

## Notes Lab8

# what is a shell scripting 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

## Some Examples how to use the Variables scripting

## Shell scripting | Variables

The **positional parameter method** uses the order of arguments in a command to assign values to variables on the command line.

Variables from \$0 to \$9 are available, and their values are defined by what the user enters.

**Table 5-6 Positional parameters**

Positional parameter	Description	Example
\$0	Represents the name of the script	./scr4 (./scr4 is position 0)
\$1 to \$9	\$1 represents the first argument, \$2 represents the second argument, and so on	./scr4 /home (./scr4 is position 0 and /home is position 1) ./scr4 /home scr1 (./scr4 is position 0, /home is position 1, and scr1 is position 2)
\$*	Represents all the positional parameters except 0	/home scr1 (just /home and scr1)
\$#	Represents the number of arguments that have a value	./scr4 /home scr1 echo \$# (\$* represents positions 1 and 2, which are /home and scr1)

## Another type of shell scripting is "Conditions" (IF Statement)

### Shell scripting | conditions

The **if statement** is used to carry out certain commands based on testing a condition and the exit status of the command.

If statements are used to control how the script will execute.

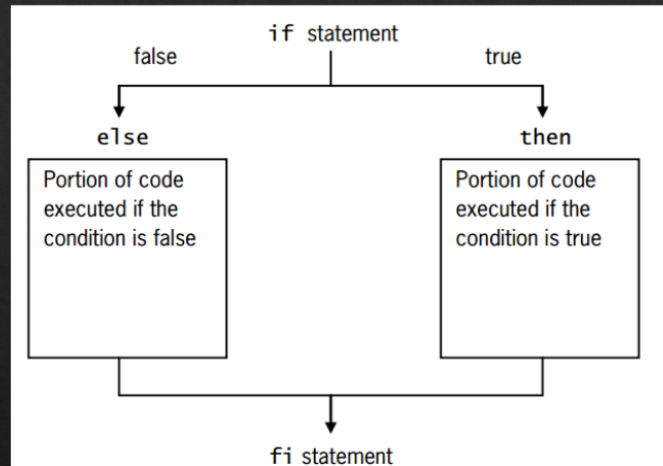
For instance, you might want a portion of the script to run if the user is in the Marketing Department and have another portion run if the user is in Human Resources.

Or you may want to run another command if the a given command executes successfully.

**if statement**—Starts the condition being tested

**then statement**—Starts the portion of code specifying what to do if the condition evaluates to true

**else statement**—Starts the portion of code specifying what to do if the condition evaluates to false



## Different type of shell scripting is "Comparison Operators" with some descriptions.

### 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

## Comparison Operators with the "String" mode

# Shell scripting | Comparison operators

## String Comparison

Comparison	Description	Example
<code>str1 = str2</code>	Checks if str1 is the same as string str2	<code>If [ \$str1 = \$str2 ]</code>
<code>str1 != str2</code>	Checks if str1 is not the same as str2	<code>If [ \$str1 != \$str2 ]</code>
<code>str1 &lt; str2</code>	Checks if str1 is less than str2	<code>If [ \$str1 &lt; \$str2 ]</code>
<code>str1 \&gt; str2</code>	Checks if str1 is greater than str2	<code>If [ \$str1 &gt; \$str2 ]</code>
<code>-n str1</code>	Checks if str1 has a length greater than zero	<code>If [ \$str1 -n ]</code>
<code>-z str1</code>	Checks if str1 has a length of zero	<code>If [ \$str1 -z ]</code>

## Comparison Operators with the "Numeric" mode

# Shell scripting | Comparison operators

## Numeric Comparison

Comparison	Description	Example
<code>n1 -eq n2</code>	Checks if n1 is equal to n2	<code>If [ \$n1 -eq \$n2 ]</code>
<code>n1 -ge n2</code>	Checks if n1 is greater than or equal to n2	<code>If [ \$n1 -ge \$n2 ]</code>
<code>n1 -gt n2</code>	Checks if n1 is greater than n2	<code>If [ \$n1 -gt \$n2 ]</code>
<code>n1 -le n2</code>	Checks if n1 is less than or equal to n2	<code>If [ \$n1 -le \$n2 ]</code>
<code>n1 -lt n2</code>	Checks if n1 is less than n2	<code>If [ \$n1 -lt \$n2 ]</code>
<code>n1 -ne n2</code>	Checks if n1 is not equal to n2	<code>If [ \$n1 -ne \$n2 ]</code>

Other type of shell scripting is "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
```

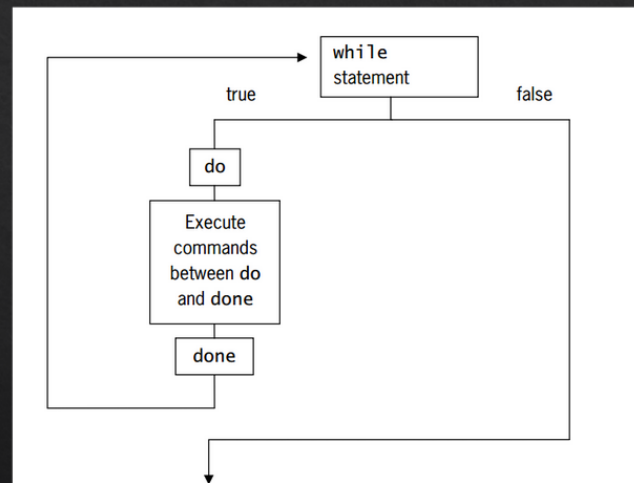


Figure 5-4 A while loop