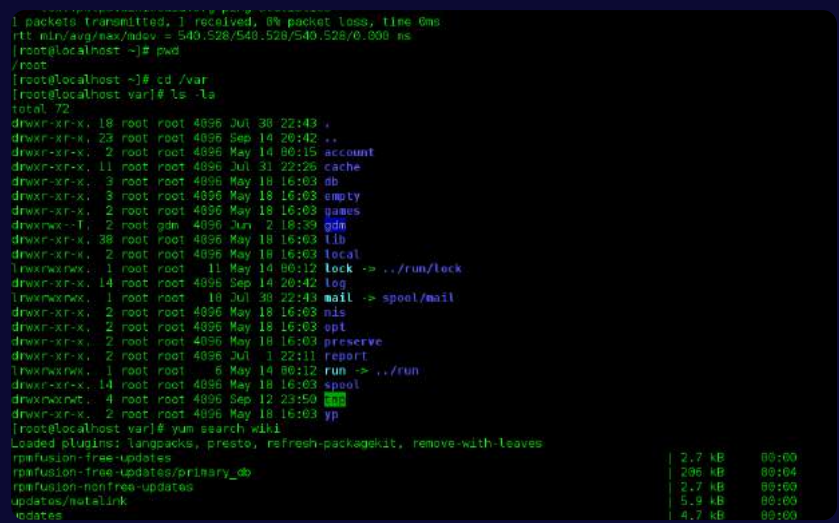
Effective Shell Scripting

Welcome to this guide about shell scripts. This presentation will show you how to write, debug and execute shell scripts with ease.

Retrieving Lines from a File



Find the line numbers

To extract a range of lines from a file with the shell, we can use a combination of commands.

Use Sed Command

We then utilize the sed command to display all the lines between the inputted start line and end line numbers.

Specify File and Range

First, specify the file name along with the starting and ending line numbers as arguments.

Removing Lines from a File

File and Keyword

Removing specific lines from one or more files can be done by searching for it's specified keyword.

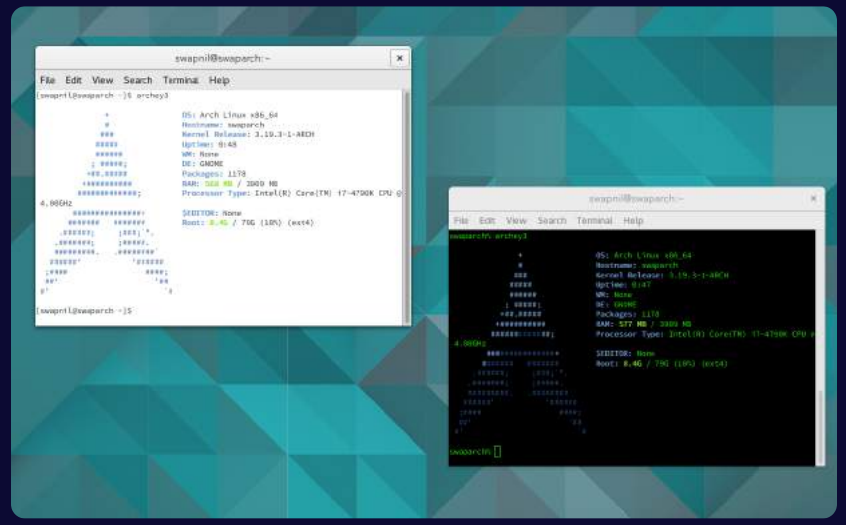
Grep Command

We use the grep command to search for the keyword. The -v option from grep will exclude lines containing the keyword while printing the rest to a defined temporary file t.

File After Removing Lines

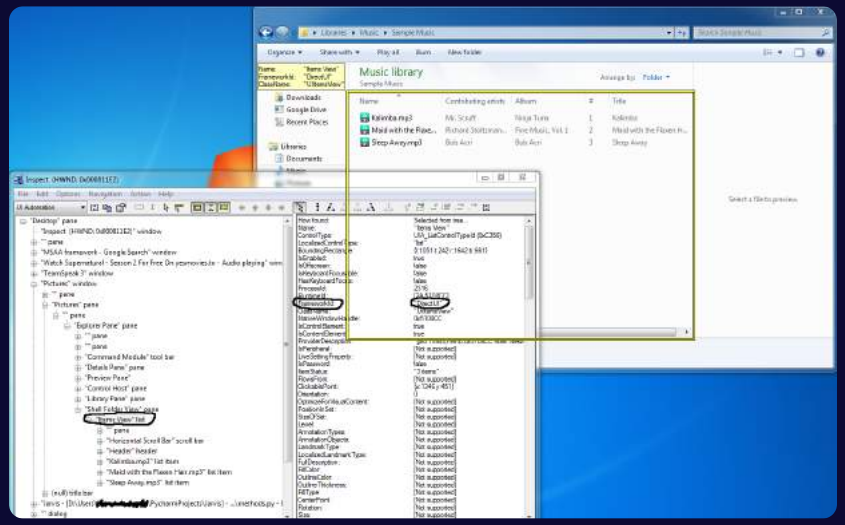
The mv command will replace the original file with the newly created t file containing only the lines without the excluded keyword.

Listing Files in Current Directory



Use ls Command

To get a list of all the files in the current directory we can simply use the ls command.



Output Returned

The output will show all the files in the current directory.

Checking File And Directory Existence

Input Desired Argument Number

We begin with the design of our script, reading user input of desired argument number.

Check if File or Directory

The script then will determine if the argument supplied is a file or a directory by using the -f and -d arguments with the if and elif statements.

Output Confirmation

The output will confirm if the argument is a file or a directory.

Command-Line Arguments & Variables

1

Command Line Passing

Command-line arguments are passed to a shell script by entering them after the script name on the command line.

2

Accessing Variables

We can access the values of passed arguments by utilizing the $1, $2, $3, and so on, variables in our script.

3

Debugging -x

The -x option for the shell will print out tracing information that can help debug complex scripts.

Comments

Use of Comments

Comments are important to add descriptive information for a script without affecting the final output.

Final Result

A good script should have clear, concise comments throughout the code to explain the purpose and function of each section for future reference.

Variables and Operators

1 Variable Types

Shell scripts use two types of variables: system variables and user-defined variables.

2 Mathematical Operators

Shell scripts use mathematical operators such as addition(+), subtraction(-), multiplication(*), division(/), and modulus(%) to perform math operations.

3 Conditional Operators

Conditional operators like <, >, == can help us perform comparisons between two variables in a script.

Flow Control Structures

If, then and Elif Statements

The if statement, with the then and elif statements can help us perform a series of actions subsequent to a conditional test result.

For Loop

A for loop can help us iterate through a series of items and perform actions for each one.

While Loop

A while loop can be used to iterate through code or input while a specified condition is met.

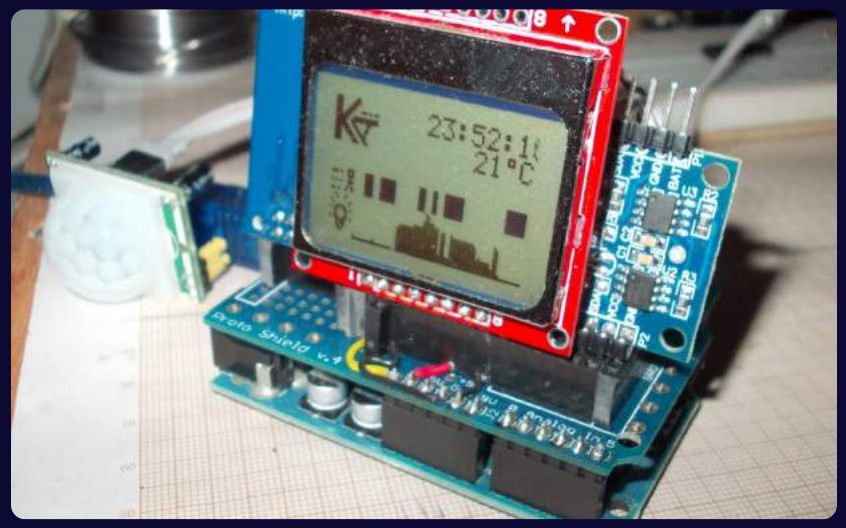
Input and Output

Standard Input - STDIN

The standard input for a script is typically the keyboard input.

Standard Output - STDOUT

The standard output will typically display output to the terminal window.



Standard Error - STDERR

The standard error is used by the shell to output errors related to the script. Errors can later be logged into a specified file.

Debugging

1

Set -x

The set -x command will print out the command that is being executed and all corresponding arguments.

2

Tracing Variables

The debug trap can be used to trace and output variables while the script is executed.

3

Error Handling

Proper error handling that logs errors will make it easier to find and fix problems in a script.

Conclusion

Start Writing Shell Scripts

We hope that this guide equips you with new knowledge and skills that you can apply in writing your shell scripts.

Happy Scripting!

Now that you have mastered shell scripting, you're ready to debug and execute your scripts with ease.