Log in Request account

Page Discussion Read View source View history Search CDOT Wiki Go

CDOT
SICT AR Meeting Area
People

get involved with CDOT

as a Student
as an Open Source Community Member
as a Company

courses

BTC640
BTH740
BTP300
DPI908
DPS901
DPS905
DPS909
DPS911
DPS914
DPS915
DPS924
DPS931
EAC234
ECL500
GAM531
GAM666
GAM670
GPU610
LUX Program
MAP524
OOP344
OPS235
OPS245
OPS335
OPS345
OPS435
OPS445
OPS535
OPS635
OSD600
OSD700
OSL640
OSL740
OSL840
Real World Mozilla
RHT524
SBR600
SEC520
SPO600
SRT210
ULI101

Tutorial3: Advanced File Management / Quoting Special Characters

Contents [hide]

ADVANCED UNIX / LINUX FILE MANAGEMENT

Main Objectives of this Practice Tutorial

- Understand the difference between **absolute** , **relative** and **relative-to-home** pathnames
- Become productive at issuing Linux commands with the most appropriate pathname
- Use **Filename Expansion** (**FNE**) Symbols: ***** , **?** , **[ ]** , **[!]**
- Use **quotation** treat special characters as just **text** when issuing Linux commands.
- Understand the quotation symbols: **Backslash \** , **single quotes ' '** and **double quotes " "**

Tutorial Reference Material

Course Notes	Pathname Type / Filename Expansion / Quoting		YouTube Videos
Slides: • Week 3 Lecture 1 Notes: PDF | PPTX • Week 3 Lecture 2 Notes: PDF | PPTX	**Pathname / Filename Expansion:** • Absolute • Relative • Relative-to-home • Filename Expansion Symbols	**Quoting Special Characters:** • Backslash \ , Single **' '** , Double **" "** **Linux Commands:** • echo	**Instructional Videos:** • Pathname Types • Filename Expansion Examples • Relative, Absolute, and Relative-to-Home Filepaths • Quoting Special Characters

course projects

Course Project List
Potential Course Projects
Contrib Opportunities

links

CDOT
Planet CDOT
FSOSS

Tools

What links here
Related changes
Special pages
Printable version
Permanent link
Page information

KEY CONCEPTS

Pathname Types

As previously mentioned, a **pathname is a fully-specified location of a unique filename** within a file system. The concept of a pathname relates to every operating system including: *Unix, Linux, MS-DOS, MS-Windows, Apple-Macintosh,* etc.

Last week, we used a pathname from our home directory to create and manipulate directories and text files. There are **different types of file pathnames** that we can use to access a directory or text file.

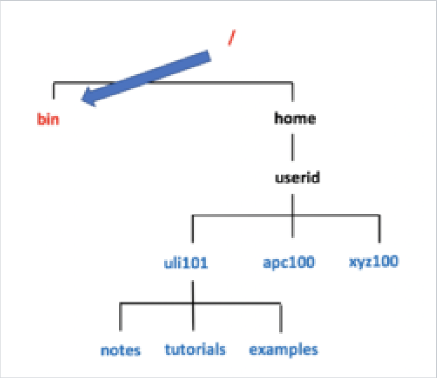
For Example:
`/home/userid/uli101/cars.txt` (**absolute pathname**)
`samples/cars.txt` (**relative pathname**)
`~/cars.txt` (**relative-to-home pathname**)

These types of file pathnames can make it more efficient (i.e. less keystrokes for users to type) when issuing Unix and Linux commands.

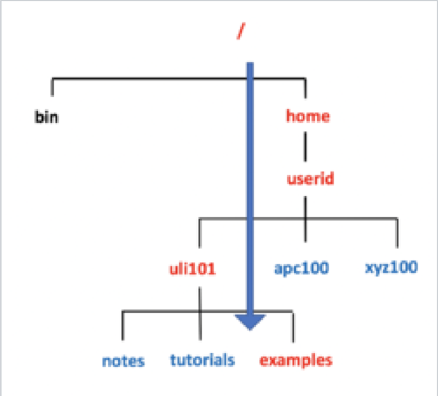
Absolute Pathnames

An **absolute pathname** is a path to a file or directory always **beginning from the root directory (i.e. /)**.

This type of pathname is referred to as **absolute** because the pathname always begins from the **root directory**, regardless the location or your current directory. In other words, this type of pathname requires that you always provide the **FULL** pathname starting with the root directory.



The directory in red displays the FULL path from the **root** directory to the **bin** directory (i.e. the absolute pathname: **/bin**.



Directories in red display the FULL path from the **root** directory to the **examples** directory (i.e. the absolute pathname: **/home/your-user-id/uli101/examples**.

Advantages of using Absolute Pathnames:

- Useful if you do not know your current directory location
- Understand the location of file within the filesystem.

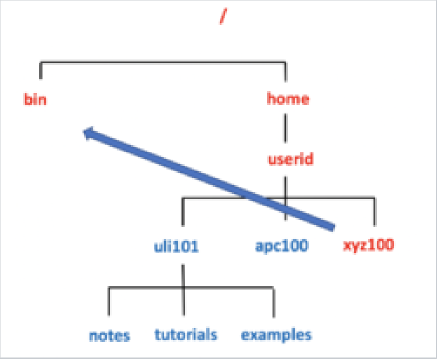
Examples:

```
ls /bin
ls /home/your-user-id/uli101/examples
```

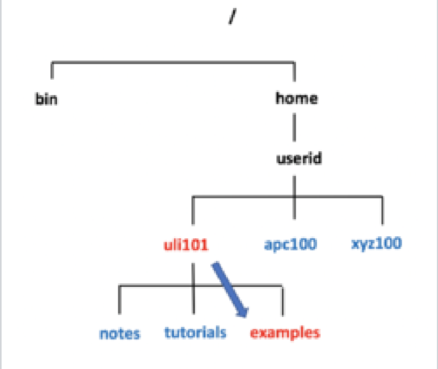
Relative Pathnames

A **relative pathname** is a path to a file or directory that begins from your

current directory. This is called relative because it is used to locate a specific file relative to your current directory.



Directories in red display the path from the current directory location (which is **xyz100**) to the **bin** directory (i.e. the relative pathname: **../../../bin**.



Directories in red display the path from the current directory location (which is **uli101**) to the **examples** directory (i.e. the relative pathname: **uli101/examples** or ./uli101/examples

NOTE: In order to use relative pathnames, it is absolutely necessary that you know the **location** of your **current directory**!

relative pathname : relative to your current directory

Relative Pathname Symbols:

. A period symbol "." represents the **current** directory

.. Two consecutive period symbols ".." represents the **parent** directory (i.e. one level up)

Advantages of using Relative Pathnames:

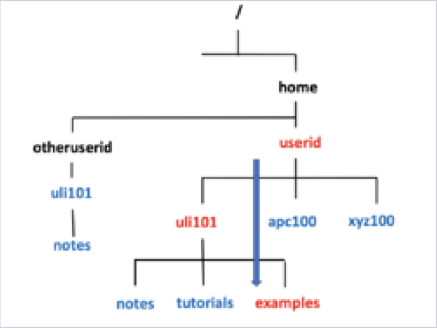
- Possible shorter pathname (less typing)

Examples:

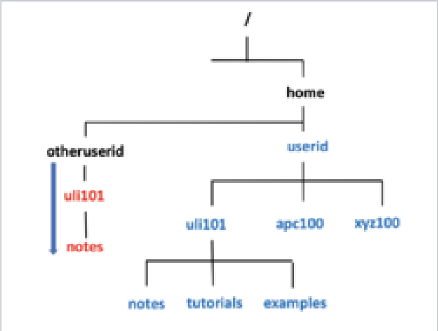
```
ls ../../../bin
ls examples
ls ./examples
```

Relative-to-home Pathnames

A **relative-to-home pathname** begins with the tilde character (i.e. ~) to represent the user's home directory.



Directories in red display the path from the home directory of the current user (which is **userid**) to the **examples** directory (i.e. the relative-to-home pathname: **~/uli101/examples**.



Directories in red display the path from another user's home directory location to their **notes** directory (i.e. the relative-to-home pathname: **~jane/uli101/notes**

Not ~/jane

The tilde character **~** stores the path of the user's home directory (i.e. **~ = /home/current-user-id**)

You can immediately place a username after the tilde to represent another user's home directory (e.g. **`~jane = /home/jane`**)

Examples:

```
ls ~/uli101/examples
ls ~murray.saul/uli101/notes
```

NOTE: Deciding which **type of pathname** to use depends on many factors including: **knowledge of current directory**, **knowledge of directory structure**,
currently directory location, and **type of file management command** that is being used.

Filename Expansion

When issuing Linux commands, it may be **more efficient** (less typing) to use **filename expansion symbols** to match files that share similar characteristics (e.g. same file extension) when issuing Linux commands.

Examples:

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 work.txt
```

Below are the most common Filename Expansion symbols and how they are used for filename expansion:

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.

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 **?**

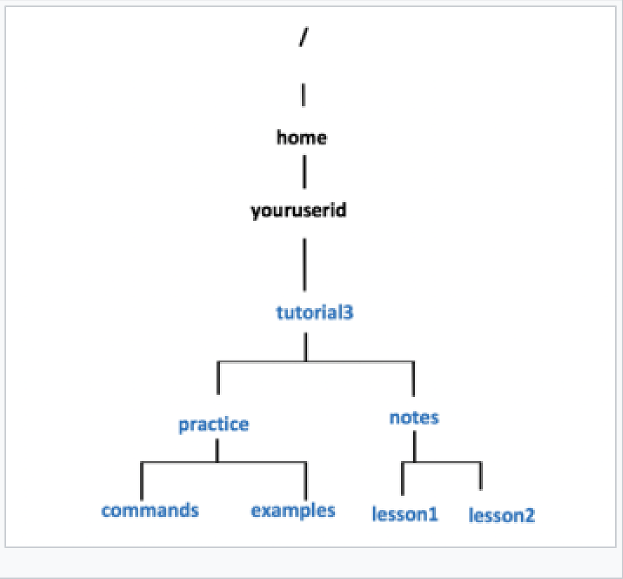
There are **3 methods** to make those special characters **act only like text characters** when issuing Linux commands (displayed in chart below):

Quoting Method	**Example**
Place the character \ before a special character	`echo \*`
Contain special characters within **double-quotes** **NOTE:** Double quotes works for most special characters, but not all special characters (such as $)	`echo "* hello *"`
Contain Special character within single **quotes** (Quotes out ALL special characters)	`echo '* hello *'`

INVESTIGATION 1: ABSOLUTE / RELATIVE / RELATIVE-TO-HOME PATHNAMES

ATTENTION: This online tutorial will be required to be completed by **Friday in week 4 by midnight** to obtain a grade of **2%** towards this course

The best way to learn about different types of file pathnames is to use them while issuing Linux commands and see which pathnames (or combination of pathnames) is the **most efficient** (i.e. requiring the **least number of keystrokes**).



Perform the Following Steps:

1. **Login** to your matrix account.

2. Issue a command to **confirm** you are located in your home directory.

   Let's create the following directory structure under your home directory by issuing the mkdir command using only absolute pathnames.

3. Issue the following Linux command to create the directory structure displayed to the right using **absolute pathnames**:

   NOTE: Just continue typing and let the text continue of separate lines. Remeber to replace the text "youruserid" with your actual Seneca-id.

   ```
   mkdir -p /home/youruserid/tutorial3/practice/commands
   /home/youruserid/tutorial3/practice/examples
   /home/youruserid/tutorial3/notes/lesson1
   /home/youruserid/tutorial3/notes/lesson2
   ```

   absolute pathnames

4. Issue the following Linux command to confirm that you properly created the directory structure:

   ```
   tree /home/youruserid/tutorial3
   ```

   You should notice that using absolute pathnames with this Linux command **requires a lot of typing**. Let's **remove** this directory structure, and issue the same command using a *relative-to-home* pathname instead.

5. To remove this directory structure, issue the following Linux command (enter "**y**" at each prompt to remove ALL contents):

   ```
   rm -ri /home/youruserid/tutorial3
   ```

6. Issue a command tree command as you did in **step #4** to confirm that the directory structure has been removed.

7. Issue the following Linux command to create the same directory structure using relative-to-home pathnames:

   NOTE: You usually generate the ~ character by Holding down **SHIFT** and press the button to the left of the number **1** above the text on your keyboard.

```
mkdir -p ~/tutorial3/practice/commands ~/tutorial3/practice/examples
~/tutorial3/notes/lesson1 ~/tutorial3/notes/lesson2
```

relative to home

Did this command require less typing than the previous command using absolute pathnames? Yes

8. Issue the **tree** command to confirm the directory structure was properly created.

   Let's remove the **tutorial3** directory and its contents and issue the same command using **relative pathnames**.

9. Issue the same command as you did in **step #5** to remove the **tutorial3** directory and its contents safely.

10. Issue a Linux command to confirm you removed the **tutorial3** directory and its contents.

11. Issue the following Linux command to create the same directory structure using relative pathnames:

   ```
   mkdir -p tutorial3/practice/commands tutorial3/practice/examples
   tutorial3/notes/lesson1 tutorial3/notes/lesson2
   ```

   relative pathnames

   because current directory is home/twwong9, can skip the heading

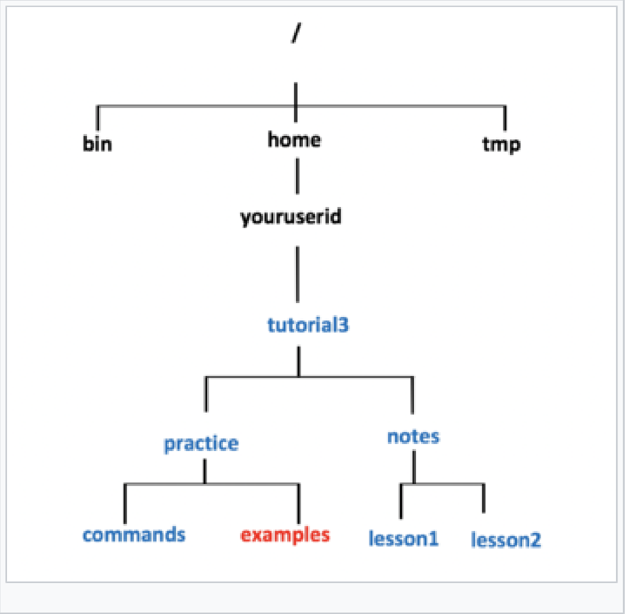
12. Issue a command to verify that the proper directory structure was created.

   QUESTION: Which **mkdir** command (pathname type) that you performed in steps **3** , **7** , and **11** required the LEAST number of keystrokes (i.e. characters)? relative pathnames

You may think that issuing Linux file management commands are better using **relative** or **relative-to-home** pathnames instead of **absolute** pathnames, but that is not always true.

Since the **current** directory location was your **home** directory, then it makes sense to use *relative* or *relative-to-home* pathnames. On the other hand, what if we changed the location to a different directory?

When performing the next series of steps, refer to the **tree diagram** on the right. Learning to reference a tree diagram on a **quiz**, **midterm** or **final exam** can help to **prevent errors and loss of marks!**



Perform the Following Steps:

1. Make certain that your current directory is **your home directory**.

2. Since we will be running Linux commands depending on the directory structure that you have created, issue the following Linux command to verify you created it correctly:
   `~uli101/week3-check-1`

3. If you encounter errors, make corrections and then re-run the checking script until you receive a

congratulations message.

4. Issue a Linux command to change to the **examples** directory in your recently-created directory structure.
   `cd tutorial3/practice/examples`

5. Issue a Linux command to confirm you are located in the *examples* directory.

6. Remembering that we are located in the **examples** directory, issue the following Linux command using a **relative** pathname to display files in the /bin directory: `ls ../../../../../bin`

7. Now issue the following Linux command using an **absolute** pathname: `ls /bin`

   Which type of pathname would be the best to use in this situation? absolute pathname

   NOTE: Using the previous command using the **relative-to-home** pathname would work,
   but it would look weird. Try to issue the command yourself! ls ~/../../bin

8. Let's copy the file called ls which is contained in the **/bin** directory to your **home** directory by using the **cp** command.
   First, issue the following Linux command to copy the ls command from the /bin directory to your home directory using absolute pathnames:
   `cp /bin/ls /home/youruserid`

9. Now let's issue the previous command using just relative pathname (remember, our current directory location is **examples**):
   `cp ../../../../../bin/ls ../..` under tutorial3
   TIP: For relative pathnames that move up multiple parent directories such as these, it is HIGHLY RECOMMENDED
   to view the tree diagram and check for the correct number of .. symbols. Students commonly make mistakes
   and lose marks on these type of questions!

10. Let's issue the command using one absolute pathname and a relative pathname: `cp /bin/ls ../..`
    What did this command do?

11. Let's issue the same command using one absolute pathname and a relative-to-home pathname: `cp /bin/ls ~`
    What did this command do?
    Which of the following file type combinations requires the LEAST number of keystrokes to copy the ls file to your home directory?

12. Let's copy the **ls** file from the **/bin** directory to your current directory (i.e. examples): `cp /bin/ls .`

13. Issue the following Linux command: `cp /bin/ls ./ls.bk`

    What does this command do? duplicate and rename the file with ls.bk under examples directory

    Let's run a checking script to make certain you performed the a few of the recently-issued commands correctly.

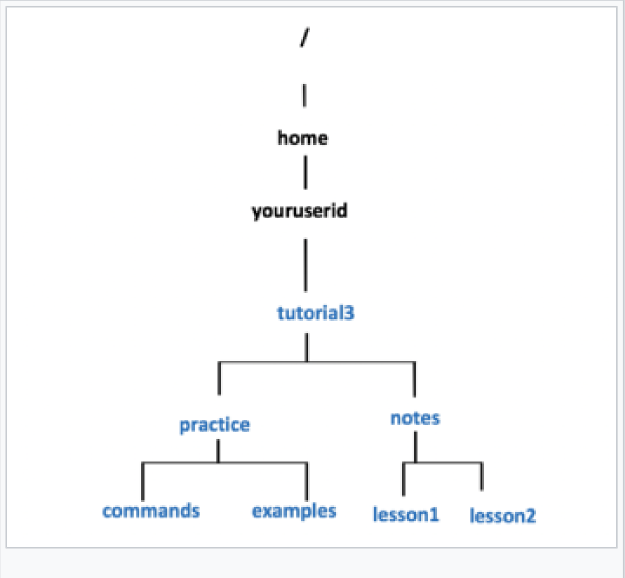
14. Issue the following: `~uli101/week3-check-2`

15. If you encounter errors, make corrections and then re-run the checking script until you receive a congratulations message,
and proceed to the next INVESTIGATION.

INVESTIGATION 2: FILENAME EXPANSION

You will now get practice issuing Linux file management commands using **filename expansion symbols**.
We will be using the directory structure that was created in the previous INVESTIGATION.

A great way to practice filename expansion, use the **touch** command to create a lot of empty filenames, write the **ls** Linux commands that use **filename expansion**, predict the filenames that will be display, and finally run the command to check your work.



Perform the following steps:

1. Issue a Linux command to move to the **examples** directory
(i.e. under *practice* directory as shown in diagram to the right).

2. Issue a Linux command to confirmed that you have moved to the **examples** directory.

3. Issue the **touch** command to create the following empty text files in the *examples* directory:
(note *upper* and *lowercase* letters)

```
abc.txt
def.text
hij.TxT
1a4.txt
123.TXT
456.txt
6u9.txt
ab2.html
1234.txt
abcdef.txt
abcde.txt
```

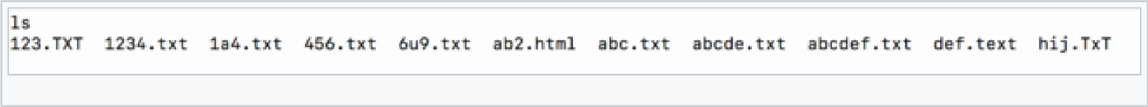
4. To verify that you properly created those files, issue the following:
`~uli101/week3-check-3`

If you encounter errors, then make corrections (eg. **viewing directory contents**, **check for correct filename syntax**,
case sensitivity, **missing files**, **files in the wrong location**, etc.) and then re-run the checking script until you receive a congratulations message, and then continue with this investigation.

ATTENTION:
Learning to **fix your mistakes** by issuing Linux commands may be required if you make mistakes in your online tutorial.

5. Issue the **ls** command to get a listing of files in your *examples* directory.

   The output should look identical to the diagram displayed below.
   You can refer to this listing to see all files so you can then predict the output from Linux commands that use filename expansion symbols.



```
ls
123.TXT  1234.txt  1a4.txt  456.txt  6u9.txt  ab2.html  abc.txt  abcde.txt  abcdef.txt  def.text  hij.TxT
```

6. What do you think the output will be from the following Linux command?
   `ls ???.txt` 1a4.txt 456.txt 6u9.txt abc.txt
   Write down the expected output on paper, then **issue the command** to check your answer.

7. What do you think the output will be from the following Linux command?
   `ls ?????.txt` abcde.txt
   Write down the expected output on paper, then **issue the command** to check your answer.

8. What do you think the output will be from the following Linux command?
   `ls ??????.txt` abcdef.txt
   Write down the expected output on paper, then **issue the command** to check your answer.

9. What do you think the output will be from the following Linux command?
   `ls [0-9].txt` No such file or directory
   Write down the expected output on paper, then **issue the command** to check your answer.br>Did the command work?
   What does this teach you about the character class [ ] symbol?

10. What do you think the output will be from the following Linux command?
    `ls [0-9][0-9][0-9].txt` 456.txt
    Write down the expected output on paper, then **issue the command** to check your answer.

11. What do you think the output will be from the following Linux command?
    `ls [a-z][a-z][a-z].txt` abc.txt
    Write down the expected output on paper, then **issue the command** to check your answer.

12. What do you think the output will be from the following Linux command (using character class with UPPERCASE letters)?:
    `ls [A-Z][A-Z][A-Z].txt` abc.txt
    Write down the expected output on paper, then **issue the command** to check your answer.

13. What do you think the output will be from the following Linux command (using character class using alpha-numeric characters)?
    `ls [a-zA-Z0-9][a-zA-Z0-9][a-zA-Z0-9].txt` 1a4.txt 456.txt 6u9.txt abc.txt
    Write down the expected output on paper, then **issue the command** to check your answer.

14. What do you think the output will be from the following Linux command?
    `ls *.txt`
    Write down the expected output on paper, then **issue the command** to check your answer. Did ALL text files get listed? Why not? Not all files get listed since not all files are with file extension (.txt)
    1234.txt 1a4.txt 456.txt 6u9.txt abcdef.txt abcde.txt abc.txt

15. What do you think the output will be from the following Linux command?
    `ls *.[tT][xX][tT]`
    Write down the expected output on paper, then **issue the command** to check your answer. Did ALL

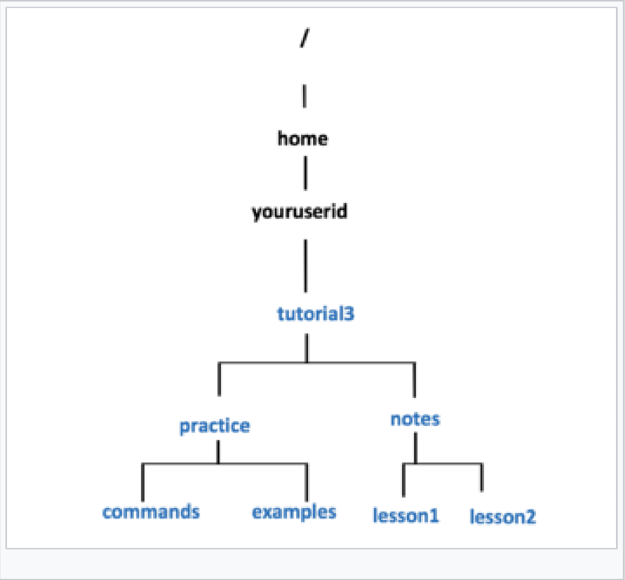
text files get listed this time? If so, why? Not all files get listed since not all files are with file extension (.txt / .TXT) 1234.txt 1a4.txt 6u9.txt abcde.txt hij.TxT 123.TXT 456.txt abcdef.txt abc.txt

16. **NOTE:** We have just been using filename expansion symbols just with the ls command. Filename expansion symbols can be used for ANY Linux file management command (e.g. **cat**, **more**, **less**, **cp**, **mv**, **rm**, **ls**, etc.).

    Let's get some practice issuing these other Linux file management commands.

17. Issue the following Linux command: `file *.[tT][xX][tT]`
    What is the purpose of this command? Which files are contained in this output?

Gives info about the contents of the file, 1234.txt 123.TXT 1a4.txt 456.txt 6u9.txt abcdef.txt abcde.txt abc.txt hij.TxT



18. Change to the **commands** directory using an **absolute** pathname (use the diagram on right-side for reference).

cd /home/twwong9/tutorial3/practice/commands

19. Issue a Linux command to confirm that you are now in the **commands** directory.

20. Issue the following Linux command (lowercase "l" NOT the number "1"):
    `cp /bin/l* .`

21. View the contents of the contents directory. What did this command do?
    copy all files start with l to current directory

22. Issue the following Linux command: `rm *`

    View the contents of the contents directory.
    What did this command do? remove all the files inside current directory

22. Issue the following Linux command (lowercase "l" NOT the number "1"):
    `cp /bin/l? .` copy all files start with l (ld ln ls) to current directory
    View the contents of the contents directory. What did this command do?

delete all other files start with l and end with not s

23. Issue the following Linux command: `rm l[!s]`
    View the contents of the contents directory. What did this command do?

24. Use a text editor (nano or vi) to create the file called **ab** in the **commands** directory that contains the line of text below,
    and then save editing changes to this file:

    ```
    This is file ab
    ```

25. Use a text editor (nano or vi) to create the file called **cd** in the **commands** directory that contains the line of text below,
    and then save editing changes to this file:

    ```
    This is file cd
    ```

26. Use a text editor (nano or vi) to create the file called **ef** in the **commands** directory that contains the line of text below,
    and then save editing changes to this file:

    ```
    This is file ef
    ```

27. Issue the following Linux command: `cat ??`

View the contents of the contents directory. What did this command do? Why does the output look strange? Because cat ?? will also output the content inside ls, which make the output look strange

NOTE: Press the keys **ctrl-c** to return to the shell prompt.

28. Issue the following Linux command: `cat [!l][!s]`

    View the contents of the contents directory. What did this command do? Does the output look better? If so, why?
    only output the content inside the text file (ab cd ef)

Proceed to the next INVESTIGATION.

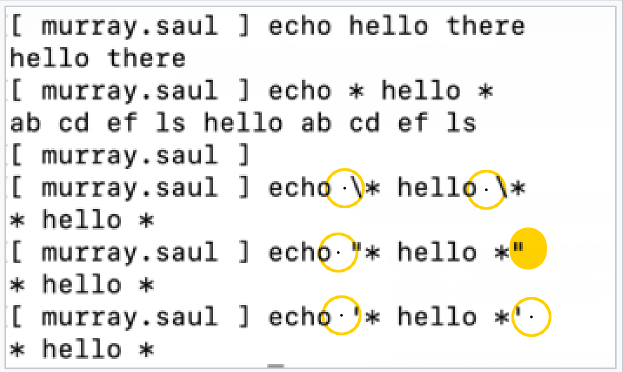
INVESTIGATION 3: QUOTING SPECIAL CHARACTERS

As discussed in the above investigation, there are some special characters that the shell uses to perform an operation
including the filename expansion symbol: *

There is a method make the shell **ignore the purpose of special characters** and treat as **regular text**.

In this investigation, you will learn **three unique methods** to quote special characters.

Perform the Following Steps:



```
[ murray.saul ] echo hello there
hello there
[ murray.saul ] echo * hello *
ab cd ef ls hello ab cd ef ls
[ murray.saul ]
[ murray.saul ] echo \* hello \*
* hello *
[ murray.saul ] echo "* hello *"
* hello *
[ murray.saul ] echo '* hello *'
* hello *
```

Using quotation to make the shell **ignore the purpose of special characters** and treat as **regular text**.

1. Issue a Linux command to confirmed that you are still in the **commands** directory (if not, change to the *commands* directory and confirm).
2. Issue the following Linux command: `echo hello there`

   NOTE: the **echo** command is used to display text onto your terminal.
3. Issue the following Linux command: `echo * hello *` ab cd ef hello ab cd ef

   What happened? What is shown in addition to the text "hello".
   Why do you think those filenames are also being displayed?
4. Issue the following Linux command: `echo \* hello \*` * hello *

   What do you notice? What does the \ character do?
5. Issue the following Linux command: `echo "* hello *"` * hello *

   Is there a difference between this command and the previous command? no difference
6. Issue the following Linux command: `echo '* hello *'` * hello *

   Is there any difference between this command and the two previous commands? no difference
7. Issue the following Linux command: `echo $USER` twwong9

NOTE: The environment variable **USER** contains the current user's login name.
The **$** character immediately followed by the environment variable name causes it to expand to the **value** that the variable contains.

8. Issue the following Linux command (using single quotes): `echo '* $USER *'`

   What happened? Why is the output display like this? * $USER *

9. Issue the following Linux command (using double quotes): `echo "* $USER *"`

   What happened? Why? * twwong9 *

10. Let's use quoting special characters for a couple of other commands. Move to the **lesson2** directory. Confirm that you are currently located in the *lesson2* directory.

    NOTE: Although it is NOT RECOMMENDED to create a filename using a special character (remember file naming rules?),
    we will create an empty file called "*"

11. Issue the following Linux command (using single quotes): `touch '*'`

12. Issue the **ls** command. Do you now see a file called "*" in addition to the other copied files?

13. Issue the following Linux command to remove the file called "*": `rm *`

14. Issue the **ls** command to view the files in your current directory. What happened?!?
    Why is it dangerous to use special characters when creating filenames? rm * will not only delete file called "*", but also delete all the files

15. Issue the following Linux command: `cp /bin/ls .`

16. Issue the following Linux command (using single quotes): `touch '*'`

17. Issue the **ls** command to view the files in your current directory.

18. Issue the following Linux command (using single quotes): `rm '*'`

19. Issue the **ls** command to confirm that the file called "*" has been removed in your current directory.
    What happened this time? Yes. The file called "*" has been removed and file called "ls" keep left

20. Complete the Review Questions sections to get additional practice.

LINUX PRACTICE QUESTIONS

The purpose of this section is to obtain extra practice to help with your quizzes, your midterm, and your final ezam.

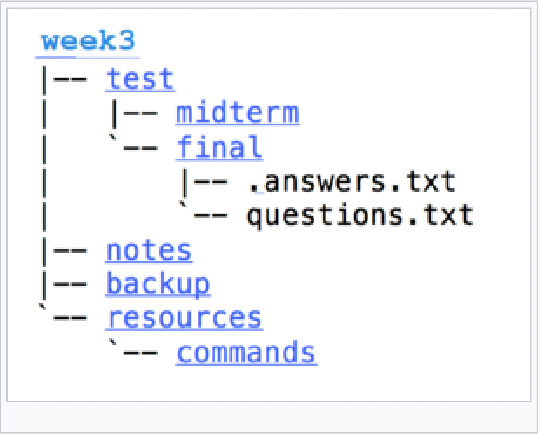
Here is a link to the MS Word Document of ALL of the questions displayed below but with extra room to answer on the document to simulate a quiz:

https://github.com/ULI101/labs/raw/main/uli101_week3_practice.docx

Your instructor may take-up these questions during class. It is up to the student to attend classes in order to obtain the answers to the following questions. Your instructor will NOT provide these answers in any other form (eg. e-mail, etc).

Review Questions:

When answering Linux command questions on this side or the back side of this page, refer to the following Inverted Tree diagram. The week3 directory is contained in your home directory. Assume that you just logged into your Matrix account. Directories are underlined.



```
week3
|-- test
|    |-- midterm
|    `-- final
|         |-- .answers.txt
|         `-- questions.txt
|-- notes
|-- backup
`-- resources
     `-- commands
```

1. Write a single Linux command using relative pathnames to create the directory structure displayed in the diagram above.
2. Write a single Linux command to create the empty files “.answers.txt” and “questions.txt” shown in the diagram above using absolute pathnames.
3. Write a Linux command to display a listing of all hidden and non-hidden filenames in the directory called “final” using a relative-to-home pathname.
4. Assuming you are in your home directory, write a Linux command to view the contents of the “.answers.txt” file using a relative pathname. You can assume this text file is very large and you want to see all of the contents.
5. Write a Linux command to change to the “backup “directory using an absolute pathname.
   Write a command to verify that you changed to that directory:

6. Assuming that you remain in the “backup” directory, write a Linux command to copy the “questions.txt file” to your current directory. You are required to only use relative pathnames.
7. Assuming that you remain in the “backup” directory, write a Linux command to delete the “questions.txt” file that is in your “backup” directory. Use a relative-to-home pathname.
8. Assuming that you are currently located in your “backup” directory, write a Linux command to safely remove the directory “week3” and all of its contents. Use an absolute pathname.
   Will your command you wrote in question 8 work if you run it? (yes/no). Why?

9. Assuming you are still located in the “backup” directory. Write a Linux command using a relative-to-home pathname to remove all files that end with the extension “.txt” in the “final” directory.
10. Write a Linux command using an absolute pathname to list all files that consist of just 4 consecutive characters that are contained in your home directory.
11. Write a Linux command using a relative pathname to list all files that begin and end with a number.
12. Write a Linux command using a relative-to-home pathname to list all files that begin with a number but ends with any character other than a number.
13. Assuming you are in your home directory. Write a Linux command using a relative pathname to view the contents of regular files whose file names only consist of 5 consecutive numbers.

14. Write a Linux command to display the following message:

*** Hello ***

15. Write a Linux command to display the following message (including quotation marks):
“This is my message”

Author: Murray Saul

Category: ULI101

This page was last edited on 29 August 2022, at 18:07.