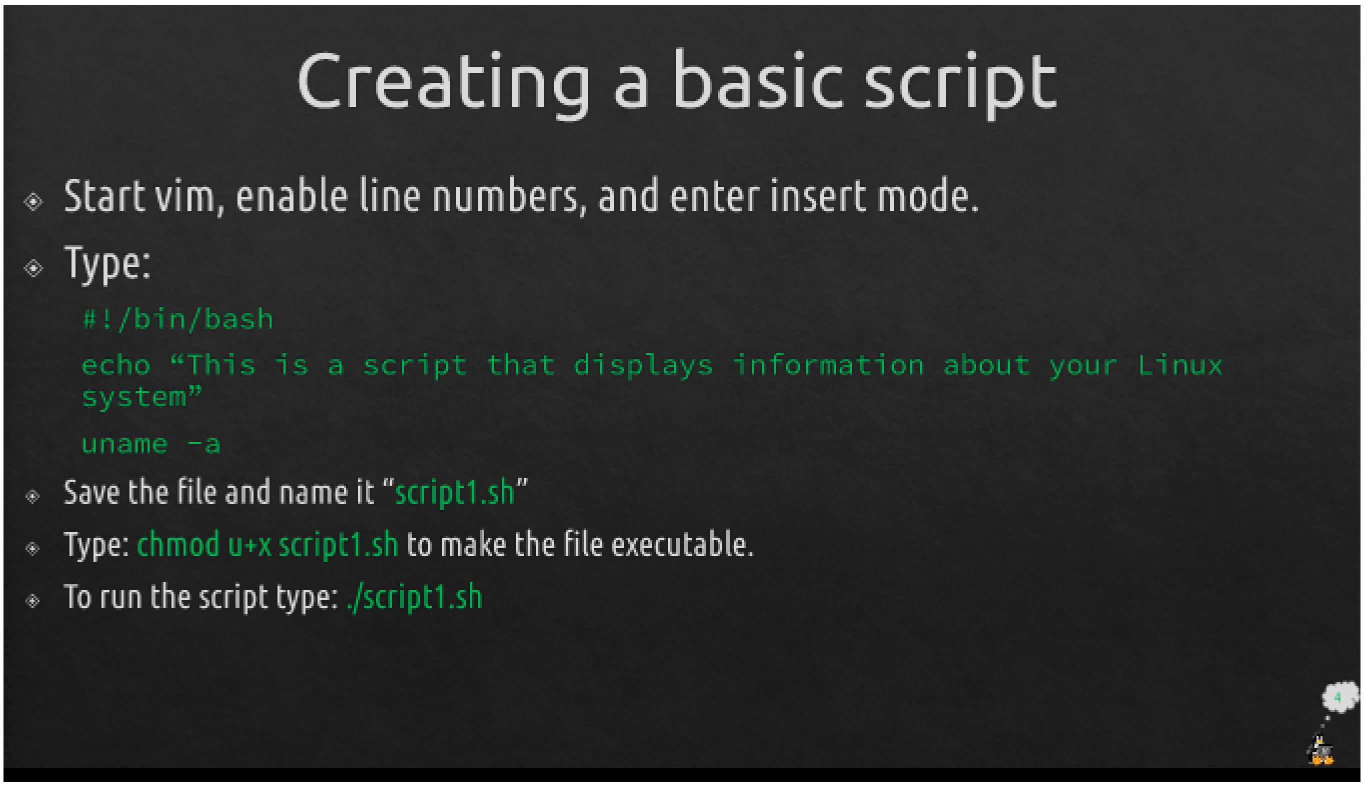
Lecture 8 | Shell scripting

Creating a basic script



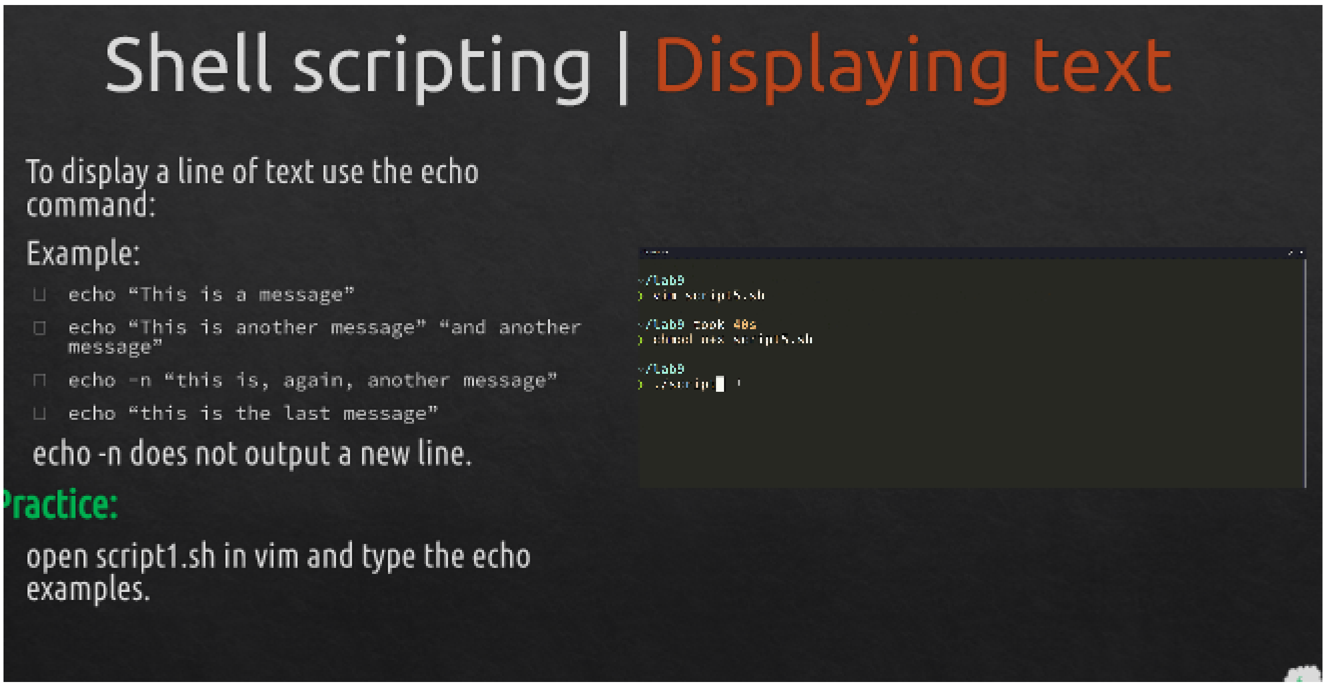
Creating a basic script

- Start vim, enable line numbers, and enter insert mode.
- Type:

```
#!/bin/bash
echo "This is a script that displays information about your Linux
system"
uname -a
```

- Save the file and name it "script1.sh"
- Type: chmod u+x script1.sh to make the file executable.
- To run the script type: ./script1.sh

Displaying text



Shell scripting | Displaying text

To display a line of text use the echo command:

Example:

```
echo "This is a message"
echo "This is another message" "and another
message"
echo -n "this is, again, another message"
echo "this is the last message"
```

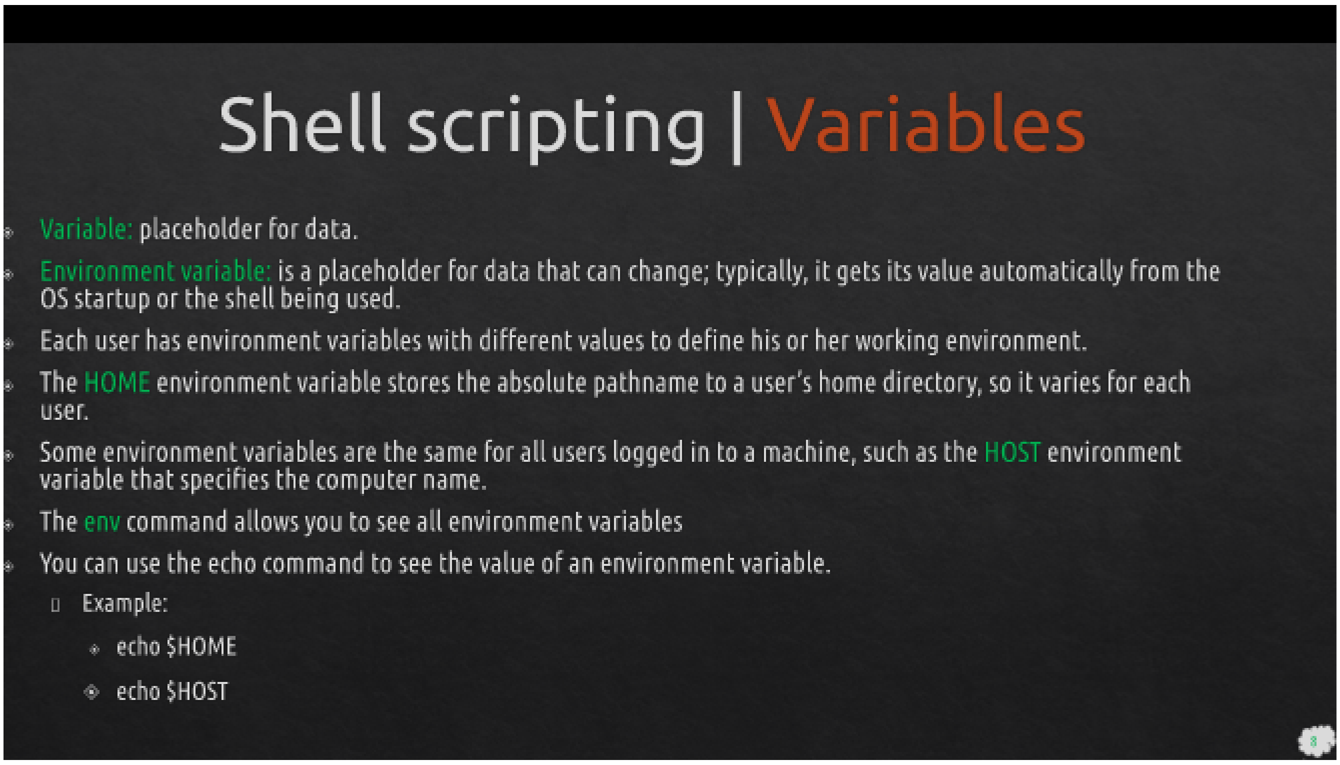
echo -n does not output a new line.

Practice:

open script1.sh in vim and type the echo examples.

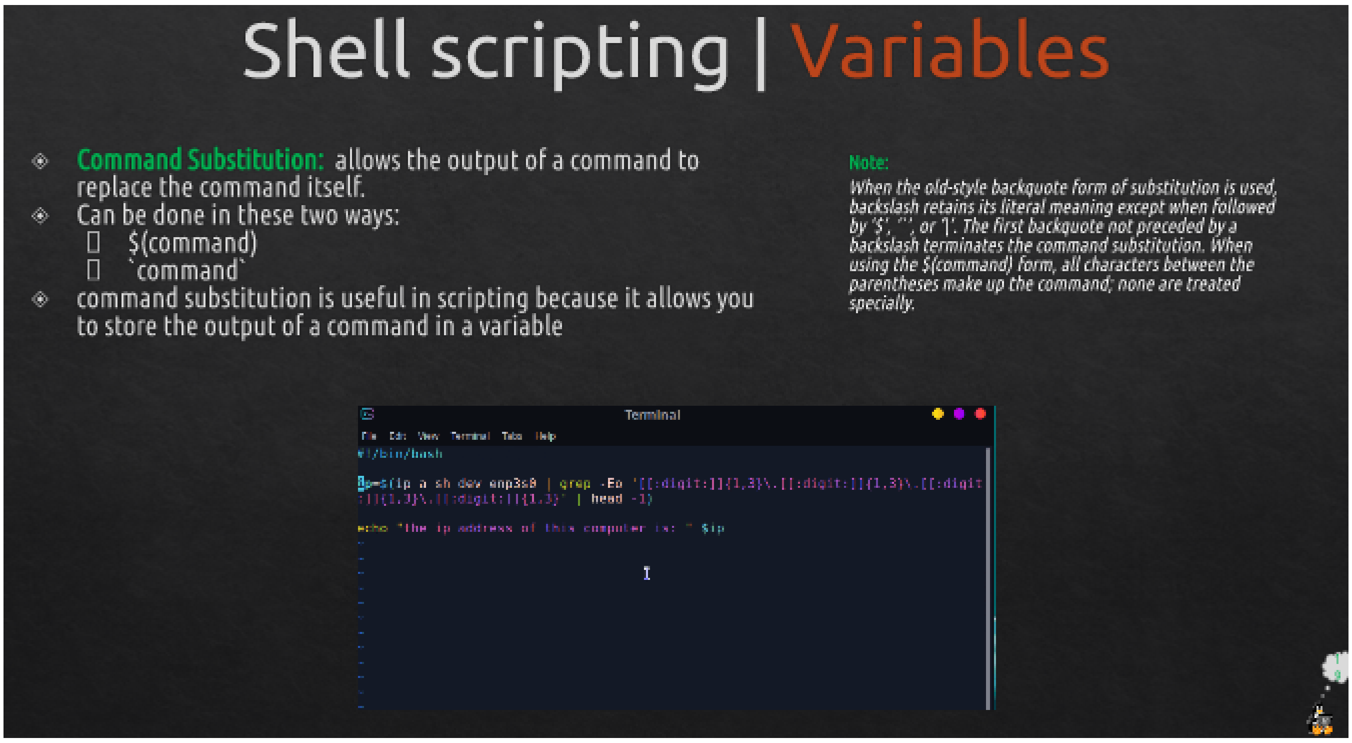
```
~/lab9
❯ vim script5.sh

~/lab9 took 40s
❯ chmod u+x script5.sh

~/lab9
❯ ./script
```

Working with variables



Shell scripting | Variables

- Variable: placeholder for data.
- Environment variable: is a placeholder for data that can change; typically, it gets its value automatically from the OS startup or the shell being used.
- Each user has environment variables with different values to define his or her working environment.
- The HOME environment variable stores the absolute pathname to a user's home directory, so it varies for each user.
- Some environment variables are the same for all users logged in to a machine, such as the HOST environment variable that specifies the computer name.
- The env command allows you to see all environment variables
- You can use the echo command to see the value of an environment variable.
  - Example:
    - echo $HOME
    - echo $HOST



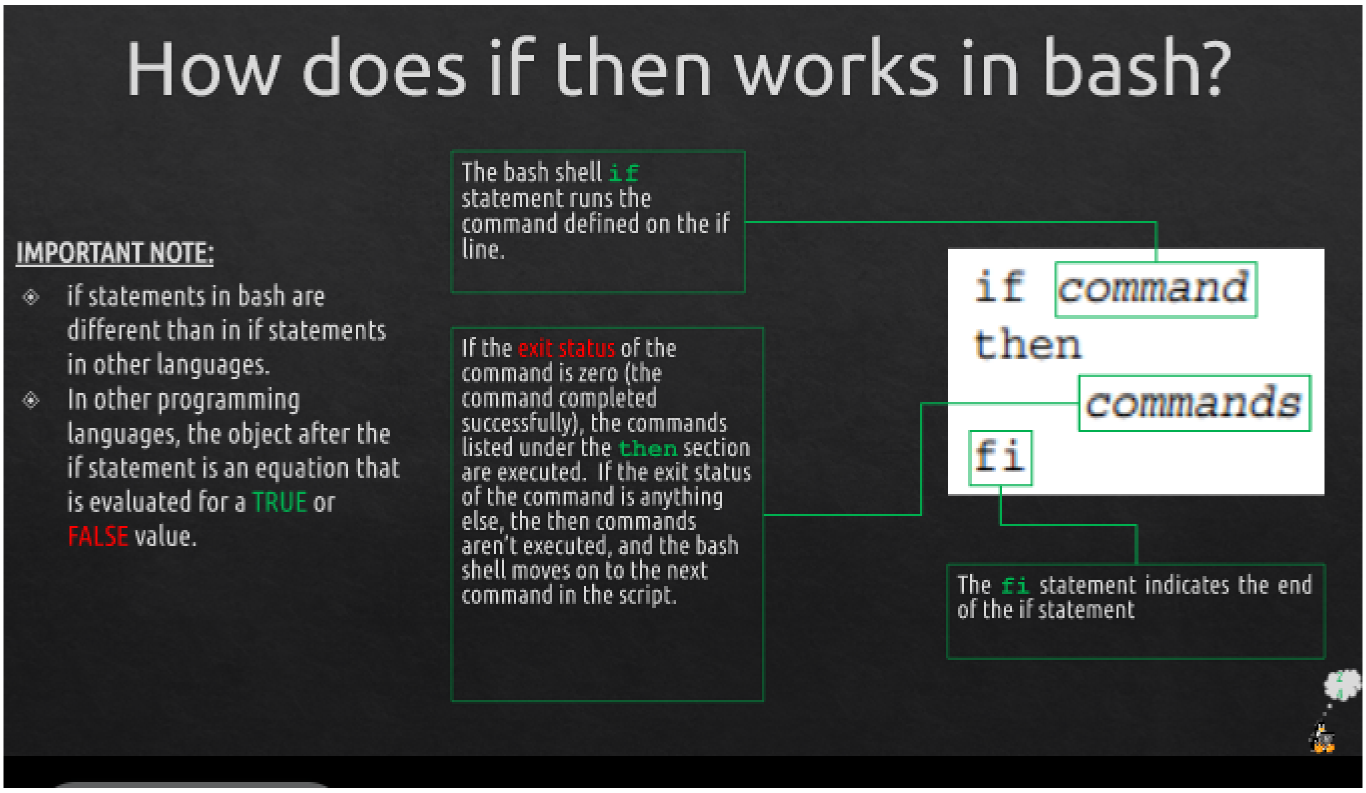
Shell scripting | Variables

- Command Substitution: allows the output of a command to replace the command itself.
- Can be done in these two ways:
  - $(command)
  - \`command\`
- command substitution is useful in scripting because it allows you to store the output of a command in a variable

Note:
When the old-style backquote form of substitution is used, backslash retains its literal meaning except when followed by '$', '`', or '\'. The first backquote not preceded by a backslash terminates the command substitution. When using the $(command) form, all characters between the parentheses make up the command; none are treated specially.

```
#!/bin/bash

ip=$(ip a sh dev enp3s0 | grep -Eo '[[:digit:]]{1,3}\.[[:digit:]]{1,3}\.[[:digit
:]]{1,3}\.[[:digit:]]{1,3}' | head -1)

echo "The ip address of this computer is: " $ip
```

How does if then works in bash?



How does if then works in bash?

IMPORTANT NOTE:

- if statements in bash are different than in if statements in other languages.
- In other programming languages, the object after the if statement is an equation that is evaluated for a TRUE or FALSE value.

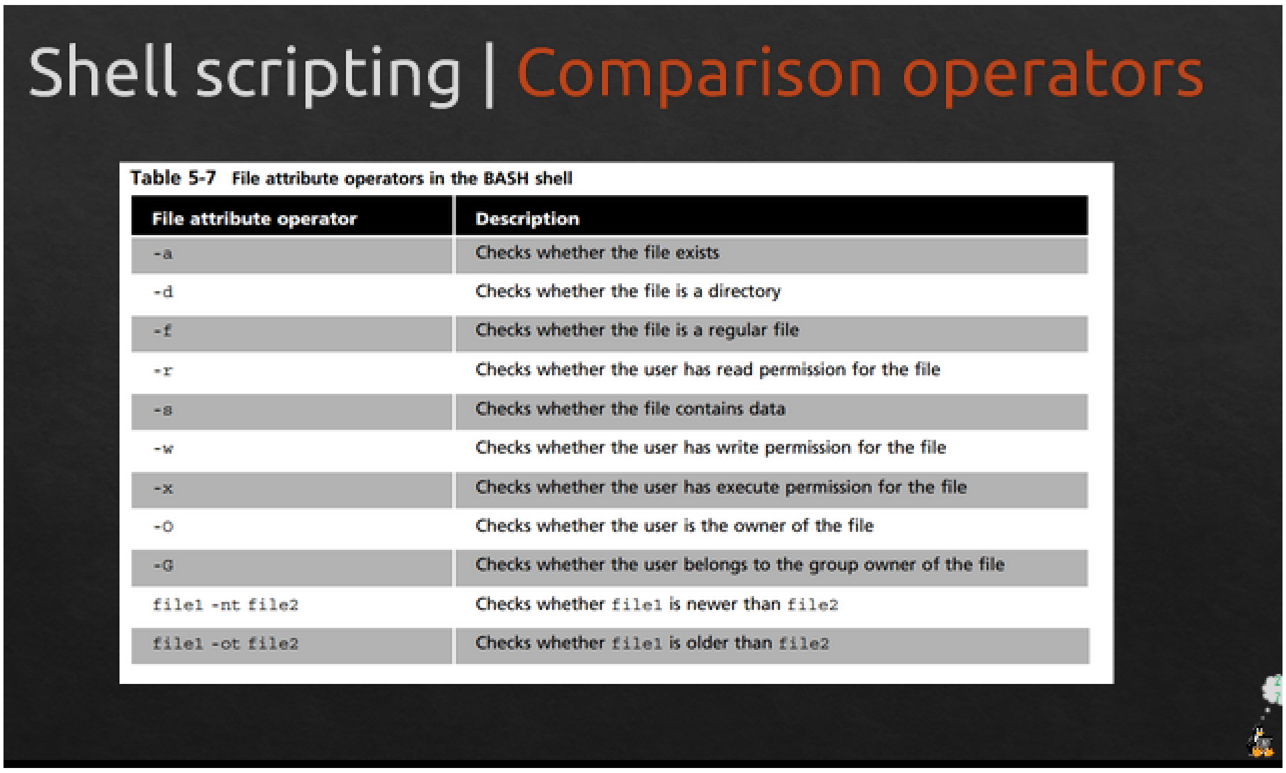
The bash shell `if` statement runs the command defined on the if line.

If the exit status of the command is zero (the command completed successfully), the commands listed under the `then` section are executed. If the exit status of the command is anything else, the then commands aren't executed, and the bash shell moves on to the next command in the script.

```
if command
then
    commands
fi
```

The `fi` statement indicates the end of the if statement

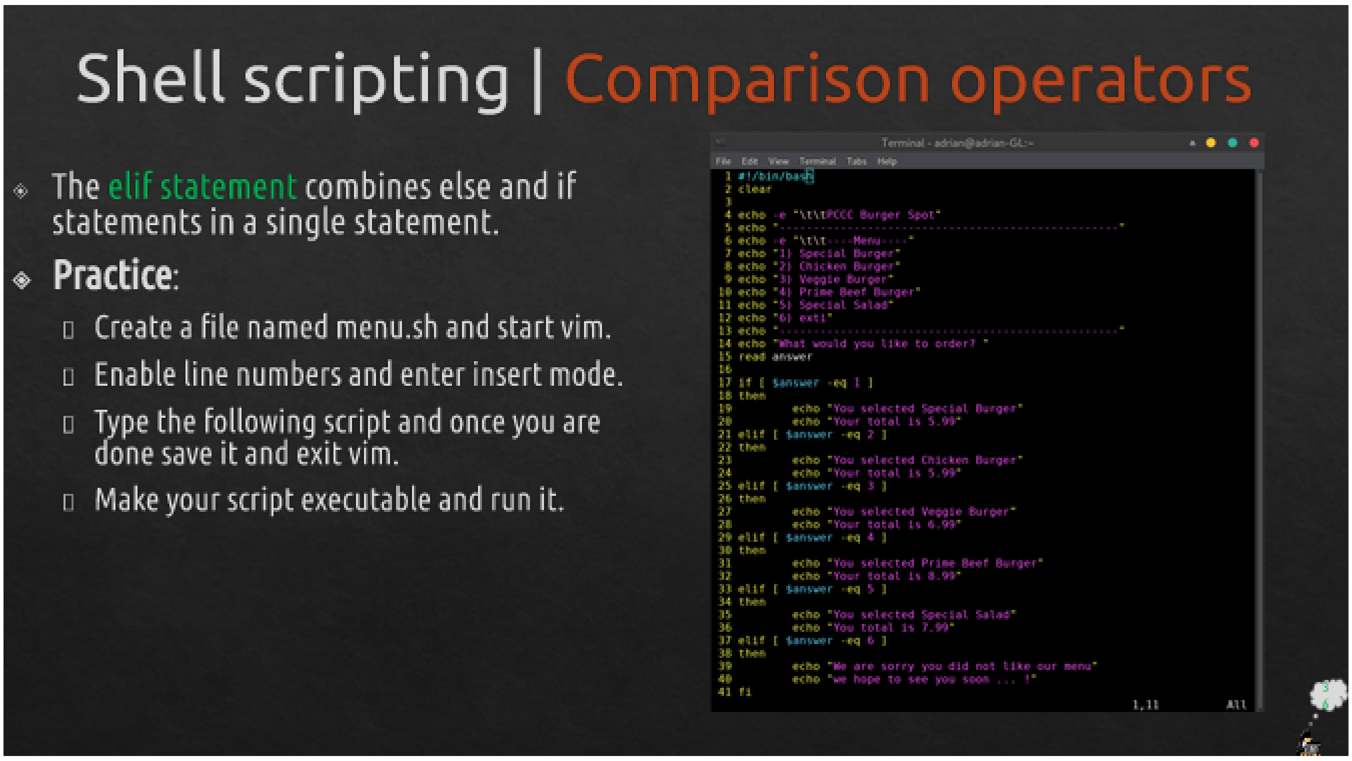
Operators



Shell scripting | Comparison operators

Table 5-7 File attribute operators in the BASH shell

File attribute operator	Description
`-a`	Checks whether the file exists
`-d`	Checks whether the file is a directory
`-f`	Checks whether the file is a regular file
`-r`	Checks whether the user has read permission for the file
`-s`	Checks whether the file contains data
`-w`	Checks whether the user has write permission for the file
`-x`	Checks whether the user has execute permission for the file
`-O`	Checks whether the user is the owner of the file
`-G`	Checks whether the user belongs to the group owner of the file
`file1 -nt file2`	Checks whether `file1` is newer than `file2`
`file1 -ot file2`	Checks whether `file1` is older than `file2`



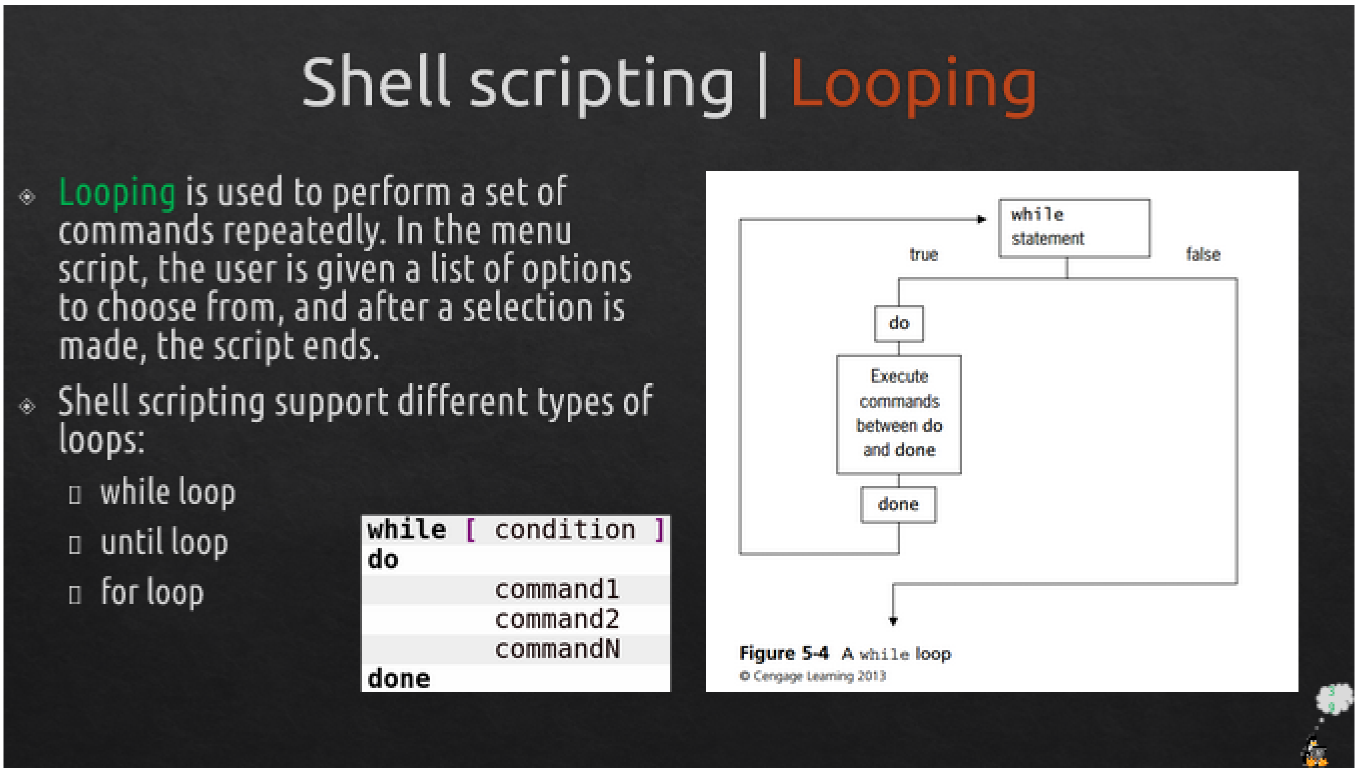
Shell scripting | Comparison operators

- The elif statement combines else and if statements in a single statement.
- **Practice:**
  - Create a file named menu.sh and start vim.
  - Enable line numbers and enter insert mode.
  - Type the following script and once you are done save it and exit vim.
  - Make your script executable and run it.

Terminal - adrian@adrian-GL:~
File Edit View Terminal Tabs Help

```
#!/bin/bash
clear

echo -e "\t\t\tPCCC Burger Spot"
echo "------------------------------------------------------------"
echo -e "\t\t\t----Menu----"
echo "1) Special Burger"
echo "2) Chicken Burger"
echo "3) Veggie Burger"
echo "4) Prime Beef Burger"
echo "5) Special Salad"
echo "6) exti"
echo "------------------------------------------------------------"
echo "What would you like to order? "
read answer

if [ $answer -eq 1 ]
then
        echo "You selected Special Burger"
        echo "Your total is 5.99"
elif [ $answer -eq 2 ]
then
        echo "You selected Chicken Burger"
        echo "Your total is 5.99"
elif [ $answer -eq 3 ]
then
        echo "You selected Veggie Burger"
        echo "Your total is 6.99"
elif [ $answer -eq 4 ]
then
        echo "You selected Prime Beef Burger"
        echo "Your total is 8.99"
elif [ $answer -eq 5 ]
then
        echo "You selected Special Salad"
        echo "You total is 7.99"
elif [ $answer -eq 6 ]
then
        echo "We are sorry you did not like our menu"
        echo "we hope to see you soon ... !"
fi
                                                    1,11          All
```

Looping



Shell scripting | Looping

- Looping is used to perform a set of commands repeatedly. In the menu script, the user is given a list of options to choose from, and after a selection is made, the script ends.
- Shell scripting support different types of loops:
  - while loop
  - until loop
  - for loop

```
while [ condition ]
do
          command1
          command2
          commandN
done
```

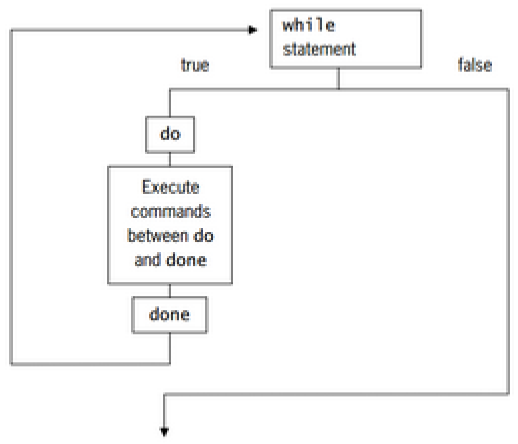


Figure 5-4 A while loop
© Cengage Learning 2013