Welcome
Linux 2022

www.adafrolabs.com

The process performed by the shell that replaces wildcard-based expressions with matches is called shell expansion or shell globbing.

BASH has the following shell expansion characters:

- * (asterisk) - matches zero or more characters
- ? - matches a single characters
- [ ] (character class) - contains a list of characters inside and it matches any single character out of the list, ranges such as [a-z] or [5-9] are supported
- [!] (exclusion character class) - similar to the above, but it contains characters not to be matched (! at the beginning of the list means "not")

The above characters can be used by themselves, can be combined with regular characters and/or combined with each other to form complex matches.

It is worthwhile to underline again that although we used these characters with commands, commands do not actually use/understand them. The shell intercepts them before the command is actually executed and all that the command sees is a wildcard-free list of files/and or directories processed literally. This essentially means that wildcards can be used with any command that uses files/and directories as its parameters.

Shell expansion characters cannot be used to match the period (.) character used to indicate hidden files. This is meant to protect hidden files from accidental deletion when using wildcards. Any wildcard expressions meant to be operating on hidden files need to start with a literal period, allowing matching of hidden files.

Shell expansion characters can only refer to existing files and/or directories - they cannot be used to create anything. Also, if there are no matches found, shell leaves the wildcard untouched and there is no error.

As mentioned above, wildcards can be used with any command but learning and practicing their use is usually done with the echo command. This command simply displays its parameters which makes it very useful to demonstrate shell expansion.

- Try the echo command in your terminal, for example: echo I love Linux
- Create a new directory in your home called week 11 and use the **cd** command to go into it.
- Now issue the following command: **echo***
- You should just see the asterisk displayed. Normally the asterisk by itself would match any non - hidden files and directories present in your pwd. In this case the week11 directory is empty, so shell did not have any matches to replace this wildcard expression and it was left untouched.
- Try the following command now: **echo ../***
- You should see familiar directory and file names, as you are essentially seeing a listing of your home directory. Keep in mind that echo with a wildcard is not meant as a replacement for the **ls** command, as it lacks the ability to provide detailed information and there is no formatting.
- Figure out a way similar to above to see hidden files and directories in your home directory. Write your answer below.
- Your command: ______________________________

While the *wildcard used by itself has its uses, most often it is used in combination with some other characters, narrowing down the number of matches. You can use such an approach to match file names starting with specific characters, ending with specific characters and/or containing specific characters in the middle of their name. For example:

a* - would match non-hidden file/directory with names starting with letter such as alpha or al .pdf

*.txt - would match non-hidden file/directory with names ending with .txt (extension), such as sample.txt or names.txt

report*9 - would match non-hidden file/directory with names starting with report and ending with 9, for example: report9 or report-sales.2009

.backup[! 3-4] - would match hidden files such as .backupA, .backup7, but not match .backup3 or .backup4

Remember that hidden files are handled separately here from non hidden files. It is a common mistake to use just the asterisk (*) to refer to "all" files, while it only refers to all non-hidden files. Any command line referring to all files must feature * and .* wildcard expression, for example:

```
cp • * * ••
```

As you should anticipate, the above will copy all files (but not directories as the -r option is not used) from your present working directory into your parent directory. The 2 wildcard expressions can be in any order here, but they need to be separated by a space. An expression of *.* would only match non-hidden files with a period in the middle of their name (remember that the hidden-file period is not matched by wildcards).

Make sure you are still in the week11 directory and that directory is empty. Using the **touch** command create the following 10 files in the week11 directory:

labtest labtest1 labtest2 tabtest13 labtext.txt labtxt .txt a ab this.labtest.txt

Additionally, using the **mkdir** directory, create the following 6 directories inside week11:

abc bbc xyz .labtest .b xy.

Try the following commands. Describe in your own words what files are displayed each time.

- echo * ______________________________
- echo .* ______________________________
- echo .* * ______________________________
- echo *.* ______________________________
- echo labtest? ______________________________
- echo ??? ______________________________
- echo ??*______________________________
- echo labtest[12] ______________________________
- echo labtest[12]* ______________________________
- echo [!l] ______________________________
- echo *.txt ______________________________
- echo *txt ______________________________
- echo *t* ______________________________

Based on the knowledge gained above and with help from a classmate, come up with the following commands while in your week11 directory. Each answer must be accomplished using wildcards. Write down your answers in the space provided.

1. Copy all files and directories with names ending with letter t into bbc

   Answer: ______________________________

2. Move the .txt file and .labtest directory into the abc directory

   Answer: ______________________________

3. Display details (ls -l) about non-hidden files)* with file names longer than 5 characters

   Answer: ______________________________

4. Remove write permissions for yourself from labtest1 and labtest1 3 files

   Answer: ______________________________

5. Remove non-hidden files which have the word labtest in their name from your pwd

   Answer: ______________________________

6. Display the tree diagram for abc and bbc directories

   Answer: ______________________________

Quoting

The special meaning associated with alpha-numeric characters in the shell causes problems when such characters are included in file names. In such cases, the shell interprets the special characters causing undesirable and sometimes destructive outcome.

To see the above process, perform the following test

- Create a file called * in your week11 directory using the GUI.
- Open up a Terminal window and use the cd command to go into week1 1 directory
- Try deleting this file using the rm command

What do you think happened? Use the ls command to verify your theory.

In order to prevent the shell from interpreting special characters we use quoting. Most shells support a number of quote characters with the most popular being:

- Single backslash (\) - quotes any single character following it

  For example: **rm *** would allow you to delete the above problem file safely

- Double quotes (") - quote an entire string, presumably containing multiple special characters. The $ characters cannot be quoted that way.

  For example: **touch "My notes.txt"** allows you to create a file with a space in its name.

- Single quotes (') - similar to double quotes with an added ability to quote the $ character

  For example: **echo 'My pwd is $PWD'** will display **My pwd is $PWD**

- Try the above command using double quotes and notice the difference.

Please note that the single quote is "straight down". A single back quote ( ) is completely different and is used for command substitution, which is beyond the scope of this course.

Finding Files

Linux shell features a very useful command called **find**, which is used to find files and directories. It has a wide range of features, allowing users to search by file name, size etc. and optionally perform an action on found files. The find command supports wildcard matches as well, but you must protect them from shell substitution using quoting.

In a simplest case, the find command has the following syntax:

> **find directory-to-start -search-criteria [file-name],** for example:
>
> **find / -name "*.txt"**

The above will search the entire file system for files with an ".txt" extension.

- Try using the **find** command to perform various searches. Use the **man find** command to get help.