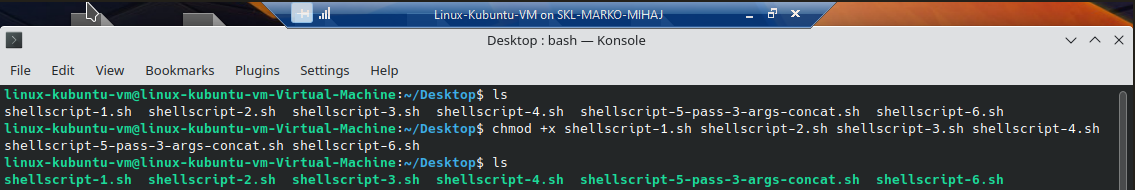
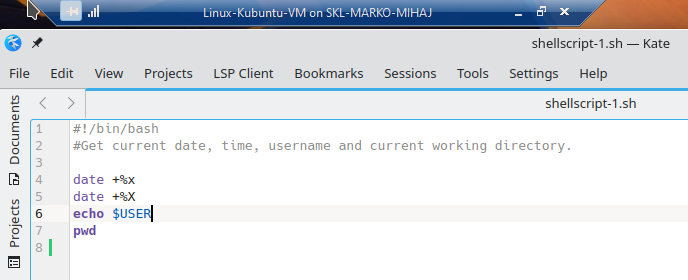
Bash Scripts Homework -Marko Mihajlov

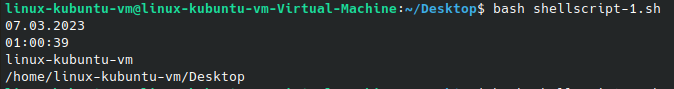


Prepping the Shell Scripts for execution by adding the execution permission to them.

1. Write a shell script to get the current date, time, username and current working directory.



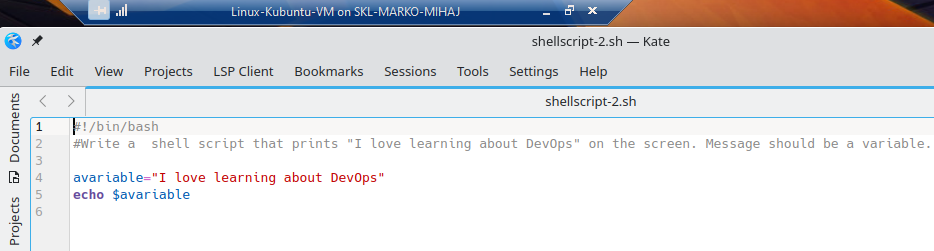
date – print or set the system date and time; flags %x locale’s date representation, %X locale’s time representation.



Running the shell script.

2. Write a shell script that prints “I love learning about DevOps” on the screen. Message should be a

variable.



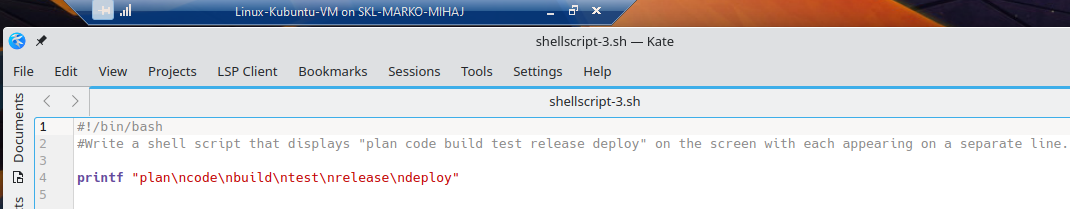
$avariable – printing the value stored inside of the string avariable..



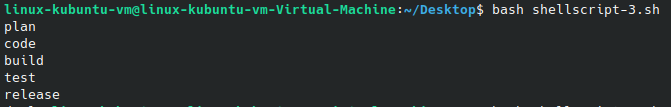
Running the shell script.

3. Write a shell script that displays “plan code build test release deploy” on the screen with each

appearing on a separate line.



\n – newline

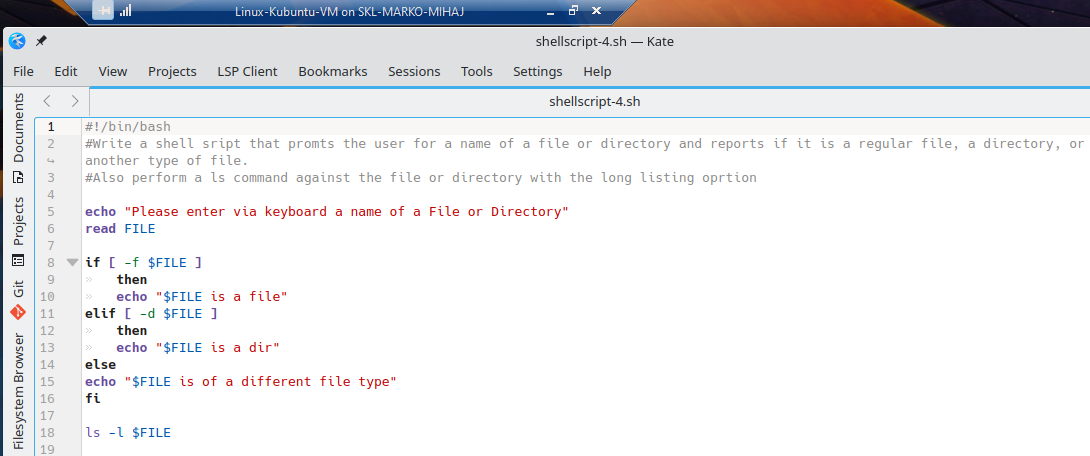


Running the shell script.

4. Write a shell script that prompts the user for a name of a file or directory and reports if it is a

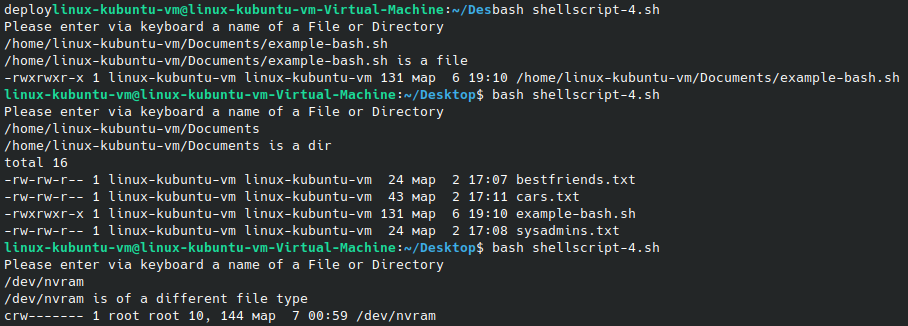
regular file, a directory, or another type of file. Also perform a ls command against the file or

directory with the long listing option.



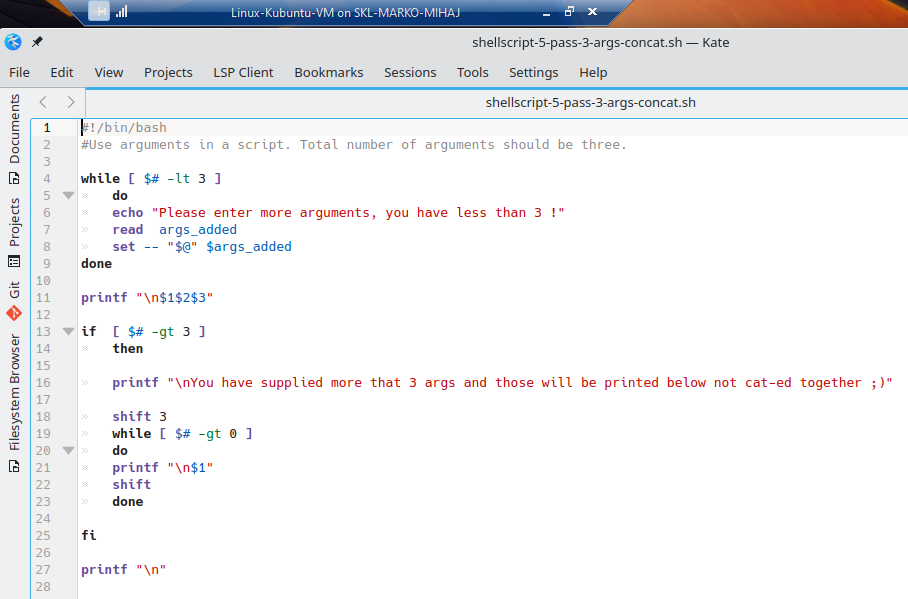
Using if statement to check if the name of the object entered is a file, dir or other filetype. Using the Cmd utility file – determine file type together with File Test Operators to check if a statement is true.

If not it goes to the next step – is it a directory.. and again if not it goes on to print that file is a different type. At the end proceeds to long list the item.



Running the shell script. 3 different times for the different scenarios. File, dir and other type of file.

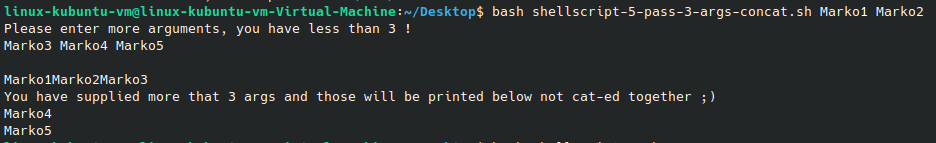
5. Use arguments in a script. Total number of arguments should be three.



Using while loop with a relational operator -lt (less than) to add more arguments until less than three becomes false then continues on. Inside of the loop it reads the values passed in the whole line from standard input and sets them one by one by the value of the current positional parameter starting with $1..$2 until it runs out of values from the read input.

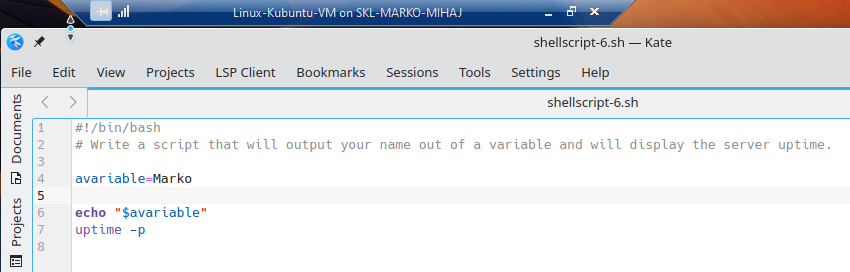
Then proceeds to print out the first 3. Afterwards if there are $# number of args is -gt (greater than) 3 it firsts gets rid of the first 3 already printed args with shift 3 and proceeds to print out the rest until $# is =0

At the end I print a newline, because it messes with the execution of the next bash script in following row. As it does in the first screenshot of the 4th script where the last line of the 3rd is in the same row.

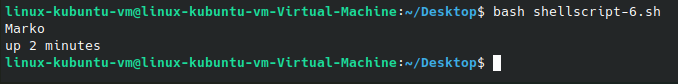


Running the shell script. At first passed 2 args to show that it asks me to input more because I have less than 3. If it enter 1 more to be exactly 3 it will just print the 3 of them out concatenated together. Instead I enter 3 more, the rest of them are printet out in new rows instead of joined.

6. Write a script that till output your name out of a variable and will display the server uptime



Uptime – Tell how long the system has been running; flag -p (pretty show uptime in pretty format)



Running the shell script.

End of file.