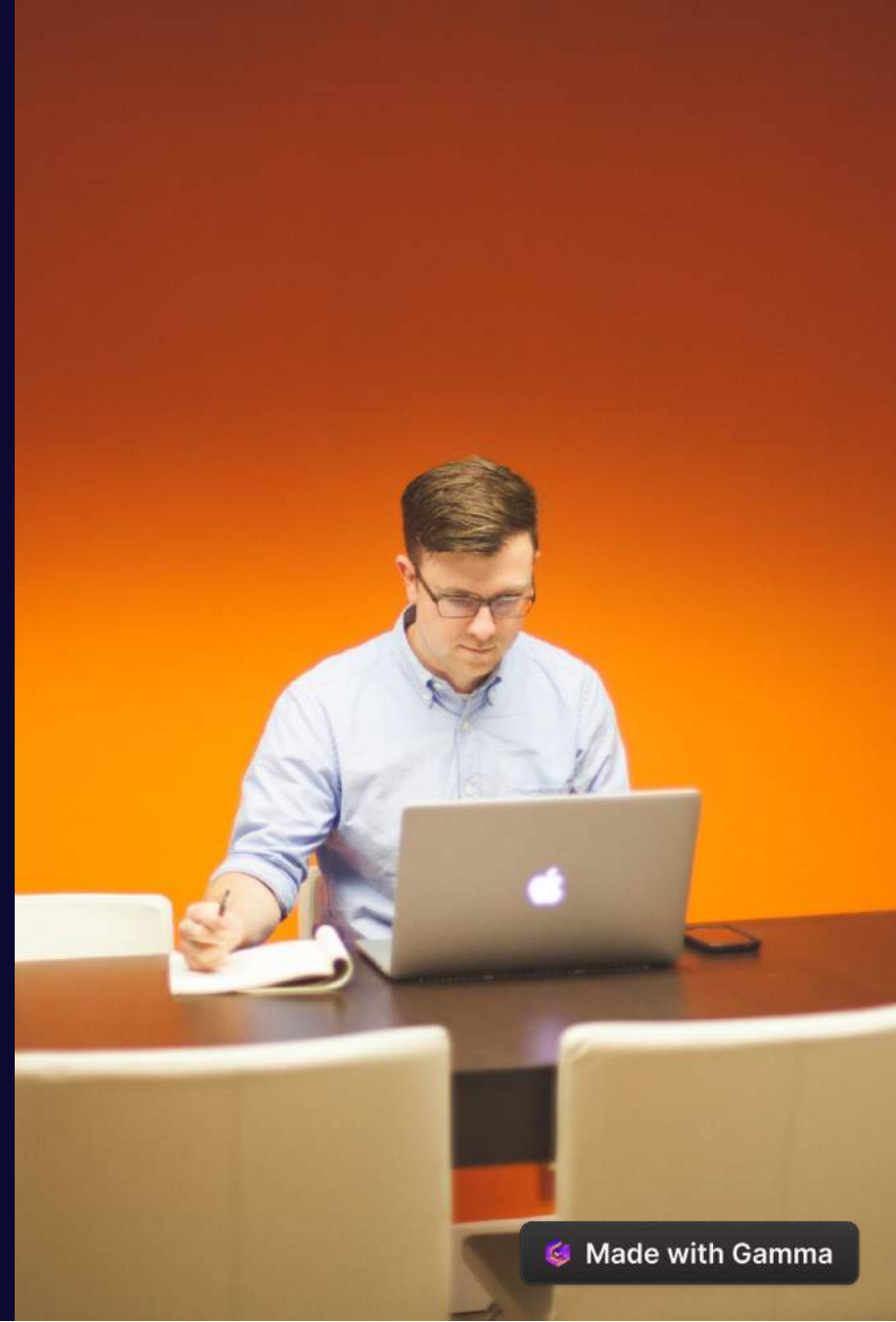


Effective Shell Scripting

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

```
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 540.528/540.528/540.528/0.000 ms
[root@localhost ~]# pwd
/root
[root@localhost ~]# cd /var
[root@localhost var]# ls -la
total 72
drwxr-xr-x. 18 root root 4096 Jul 30 22:43 .
drwxr-xr-x. 23 root root 4096 Sep 14 20:42 ..
drwxr-xr-x. 2 root root 4096 May 14 00:15 account
drwxr-xr-x. 11 root root 4096 Jul 31 22:26 cache
drwxr-xr-x. 3 root root 4096 May 18 16:03 db
drwxr-xr-x. 3 root root 4096 May 18 16:03 empty
drwxr-xr-x. 2 root root 4096 May 18 16:03 games
drwxr-xr-x. 2 root gdm 4096 Jun 2 10:39 gdm
drwxr-xr-x. 38 root root 4096 May 18 16:03 lib
drwxr-xr-x. 2 root root 4096 May 18 16:03 local
lrwxrwxrwx. 1 root root 11 May 14 00:12 lock -> ../run/lock
drwxr-xr-x. 14 root root 4096 Sep 14 20:42 log
lrwxrwxrwx. 1 root root 10 Jul 30 22:43 mail -> spool/mail
drwxr-xr-x. 2 root root 4096 May 18 16:03 nis
drwxr-xr-x. 2 root root 4096 May 18 16:03 opt
drwxr-xr-x. 2 root root 4096 May 18 16:03 preserve
drwxr-xr-x. 2 root root 4096 Jul 1 22:11 report
lrwxrwxrwx. 1 root root 6 May 14 00:12 run -> ../run
drwxr-xr-x. 14 root root 4096 May 18 16:03 spool
drwxr-xr-x. 4 root root 4096 Sep 12 23:50 tmp
drwxr-xr-x. 2 root root 4096 May 18 16:03 yp
[root@localhost var]# yum search wiki
Loaded plugins: langpacks, presto, refresh-packagekit, remove-with-leaves
rpmfusion-free-updates                2.7 kB    00:00
rpmfusion-free-updates/primary_db    296 kB    00:04
rpmfusion-nonfree-updates             2.7 kB    00:00
updates/metalink                     5.9 kB    00:00
updates                               4.7 kB    00:00
```

Find the line numbers

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



Specify File and Range

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

```
0. Sep 2015 bin -> usr/bin
19. Sep 09:31 boot
21. Sep 15:50 dev
19. Sep 09:32 etc
21. Sep 15:52 home
7 30. Sep 2015 lib -> usr/lib
34 23. Jul 2015 lib64 -> usr/lib
96 1. Aug 22:45 lost+found
396 30. Sep 2015 mnt
16 21. Sep 2015 opt
0 21. Sep 15:52 private -> /home/encrypted
4096 12. Aug 15:37 proc
560 21. Sep 15:50 root
7 30. Sep 2015 run
4096 30. Sep 2015 sbin -> usr/bin
0 21. Sep 15:51 srv
300 21. Sep 15:45 sys
4096 12. Aug 15:39 usr
14 4096 23. Jul 10:25
```

Use Sed Command

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

Removing Lines from a File

File and Keyword

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

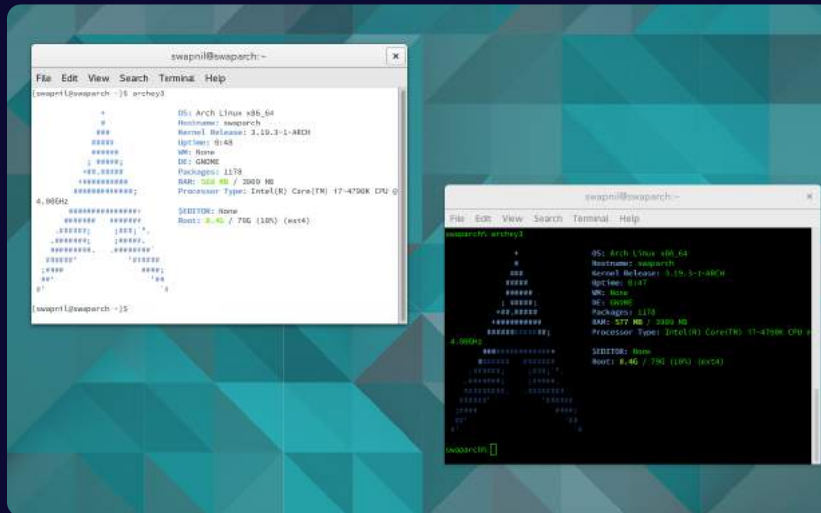
Grep Command

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



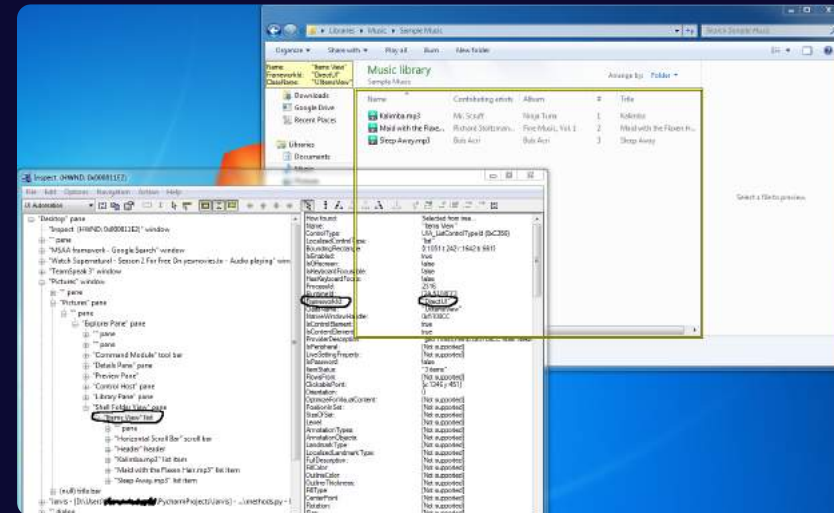
The image shows two terminal windows. The left window displays the output of the 'ls' command, showing a list of files and directories in the current directory. The right window shows the output of the 'ls -l' command, providing more detailed information about the files, including permissions, size, and modification date.

```
swapon@swapon:~$ ls
.  ..  .config  .local  .local/share  .ssh  Desktop  Downloads  Music  Pictures  Public  Templates  Videos

swapon@swapon:~$ ls -l
total 12
drwxr-xr-x 12 swapon swapon 4096 Jan 10 10:10 .
drwxr-xr-x  3 root   root    4096 Jan 10 10:10 ..
drwxr-xr-x  2 root   root    4096 Jan 10 10:10 .config
drwxr-xr-x  3 swapon swapon 4096 Jan 10 10:10 .local
drwxr-xr-x  2 swapon swapon 4096 Jan 10 10:10 .local/share
drwxr-xr-x  2 swapon swapon 4096 Jan 10 10:10 .ssh
drwxr-xr-x  2 root   root    4096 Jan 10 10:10 Desktop
drwxr-xr-x  2 root   root    4096 Jan 10 10:10 Downloads
drwxr-xr-x  2 root   root    4096 Jan 10 10:10 Music
drwxr-xr-x  2 root   root    4096 Jan 10 10:10 Pictures
drwxr-xr-x  2 root   root    4096 Jan 10 10:10 Public
drwxr-xr-x  2 root   root    4096 Jan 10 10:10 Templates
drwxr-xr-x  2 root   root    4096 Jan 10 10:10 Videos
```

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

Accessing Variables

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

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

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



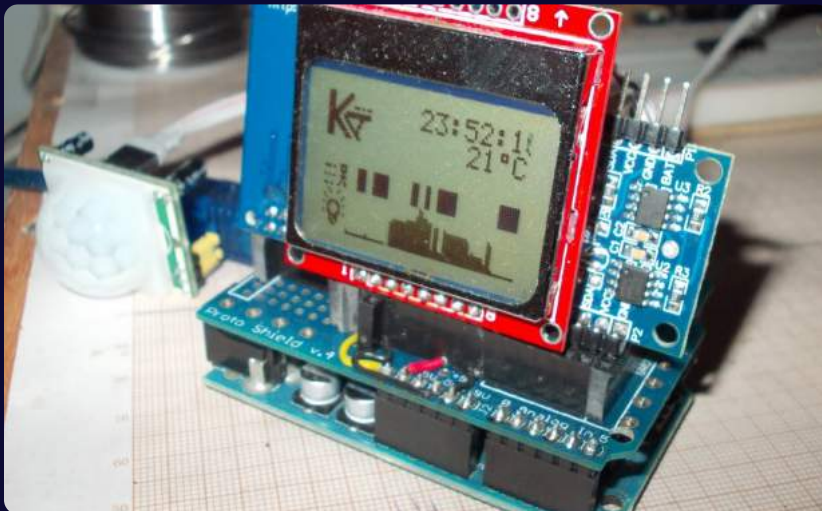
Standard Input - STDIN

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



Standard Output - STDOUT

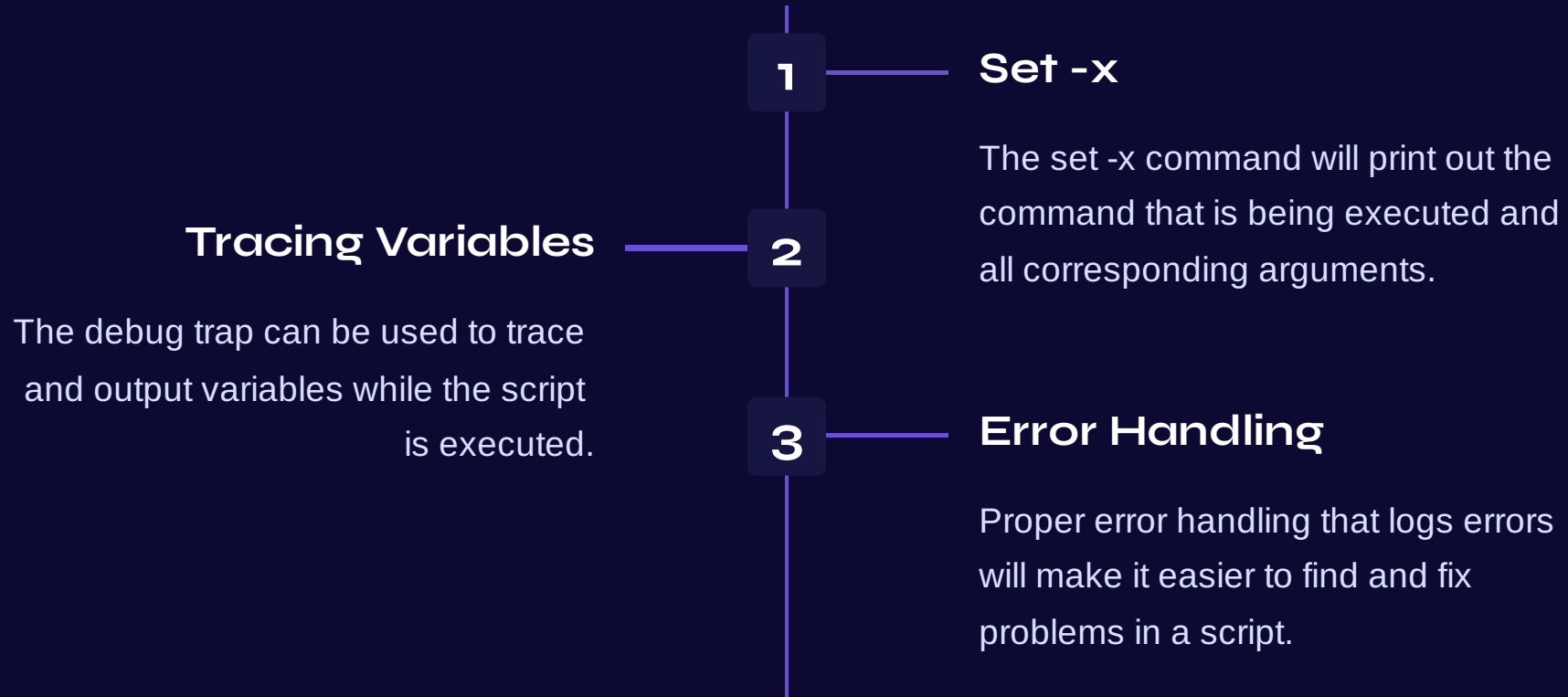
The standard output will typically display output to the terminal window.



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



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.