ULI101: INTRODUCTION TO UNIX / LINUX AND THE INTERNET

WEEK 5: LESSON 1

ADDITIONAL LINUX COMMANDS
REDIRECTION SYMBOLS
/DEV/NULL FILE , THE HERE DOCUMENT

LESSON 5.1 TOPICS

Redirection – Part 1

- Additional Commands (**tr**, **cut**, **wc**)
- Concepts:
  - **Standard Input, Standard Output**, **Standard Error**
- Redirection Symbols: (<, >, >>, 2>, 2>>)
- Additional Redirection Concepts:
  - **/dev/null** File, The **Here Document**

Perform Week 5 Tutorial

- Investigation 1

ADDITIONAL FILE MANIPULATION COMMANDS

Additional Text File Manipulation Commands

Here are some additional commands to manipulate content of text files.

Command	Description
`tr`	Used to **translate** characters to different characters. lower case to upper case tr -d delete eg. `tr 'a-z' 'A-Z' < filename`
`cut`	Used to **extract** fields and characters from records. The option **-c** option is used to cut by a character or a range of characters. The **-f** option indicates the field number or field range to display (this may require using the **-d** option to indicate the field separator (delimiter) which is tab by default). eg. `cut -c1-5 filename` , `cut -d":" -f2 filename`
`wc`	Displays various **counts** of the contents of a file. The **–l** option displays the number of lines, the **–w** option displays the number of words, and the **–c** option displays the number of characters. eg. `wc filename` , `wc -l filename` , `wc -w filename`

ADDITIONAL FILE MANIPULATION COMMANDS

Instructor Demonstration

Your instructor will now demonstrate using the following Linux commands:

- **tr**
- **cut**
- **wc**

REDIRECTION

Redirection can be defined as **changing** *from* where commands **read input** *to* where commands **send output**. You can redirect the input and output of a command.

For redirection, **meta characters** are used.

Redirection can be into a **file** (shell meta characters are angle **brackets** '<', '>') or a **program** (shell meta characters are **pipe** symbol '|').

Reference: https://www.javatpoint.com/linux-input-output-redirection

REDIRECTION

A lot of programs (as we've seen in previous sections) allow us to supply a file as a command line argument and it will read and process the contents of that file. You'll notice that when we ran wc supplying the file to process as a command line argument, the output from the program included the name of the file that was processed. When we ran it redirecting the contents of the file into wc the file name was not printed. This is because whenever we use redirection or piping, the data is sent anonymously. So in the above example, wc recieved some content to process, but it has no knowledge of where it came from so it may not print this information. As a result, this mechanism is often used in order to get ancillary data (which may not be required) to not be printed

Standard input (**stdin**) is a term which describes from where a command receives **input**.

The meta character "**<**" will redirect **stdin** into a command.

This would only apply to Unix/Linux commands that can **accept** *stdin* like **cat**, **more**, **less**, **sort**, **grep**, **uniq**, ***head***, ***tail***, **tr**, **cut**, and **wc**.

Examples:

```
tr 'a-z' 'A-Z' < words.txt
cat < abc.txt
sort < xyz.txt
```

sometimes we may wish to save it into a file to keep as a record, feed into another system, or send to someone else

REDIRECTION

When piping and redirecting, the actual data will always be the same, but the formatting of that data may be slightly different to what is normally printed to the screen. Keep this in mind.

saving to an existing file

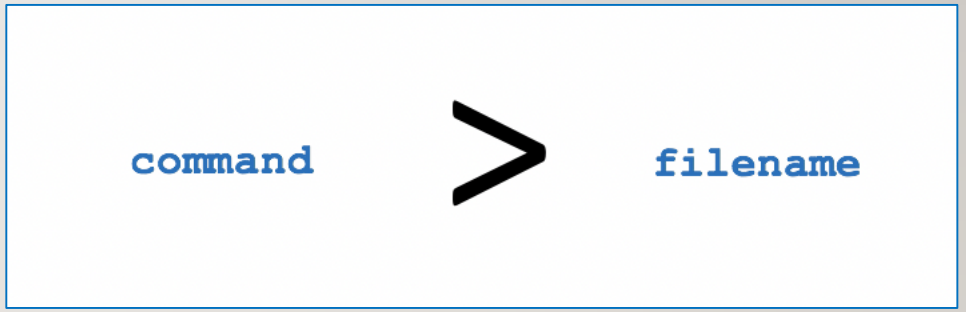
Standard output (**stdout**) describes where a command sends its **output**.

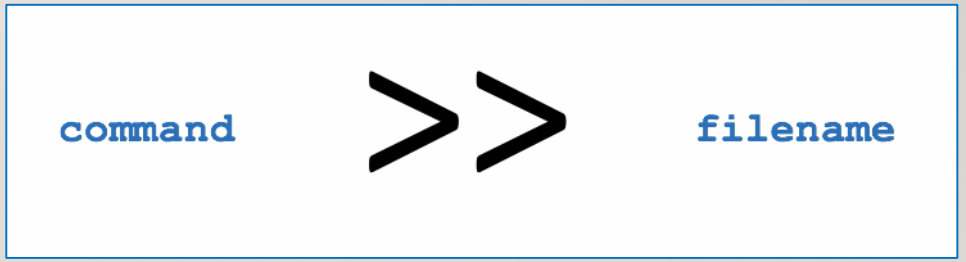
The meta character "**>**" will redirect **stdout** to a file either **creating** a new file if it doesn't exist or **overwriting** the content of an existing file.

The meta characters "**>>**" will redirect **stdout** to a file either **creating** a new file if it doesn't exist or **adding** stdout to the **bottom** to the existing file's contents.

Examples:

```
ls -l
ls -l > detailed-listing.txt
ls /bin >> output.txt
```





REDIRECTION

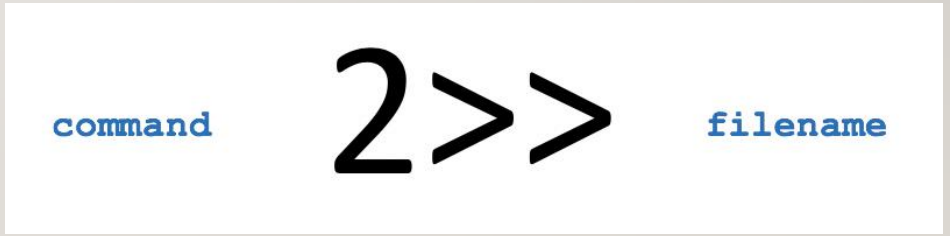
Standard Error (stderr) describes where a command sends its **error messages**.

The meta characters "**2>**" will redirect **stderr** to a file either **creating** a new file if it doesn't exist or **overwriting** the content of an existing file.

The meta characters "**2>>**" will redirect **stderr** to a file either **creating** a new file if it doesn't exist or **adding** stdout to the **bottom** to the existing file's contents.

Examples:

```
PWD
PWD 2> error-message.txt
PWD 2 >> error-messages.txt
PWD 2> /dev/null
```



REDIRECTION

The **/dev/null** file (sometimes called the **bit bucket** or **black hole**) is a special system file that **discards** stdout or stderr.

This is useful to "*throw-away*" **unwanted** command output or errors.

Examples:

```
ls 2> /dev/null
ls > /dev/null
find / -name "tempfile" 2> /dev/null
```

REDIRECTION

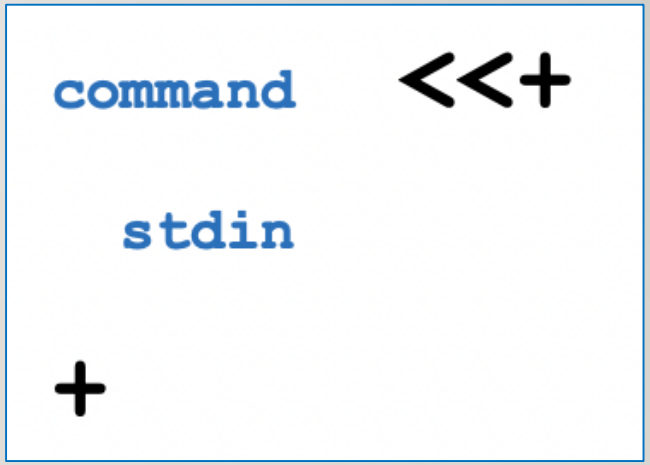
The **Here Document** allows stdin to be redirected into a command **within** the command-line.

The meta characters "**<<+**" will redirect **stdin** into the command.
The **+** symbol is used to identify the beginning and ending of the stdin.

You can use ANY symbol or series of characters to mark stdin as long as that symbols or characters are IDENTICAL and the ending symbol or characters are on a **separate** line with only that symbol or characters.

Example:

```
cat <<+
Line 1
Line 2
Line 3
+
```



REDIRECTION

```
tr a-z A-Z << EOF
> hello
> line two
> Therere
> Line four
> EOF
```

Instructor Demonstration

Your instructor will now demonstrate redirection:

- Standard Input
- Standard Output
- Standard Error
- Both Standard Output and Standard Error
- Both Standard Input and Standard Output
- Redirecting to **/dev/null**
- The **Here Document**

HOMEWORK

Getting Practice

Perform **Week 5 Tutorial**

(Due: Friday Week 6 @ midnight for a 2% grade):

- INVESTIGATION 1: BASICS OF REDIRECTION
- LINUX PRACTICE QUESTIONS (Questions 1 – 4)