02_Hello_World

April 13, 2024

1 Steps For SetUp

To set up your Ubuntu system for learning Bash scripting, follow these steps:

1. **Open Terminal**: You can open the terminal by pressing `Ctrl + Alt + T` or by searching for 'Terminal' in the application launcher.
2. **Check Bash Version**: Bash is pre-installed on most Linux distributions. You can check if it's installed and its version by typing `bash --version` in the terminal.
3. **Text Editor**: You'll need a text editor to write your scripts. Ubuntu comes with a basic editor called `gedit`. You can also use more advanced editors like `vim`, `nano`, or `emacs`. If you prefer a graphical editor, you can use `Visual Studio Code`, `Sublime Text`, or `Atom`.
4. **File Permissions**: To run a Bash script, you'll need to make it executable. You can do this with the `chmod` command. For example, if your script is named `script.sh`, you would type `chmod +x script.sh` in the terminal.
5. **Running Scripts**: To run your script, you can type `./script.sh` in the terminal. The `./` before the script name means "in the current directory".
6. **Learning Resources**: There are many online resources for learning Bash scripting. Some popular ones include the Bash Guide from the GNU project, the Bash scripting tutorial on the Linux Documentation Project, and various courses on platforms like Coursera and Udemy.

Remember, the key to learning Bash scripting (like any programming language) is practice. Try to automate small tasks on your computer using scripts to get hands-on experience.

2 Hello World!

Here's a simple "Hello, World!" program in Bash:

[ ]:
```
#!/bin/bash
echo "Hello, World!"
```

Let's break it down:

1. `#!/bin/bash`: This is called a shebang. It tells the system that this script should be executed with the Bash shell. It's always the first line in the script.
2. `echo "Hello, World!"`: The `echo` command is used to print its arguments to the standard output, which is usually the terminal. In this case, it prints the string "Hello, World!".

To run this script, save it to a file (for example, `hello_world.sh`), give it execute permissions with `chmod +x hello_world.sh`, and then run it with `./hello_world.sh`.

3 What is Shebang?

A shebang (`#!`) is a two-character sequence at the beginning of a script that is used to specify the interpreter for the execution of the script. It is followed by the path to the interpreter, such as `/bin/bash`, `/usr/bin/env python`, or `/usr/bin/perl`.

For example, in a Bash script, the shebang would be `#!/bin/bash`. This tells the system that the script should be executed using the Bash shell.

Similarly, in a Python script, the shebang might be `#!/usr/bin/env python`. This tells the system to use the `env` command to find the Python interpreter in the system's `PATH`.

The shebang must be the very first thing in the file, and it is not treated as a comment. It's only used if the script is executed directly, like `./script.sh` or `./script.py`. If the script is passed as an argument to the interpreter (like `bash script.sh` or `python script.py`), the shebang is ignored.

4 How to comments in shell script?

In shell scripting, you can add comments by using the `#` symbol. Any text following the `#` symbol on a line is considered a comment and is ignored by the shell interpreter. Here's an example:

[ ]:
```
#!/bin/bash
# This is a comment in a shell script
echo "Hello, World!" # This is a comment on the same line as a command
```

In this script, the first line is not a comment but a shebang that specifies the interpreter for the script. The second line is a comment that provides some information about the script. The third line has a command followed by a comment.

Note that shell scripting does not support multi-line comments like some other languages, so you will need to use a `#` at the beginning of each line you wish to comment out.

4.1 Multi-line comments

In Bash scripting, there's no direct way to create multi-line comments like in some other programming languages. However, there are a couple of workarounds:

1. **Using multiple single-line comments**: This is the most straightforward way. You simply put a `#` at the beginning of each line you want to comment.

[ ]:
```
# This is the first line of the comment
# This is the second line of the comment
# And this is the third line
```

2. **Using a Here Document (Heredoc)**: A Here Document is a section of a shell script that directs the shell to read input from the current source until a line containing only the

delimiter is seen. All of the lines read up to that point are then used as the standard input for a command. However, if you don't provide a command, it effectively acts as a multi-line comment.

[ ]:
```
: << 'END_COMMENT'
This is a multi-line comment
It can span multiple lines
END_COMMENT
```

In this example, `:` is a shell built-in command that does nothing, and `END_COMMENT` is the delimiter indicating the end of the Here Document. The shell will read all lines until it finds the `END_COMMENT` delimiter, effectively ignoring all those lines.

5 Thank You!