OSL640: INTRODUCTION TO OPEN SOURCE SYSTEMS

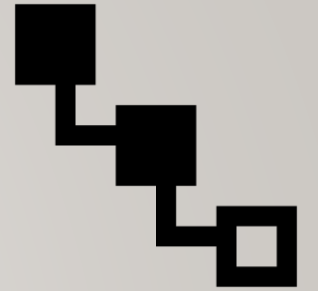## WEEK 3: LESSON 1

ADVANCED FILE MANAGEMENT

LESSON 1 TOPICS

File Pathname Types

- Absolute File Pathnames

- Relative File Pathnames

- Relative-to-home File Pathnames

- Demonstration

Perform Week 3 Tutorial

- Investigation 1

- Review Questions (Questions 1 – 8)

FILE PATHNAME TYPES

Purpose of File Pathnames

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 pathnames** that we can use to access a directory or text file.

For Example:

/home/userid/osl640/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.
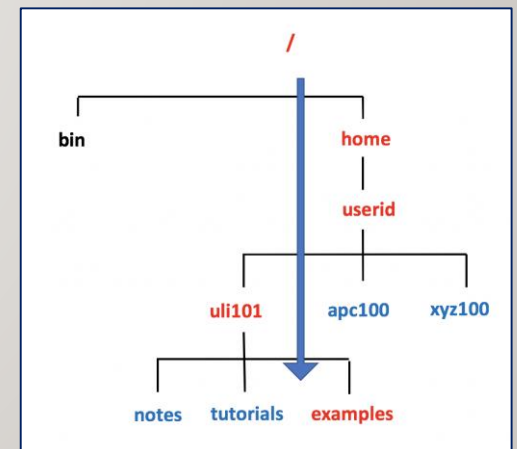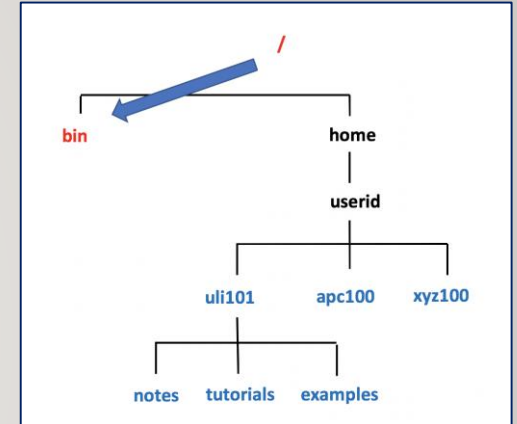
FILE PATHNAME TYPES

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 ABSOLUTELY from the **root directory** regardless of your current directory location.

In other words, this type of pathname requires that you always provide the **FULL** pathname starting with the root directory.

Remember the Rhyme: *"If it is ABSOLUTE, it begins with ROOT!"*

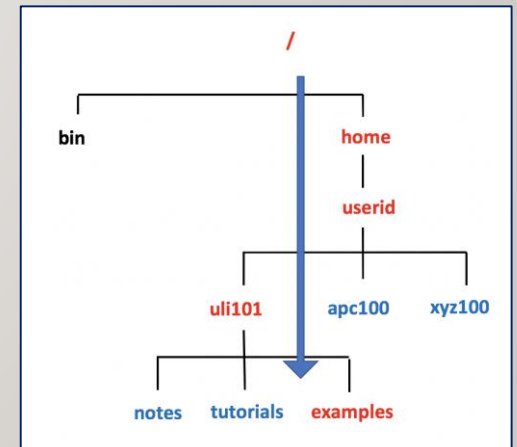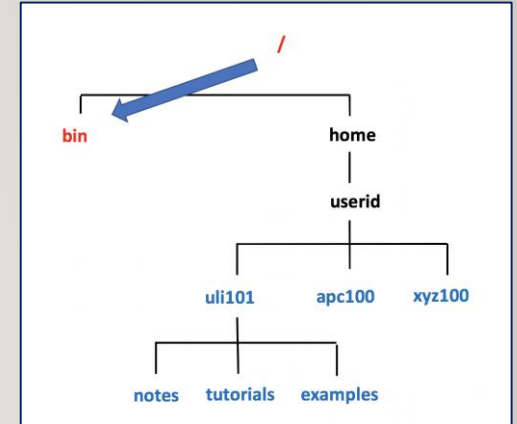# FILE PATHNAME TYPES

Absolute Pathnames

Advantages of using Absolute Pathnames:

- Useful if you do NOT know your current directory location
- Helps you to understand the FULL layout of pathname

Examples:

`/bin`
`/home/userid/uli101/examples`

FILE PATHNAME TYPES
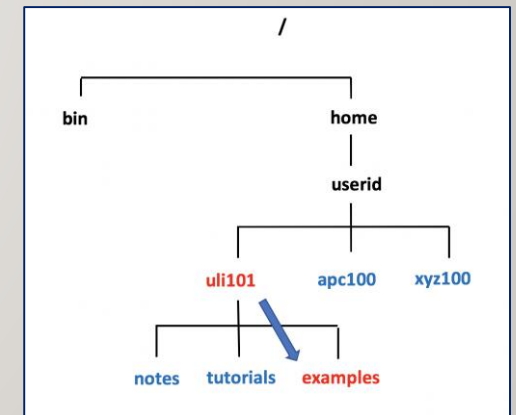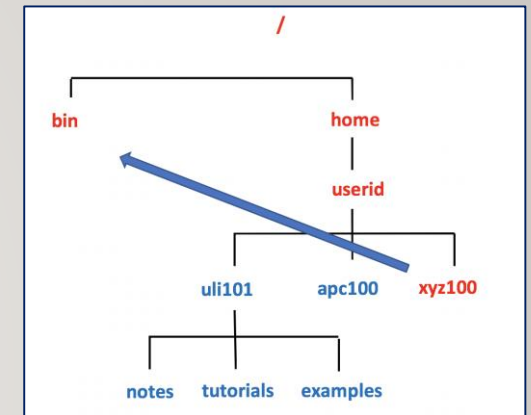
Relative Pathnames

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

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

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

FILE PATHNAME TYPES

Relative Pathnames

Relative Pathname Symbols:

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

- Two periods ".." represents the **parent** directory
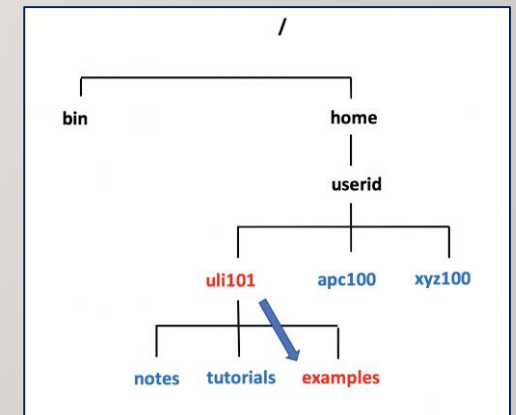 (i.e. one directory level up)

Advantages of using Relative Pathnames:
- Possibly a shorter pathname (less typing)

Examples:

../../../bin
examples
./examples
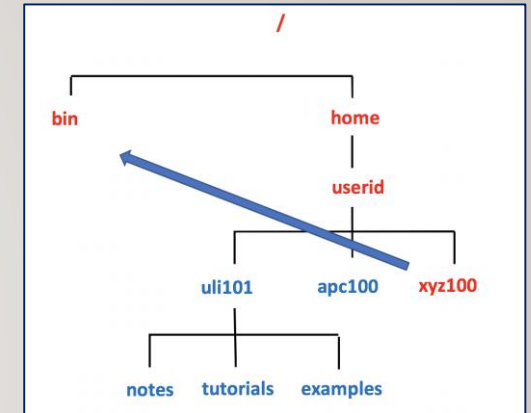
FILE PATHNAME TYPES

Relative-to-home Pathnames

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

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

Advantages of using Relative-to-Home Pathnames:

- Possibly a shorter pathname (less typing)

You can place a **username** IMMEDIATELY **after** the tilde character to
represent another user's home directory (for example: **~jane = /home/jane**)

Examples:

```
~/uli101/examples
~/uli101/notes
~murray.saul/uli101/notes
```
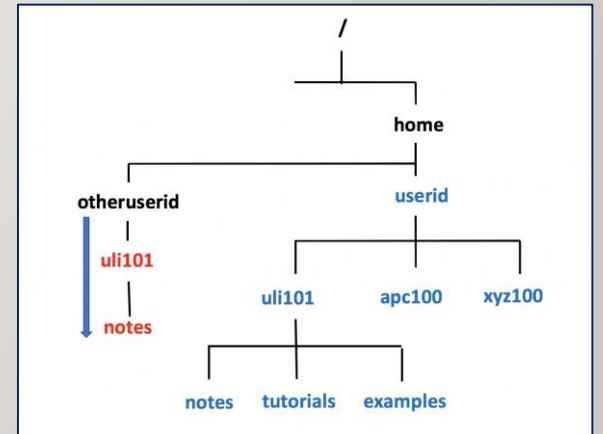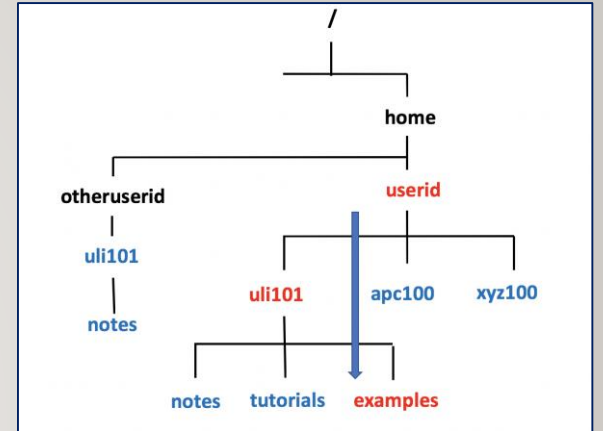
FILE PATHNAME TYPES

Instructor Demonstration

Your instructor will now demonstrate how to issue Unix / Linux commands
using absolute, relative and relative-to-home pathnames
for directory / file management:

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

HANDS-ON TIME / HOMEWORK

Getting Practice

Perform online **Tutorial3: Advanced File Management / Quoting Special Characters** (**ctrl-click** to open link):

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

- LINUX PRACTICE QUESTIONS (Questions 1 – 8)

OSL640: INTRODUCTION TO OPEN SOURCE SYSTEMS

WEEK 3: LESSON 2

FILENAME EXPANSION
QUOTING SPECIAL CHARACTERS

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 <u>know</u> 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 work.txt
```

FILENAME EXPANSION

Common File Expansion Symbols

Below are the most common Filename 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.*
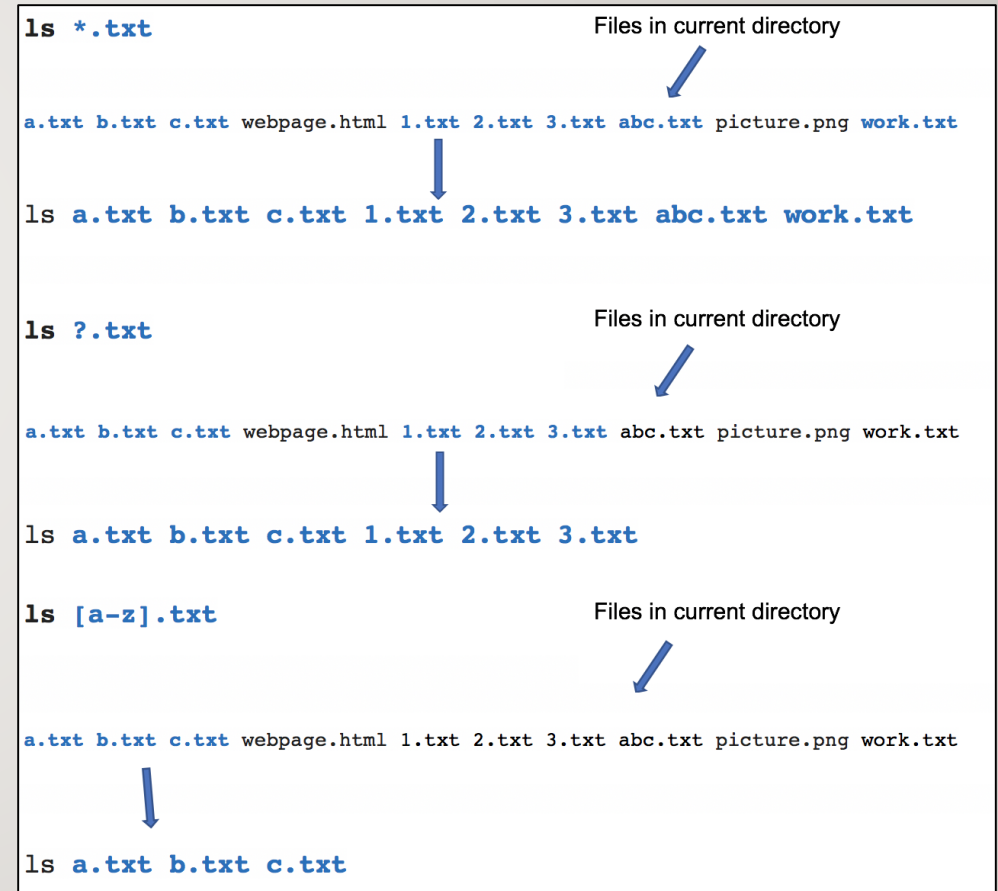
Reference: https://www.linuxnix.com/10-file-globbing-examples-linux-unix/

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.

```
ls *.txt                                    Files in current directory

a.txt b.txt c.txt webpage.html 1.txt 2.txt 3.txt abc.txt picture.png work.txt

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


ls ?.txt                                    Files in current directory

a.txt b.txt c.txt webpage.html 1.txt 2.txt 3.txt abc.txt picture.png work.txt

ls a.txt b.txt c.txt 1.txt 2.txt 3.txt

ls [a-z].txt                                Files in current directory


a.txt b.txt c.txt webpage.html 1.txt 2.txt 3.txt abc.txt picture.png work.txt

ls a.txt b.txt c.txt
```

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

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 is a method to instruct the Linux shell to ignore that special character and use only as **regular text**.

There are **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 \ <u>before</u> a special character (works for ALL special characters)	`echo \*`
Contain Special character within single **quotes** ' ' (work for ALL special characters)	`echo '* hello *'`
Contain special characters within **double-quotes** " " **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)	`echo "* hello *"`

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

HANDS-ON TIME / HOMEWORK

Getting Practice

To get practice perform the online tutorial
Tutorial3: Unix / Linux File Management (**ctrl-click** to open link):

- INVESTIGATION 2: FILENAME EXPANSION

- INVESTIGATION 3: QUOTING SPECIAL CHARACTERS

- LINUX PRACTICE QUESTIONS (Questions 9 – 13)