Using Linux Basename Command in Bash Scripts

The basename command in Linux prints the final component in a file path. This is particularly helpful in bash scripts where you want to extract the file name from the long file path.

Let me show it to you with some examples.

Basename command

The basename command has two kind of syntax. First one involves a suffix:

basename PATH [suffix]

Second one allows you to add options:

basename OPTION PATH

You cannot combine the options with suffix. Don't be confused just yet. Follow the examples and then you'll understand what I want to say.

Using basename command with a file path will give the file name:

```
basename /home/user/data/filename.txt
filename.txt
```

The basename command is quite stupid actually. It doesn't really recognizes file path. It just looks for the slashes (/) and prints the whatever is after the last slash.

For example, if I run the above example by removing the file name, here's what it will yield.

```
basename /home/user/data
data
```

Remove file extension with suffix

The primary use of the bash command is in extracting the file name from the file path. You can also remove the file extension while extracting the file name.

Just mention what you want to remove from the end of the output. So let's say, you want to remove the .txt from filename.txt. Just add it in the end of the basename command:

```
basename /home/user/data/filename.txt .txt
filename
```

You can also use the -s option for suffix:

```
basename -s .txt /home/user/data/filename.txt
filename
```

The suffix is removed from the end of the final component of the input. If doesn't really figures out the extension of the file. If you provide txt (without the dot) instead of .txt, you'll get 'filename.' (with the dot at the end).

READ Using Linux Sleep Command in Bash Scripts

Also, if you provide a suffix that is not at the end of the component, the output remains as if there was no suffix.

```
basename /home/user/data/filename.txt name
filename.txt
```

Using basename with multiple path

With the option -a, you can use multiple paths at the same time.

```
basename -a /home/user/data/filename1.txt /home/user/data/filename2.txt
filename1.txt
filename2.txt
```

You can use suffix option -s with -a but with some limitations. You can only provide one suffix to all the file paths.

```
basename -as .txt /home/user/data/filename1.txt /home/user/data/filename2.txt
```

filename1
filename2

You cannot assign individual suffices. It won' work.

You can also separate output with NULL instead of the newline with -z option.

Using basename in bash script

I showed some examples of the basename command. Let's see couple of examples of basename in bash scripts.

Suppose you have a file path variable and you want to store the file name from the path in a variable. This could be a simple script:

```
pathname="/home/dir/data/filename"

result=$(basename "$pathname")

echo $result
```

Another example is where you want to rename the file extensions. Of course you can use the rename command to batch rename files but this is just an example.

So I wrote this sample script with the purpose of replacing the file extensions:

```
for file in *$1; do
if [ -f $file ]; then
 mv $file `basename $file .$1`.$2
fi
done
```

Did you notice that i put a check if it is file or not in bash script so that it doesn't change a matching directory?

You can use the above script like this:

```
./myscript.sh html htm
```

And it will rename all the files in current directory with html at the end to htm.

READ 5 Practical Examples of the Read Command in Linux

This was just a few examples. You can use it as per your requirements.

The basename command is complemented with dirname command. Unlike basename, the dirname command prints all the path except the last component.

I hope you liked this tutorial. As always, feel free to ask questions or provide suggestions in the comment section.