

ULI101: INTRODUCTION TO UNIX / LINUX AND THE INTERNET

WEEK 3: LESSON 2

FILENAME EXPANSION QUOTING SPECIAL CHARACTERS

PHOTOS AND ICONS USED IN THIS SLIDE SHOW ARE LICENSED UNDER [CC BY-SA](#)

LESSON 1 TOPICS


File Name Expansion

- Purpose
- Special characters for Filename Expansion: * , ? , [] , [!]
- Demonstration

Quoting Special Characters

- Purpose
- Backslash \ , Single Quotes ` ` , Double Quotes ` "`
- Demonstration

Perform Week 3 Tutorial

- INVESTIGATIONS 2 and 3
 - LINUX PRACTICE QUESTIONS (Questions 9 – 13)
- 

FILENAME EXPANSION

Filename Expansion

This command displayed below is **inefficient**: it requires a LOT of typing and requires that the user know all the filenames within the current directory.

```
ls a.txt b.txt c.txt 1.txt 2.txt 3.txt abc.txt work.txt  
a.txt b.txt c.txt 1.txt 2.txt 3.txt abc.txt work.txt
```

Filename expansion is the use of **special characters** to allow the shell to **match** files that share the **same characteristics** to help save the user save time when issuing Unix / Linux file management commands.

You can use a special character to indicate to the Bash shell to match all files that end with the extension ".txt":

```
ls *.txt  
a.txt b.txt c.txt 1.txt 2.txt 3.txt abc.txt
```

FILENAME EXPANSION

Common File Expansion Symbols

Filename Expansion Symbol	Purpose
*	Asterisk (*) to represent 0 or more characters
?	Question mark (?) to represent exactly one character (any character)
[]	Square brackets ([]) to represent and match for the character enclosed within the square brackets . It represents ONLY ONE character: it's like a Question Mark (?) but with conditions or restrictions
[!]	Square brackets containing an exclamation mark immediately after the open square bracket ([!]) to represent and match and OPPOSITE character for the character enclosed within the square brackets.

FILENAME EXPANSION

How Does File Expansion Work? (Process of “Globbing”)

File Globbing is a feature provided by the UNIX/Linux shell to represent multiple filenames by using special characters called wildcards with a single file name. A wildcard is essentially a symbol which may be used to substitute for one or more characters. Therefore, we can use wildcards for generating the appropriate combination of file names as per our requirement.

Reference: <https://www.linuxnix.com/10-file-globbing-examples-linux-unix/>

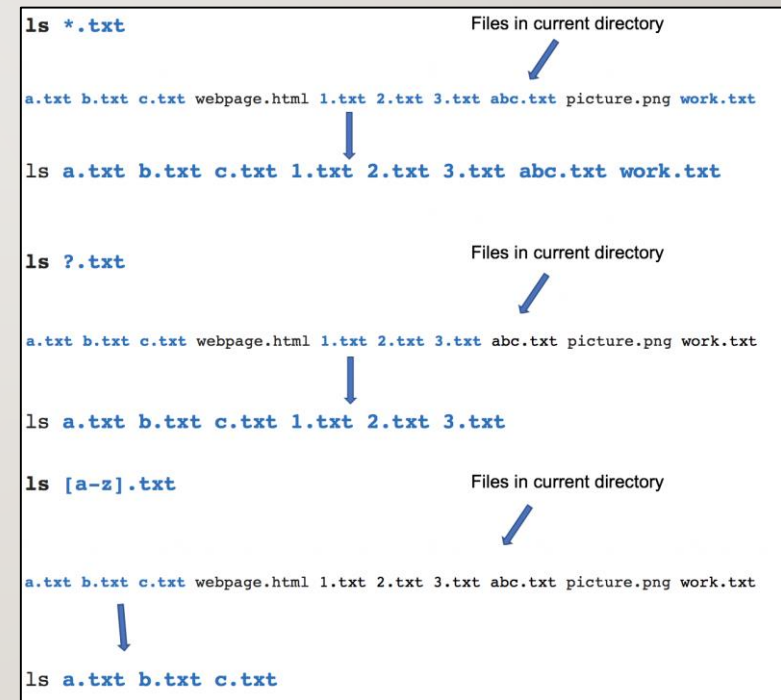
FILENAME EXPANSION

How Does this Work? (Globbing)

As shown in the diagram on the right, when the **ls** command is issued with a filename expansion symbol (like *****), the Bash shell **searches** for all files in the current directory that match files that end with the extension **".txt"**.

The shell replaces ***.txt** with all the files that end with the extension **.txt** in the current directory and runs that command.

You do not see that happen in the shell...
it is a process that occurs "behind the scenes".
Instead, you only see the results of the command.



FILENAME EXPANSION



Instructor Demonstration

Your instructor will now demonstrate how to issue Unix / Linux commands using various **filename expansion symbols** for file management:

- Creating / Removing Directories
- Moving Files / Directories
- Copying Files / Directories
- Listing Directory Contents
- Removing Regular Files

COMMAND HISTORY

Command History:

The `~/.bash_history` file stores recently executed command lines.

There are several techniques using the `~/.bash_history` file to run previously-issued commands.

Examples:

`<up>` or `<down>` move to **previous** or **next** command in Bash shell prompt

`fc -l` display last **16** commands

`history | more` display all stored commands

`!#` **re-executes** command by command number (obtained from *history* command)

`!abc` **re-executes** last command beginning with string "abc"

QUOTING SPECIAL CHARACTERS

“

”

Quoting Special Characters

As discussed in the above section, there are some special characters that the shell uses to perform an operation; for example, the filename expansion symbols: *, ?, [] or [!]

There are methods to instruct the Linux shell to ignore that special character and use only as **regular text**.

These **3 methods** to make those special characters **act like text characters** (referred to "**quoting**" special characters).

These methods are displayed in the next slide.



QUOTING SPECIAL CHARACTERS

Quoting Special Characters (Methods)

The most common filename expansion symbols are displayed below:

Quoting Method	Example
Place the character <code>\</code> <u>before</u> a special character (works for ALL special characters)	<code>echo *</code>
Contain Special character within single quotes <code>' '</code> (work for ALL special characters)	<code>echo '* hello *'</code>
Contain special characters within double-quotes <code>" "</code> NOTE: Double quotes works for most special characters, but not all special characters (such as \$variable-name - variables are discussed <u>later</u> in this course)	<code>echo "* hello *"</code>

QUOTING SPECIAL CHARACTERS



Instructor Demonstration

Your instructor will now demonstrate how to issue Unix / Linux commands **quoting special characters**, their **uses** and their **consequences**:

- Displaying Text
- Creating / Removing Directories
- Listing Directory Contents
- Removing Regular Files

HOMEWORK

Getting Practice

Perform the online tutorial **Tutorial 3: Unix / Linux File Management**
(Due: Friday Week 4 @ midnight for a 2% grade):

- [INVESTIGATION 2: FILENAME EXPANSION](#)
- [INVESTIGATION 3: QUOTING SPECIAL CHARACTERS](#)
- [LINUX PRACTICE QUESTIONS](#) (Questions 9 – 13)