BeagleBone Tutorial: Introduction to Linux

Table of Contents

Introduction

This tutorial discusses a set of commands available in the Bash to interact with the Linux Operating System.

Prerequisite Tutorials

Users should ensure they are familiar with the document labelled ***BeagleBone Tutorial – Introduction*** before proceeding.

List of Required Equipment and Materials

- 1x BeagleBone Green Wireless
- 1x USB 2.0 A-Male to Micro B Cable (micro USB cable)
- A personal computer

Bash – Bourne Again SHell

The first thing a user will see on successful login to the BeagleBone is the ***GNU Project's Bourne Again SHell*** **(Bash)**. The Bash uses a Command Line Interface (CLI) that provides users with a set of commands that enable access to the kernel of the Debian Linux Operating System. It is different from other software programs that use a Graphic User Interface (GUI) such as the web browser on a personal computer.

While GUI's are designed for typical users, CLI's can provide tools more suitable tools for advanced users and developers. Consider the task of copying files from one directory to another directory. The source directory is labelled **pictures/** and it contains some files. The file extensions of the files in **pictures/** include **.jpeg**, and **.png**. All files with the file extension **.jpeg** will be copied to a new directory labelled **JPEG_FILES**. All files with the file extension **.png** will be copied to a new directory labelled **PNG_FILES**. To do this on a file manager employing a GUI, such as **Finder** on Mac OS X or **File Explorer** on Windows, the user would need to perform the steps below.

1. Navigate to the parent directory of the directory **pictures/**
2. Create the new directories
3. Navigate back to **pictures/**
4. Find all files with the **.jpeg** file extension
5. Copy the **.jpeg** files
6. Navigate to **pictures/JPEG_FILES/**
7. Paste the **.jpeg** files
8. Navigate back to **pictures/**
9. Find all files with the **.png** file extension
10. Copy the **.png** files
11. Navigate to **pictures/PNG_FILES/**
12. Paste the **.png** files

To do this same task on the Bash, the user would need to issue the commands shown below.

1. **$ cd /path/to/pictures/**
2. **$ mkdir ../JPEG_FILES/ ../PNG_FILES/**
3. **$ cp *.jpeg ../JPEG_FILES/**
4. **$ cp *.png ../PNG_FILES/**

The next section will examine some commonly used commands, operators and concepts available in the Bash.

NOTE:

- **To issue a command, type the command into the shell prompt, and press [Enter].**
- **The dollar sign '$' means it is a shell prompt and it is not a part of the command. For example, to issue the below command:**

  $ ls

  Type "ls" (without the quotation marks) into the shell prompt and press [Enter].

1. Log into the BeagleBone. For more information, refer the document labelled ***BeagleBone Tutorial – Introduction***.

2. To ***make a directory***, use the **mkdir** command.

   Issue the command below.

   $ mkdir directory1

   mkdir is a command that creates a directory if it does not already exist.

   For more information, refer to the link below.
   http://man7.org/linux/man-pages/man1/mkdir.1.html

3. To ***change the directory,*** use the **cd** command.

   $ cd directory1

```
root@beaglebone:~# mkdir directory1
root@beaglebone:~# cd directory1
root@beaglebone:~/directory1#
```

Figure 1: Bash session after creating a new directory and navigating to it

   $ cd ..

   One single period "." is equivalent the current path of the current working directory. Two consecutive periods ".." are equivalent to the path to the of the parent of the current working directory.

```
root@beaglebone:~/directory1# cd ..
root@beaglebone:~#
```

Figure 2: Navigating to the parent directory by issuing the command "cd .."

   For more information, refer to the below link.
   http://man7.org/linux/man-pages/man1/cd.1p.html

4. **$ cd directory1/**

5. In the Debian Linux Operating System, there are three standard streams. These streams are: standard input (stdin), standard output (stdout), and standard error (stderr).

   To print a string to standard output, use the **echo** command.

   $ echo hello

```
root@beaglebone:~/directory1# echo hello
hello
root@beaglebone:~/directory1#
```

Figure 3: Bash session after successfully issuing the echo command

   When strings are written to standard output; standard output will display the strings to the text terminal.

   For more information, refer to the link below.
   http://man7.org/linux/man-pages/man1/echo.1.html

6. To redirect standard output to a file, use '>' operator.

   $ echo foo > file1.txt

```
root@beaglebone:~/directory1# echo foo > file1.txt
root@beaglebone:~/directory1#
```

Figure 4: Notice how there is no output on the terminal when redirecting stdout to a file

7. To *con**cat**enate* files and print the result to stdout, use the **cat** command.

   Issue the below command.

   $ cat file1.txt

```
root@beaglebone:~/directory1# cat file1.txt
foo
root@beaglebone:~/directory1#
```

Figure 5: Bash session after successfully issuing the cat command

   For more information, refer to the link below.
   http://man7.org/linux/man-pages/man1/cat.1.html

8. Redirection with **>** performs a **write** operation. Any contents previously stored in the file will be lost.

   $ echo hello > file1.txt
   $ cat file1.txt
   The content has changed from "foo" to "hello".

```
root@beaglebone:~/directory1# cat file1.txt
hello
```

Figure 6: Demonstrating that the '>' operator overwrites the output file

9. The '**>>**' redirects the standard output and append to a file.

   $ echo world >> file1.txt

10. **$ cat file1.txt**
    Now, the content of the file includes both "hello" and "world".

```
root@beaglebone:~/directory1# cat file1.txt
hello
world
root@beaglebone:~/directory1#
```

Figure 7: Bash after successfully issuing the command cat

11. To ***copy*** a file, use the command **cp**.

    The format for this command is as follows: **cp <source_file> <destination_file>**

    $ cp file1.txt file2.txt

    For more information, refer to the link below.
    http://man7.org/linux/man-pages/man1/cp.1.html

12. **$ cat file2.txt**
    Notice that the contents of file2.txt are the same as the contents of file1.txt.

```
root@beaglebone:~/directory1# cat file2.txt
hello
world
root@beaglebone:~/directory1#
```

Figure 8: Bash after successfully issuing the command cat

13. To see a ***list*** of files and directories contained in the current working directory on the Bash, use the **ls** (LS) command. This functionality is similar to file management programs such as **Finder** on Mac or **File Explorer** on Windows.

    $ ls

    Add **–l** to the list of options by entering it as an argument to the **ls** command.

    $ ls –l

    Enabling the **–l**, causes the **ls** command to provide a long listing format. The displayed output includes the permission, the owner, the size, the modified time, and the name of each file or directory.

```
root@beaglebone:~/directory1# ls
file1.txt  file2.txt
root@beaglebone:~/directory1# ls -l
total 8
-rw-r--r-- 1 root root 12 Oct 20 19:23 file1.txt
-rw-r--r-- 1 root root 12 Oct 20 21:47 file2.txt
root@beaglebone:~/directory1#
```

    Figure 9: Bash output after issuing the commands "ls" and "ls -l"

    For more information, refer to the link below.

    http://man7.org/linux/man-pages/man1/ls.1.html

14. Bash provides a **command-line history** feature. To access this, press the up-arrow key to look at previous commands, and press the down arrow key to look for more recent commands. Use the left and right arrow keys to edit the command. Press [Enter] to execute the displayed command.

    Follow the below steps to better understand this tool.

    Press the up-arrow key once. **DO NOT PRESS [Enter]**.
    The command "**ls -l**", is displayed in the shell prompt.

    Press the up-arrow key twice. **DO NOT PRESS [Enter]**.
    The command "**cat file2.txt**" is displayed in the shell prompt.

    Press the down-arrow key once. **DO NOT PRESS [Enter]**.
    The command "**ls**" is displayed in the shell prompt. **Press [Enter]**.

15. **Command-line completion** is a feature available on Bash that completes partially typed commands when a user presses [Tab].

    Type **cat file1** into the shell prompt. **DO NOT PRESS [Enter]**,

    Press [Tab].
    The shell prompt now displays **cat file1.txt**
    Press [Enter].

    Type **cat file** into the prompt. **DO NOT PRESS [Enter]**.
    Press [Tab] twice **without pressing [Enter]**.

    This displays file1.txt and file2.txt.

    Press [Tab] once **without pressing [Enter]**. This will list the all possible completions that start with "file" again. **Press [Enter]**.

16. **$ cd ..**
    $ mkdir directory2

17. To ***move or rename*** a file or a directory, use the **mv** command.

    Issue the commands below.
    $ cd directory1
    $ echo foo > foo.txt
    $ cat foo.txt
    Notice how foo.txt contains "foo".
    $ mv foo.txt bar.txt
    $ cat foo.txt
    Notice how the shell prompt returns an error message.
    This is because the name **foo.txt** is no longer associated with a file in the current working directory.
    $ cat bar.txt
    Notice how the string **foo** is displayed on standard out. This is what was contained in **foo.txt** before.

```
root@beaglebone:~/directory1# echo foo > foo.txt
root@beaglebone:~/directory1# cat foo.txt
foo
root@beaglebone:~/directory1# mv foo.txt bar.txt
root@beaglebone:~/directory1# cat foo.txt
cat: foo.txt: No such file or directory
root@beaglebone:~/directory1# cat bar.txt
foo
```

Figure 10: Bash after issuing each of the above commands

$ cd ..
$ mv directory1/* directory2
Using the **mv** command in this context moves files to the specified destination. * is a wildcard character that represents zero or more characters.

directory1/* expands to mean every file and subdirectory in **directory1/**.

Note: Entering a misconfigured command with a wildcard character can cause extensive damage to an operating system's environment.

$ ls directory1
This command lists the contents of the directory **directory1**. Nothing is listed because the contents of this directory were moved to **directory2**.

$ ls directory2

```
root@beaglebone:~# ls directory1
root@beaglebone:~# ls directory2
file1.txt  file2.txt
root@beaglebone:~#
```

Figure 11: Contents of directory1 and directory2

18. **$ cd directory2**

19. To *delete or **rem**ove* files or directories, use the command **rm**.

Note: Recovering a file or directory deleted by the rm command may be impossible.

$ ls
$ rm file1.txt
$ ls
Notice how file1.txt is no longer present in the list of files.

```
root@beaglebone:~/directory2# rm file1.txt
root@beaglebone:~/directory2# ls
file2.txt
```

Figure 12: Contents of directory2 after deleting file1.txt

To delete a folder, add the option **-r**.

$ mkdir test1
$ ls
$ rm test1

Notice you get an error
$ rm -r test1
$ ls

```
root@beaglebone:~/directory2# mkdir test1
root@beaglebone:~/directory2# ls
file2.txt  test1
root@beaglebone:~/directory2# rm test1
rm: cannot remove 'test1': Is a directory
root@beaglebone:~/directory2# rm -r test1
root@beaglebone:~/directory2# ls
file2.txt
root@beaglebone:~/directory2#
```

Figure 13: Contents of directory2 after deleting test1/

$ cd
$ rm -r directory1
$ rm -r directory2

20. To reboot the BeagleBone, issue the below command.

    $ reboot

    Do not disconnect any cables connected while rebooting.

    After a completing a successful reboot, users will need to enter their login credentials once again to access the Linux Operating System shell program.

21. To ***shut down*** the BeagleBone, issue the below command.

    Do not disconnect any cables until the BeagleBone has successfully completed the shutdown process.
    $ shutdown -h now