

14.1.4 Scripting Facts

At its most basic level, a bash script is a set of commands stored in a file. When the bash shell reads the file, it executes the commands as if they were typed at the keyboard.

This lesson covers the following topics:

- Basic bash script rules and components
- The time command

Basic Bash Script Rules and Components



There are basic rules that govern how a bash script should be written and run.

- Specify the bash shell is used to run the script.
- Use comments to explain what the script does.
- Use **exit 0** to end the script.
- Assign execute permissions to the script with the **chmod** command.
- Use one of the following methods to run the script:
 - Add the folder that contains the script to the **PATH** environment variable, then enter the script name at the shell prompt.
 - Save the script in a folder that is already in the **PATH**, such as **/usr/bin** or **/bin**, then enter the name of the script.
 - Type the full path name to the script to run the script from anywhere.
 - Type **./script_name** to run the script if it resides in the current directory. (**./** indicates the present working directory.)

The following table lists some simple scripting components:

Component	Description	Examples
Shell declaration	The shell declaration should come first in the script. It starts with a number sign and exclamation point (#!), followed by the path to the shell executable. A common nickname for this line is shebang.	#!/bin/bash specifies that the script will run in the bash shell.
Comments	Comments begin with a number sign (#). The shell ignores these lines when running the script. Comments help communicate how	

	the script was constructed and what it is designed to do.	
Commands	<p>When a script runs, it executes commands as if they were entered at the command line. Commands can be typed on a single line or separated using a semi-colon (;). The echo command displays information on the screen. It can display a literal value or a variable. Keep the following in mind:</p> <ul style="list-style-type: none"> • It is a good practice to always use quotes when you want a variable to represent a literal value. • Use a dollar sign (\$) to display the value of a variable. • Use a backslash (\) to display special characters. <p>The read command creates a variable and prompts the user to type in text. It assigns the value the user types to the variable. By default, the user input is treated as a text string.</p>	<pre>#!/bin/bash ls /home/user/Pictures exit 0</pre> <p>This script uses the ls command to list the contents of the <code>/home/user/Pictures</code> directory.</p> <p>echo "Hello, Mr. Smith." displays Hello, Mr. Smith. on the screen. echo \"Hello, Mr. Smith\" displays "Hello, Mr. Smith." on the screen. echo pwd displays pwd on the screen. echo \$variable1 displays the value of variable1. echo \variable1 displays the literal string <code>\$variable1</code>.</p> <pre>#!/bin/bash echo "What is your name?" read variable1 echo "Hello," \$variable1." exit 0</pre> <p>The script prints What is your name? on the screen, prompts the user for input, captures the user's input in a variable named <code>variable1</code>, then displays the contents of that variable on the screen.</p>
Variables	<p>Variables hold values that the script uses when running. These values can be either numbers or text. Keep the following in mind when using variables:</p> <ul style="list-style-type: none"> • Linux script variables are commonly written using all capital letters. This helps programmers quickly identify them. • When creating variables, place the equals sign (=) immediately after the variable with no space. If a space follows the variable name, the script treats it as a command, and tries to execute it. 	<p>variable1=Hello assigns variable1 the value of Hello.</p> <p>variable1 = Hello causes an error because the script tries to run the command <code>variable1</code>, which by default does not exist.</p> <p>variable1="Hello, Mr. Smith" assigns variable1 the value of Hello, Mr. Smith.</p> <p>variable1=Hello, Mr. Smith assigns variable1 the value of Hello, then displays an error because it treats Mr. as a command and tries to execute it.</p> <p>variable1=pwd assigns variable1 the value of pwd.</p> <p>variable1= pwd assigns variable1 the value of the result of running the pwd command. For example, <code>/home/jdoe</code>.</p> <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"> <p>You can manipulate environment variables from within a script. For example, you could modify the</p> </div>

	<ul style="list-style-type: none"> • Use a space after the equals sign (=) only when you want the variable to be the output of a command. • Use quotes if a variable value has a space in it. It is a good practice to always use quotes when you want a variable to represent a character string. 	<div>  MAILTO environment variable by including MAILTO=root in a script. This will cause notifications or other conditional events to be mailed to the root superuser account by default. </div>
Variable declarations	<p>The declare -i command is used to type a variable as an integer. It can only contain whole numbers. It can't contain text or numbers with decimal places.</p> <p>The declare -f command is used to display all defined functions or just a specific defined function.</p>	<pre>#!/bin/bash declare -i num1 declare -i num2 declare -i total num1=7 num2=5 total=num1+numw echo \$total exit 0</pre> <p>The output of this script is the integer 12, because the shell treats the variable values as integers.</p> <div>  <p>Numeric text strings can also be converted to integers using the following format: echo \$[<i>num1</i>+<i>num2</i>] gives an output of 12 regardless of whether the declare command is used.</p> </div>

The time Command

The time command is used to determine how long a given command takes to run. It is useful for testing the performance of your scripts and commands.

Command	Description	Examples
time <i>command</i>	The output shows three values as follows:	time wget wget https://wordpress.org/latest.zip use the wget command to download the zip file, but also displays the time it took:

	<ul style="list-style-type: none">• real - The time from the moment the Enter key was pressed until the moment the command is completed.• user - The amount of CPU time spent in user mode.• system - The amount of CPU time spent in kernel mode.	<div><div>real</div><div>0m1.604s</div></div> <div><div>user</div><div>0m0.042s</div></div> <div><div>sys</div><div>0m0.099s</div></div>
--	--	--

Copyright © 2022 TestOut Corporation All rights reserved.