ULI101: INTRODUCTION TO UNIX / LINUX AND THE INTERNET

WEEK 5: LESSON 1

REDIRECTION – PART 1
ADDITIONAL LINUX COMMANDS
REDIRECTION SYMBOLS | /DEV/NULL | THE HERE DOCUMENT

LESSON 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
- Review Questions (Questions 1 – 4)

REDIRECTION

Additional Text File Manipulation Commands

The following Linux commands can be used with redirection.

Refer to the table below for additional text file manipulation commands:

Command	Description
tr	Used to **translate** characters to different characters. 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`

REDIRECTION

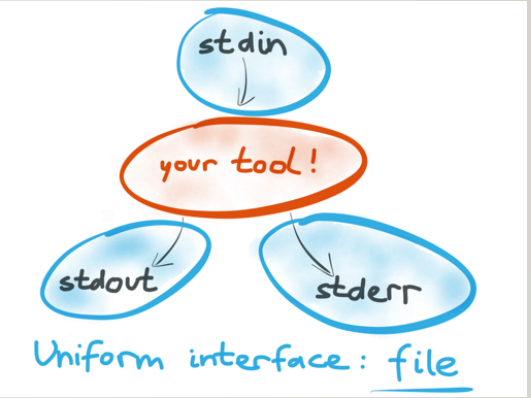
Instructor Demonstration

Your instructor will now demonstrate using the following Linux commands:

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

REDIRECTION

*...The three input/output (I/O) connections are called **standard input** (**stdin**), **standard output** (**stdout**) and **standard error** (**stderr**). Originally I/O happened via a physically connected system console (input via keyboard, output via monitor), but standard streams **abstract** this.*

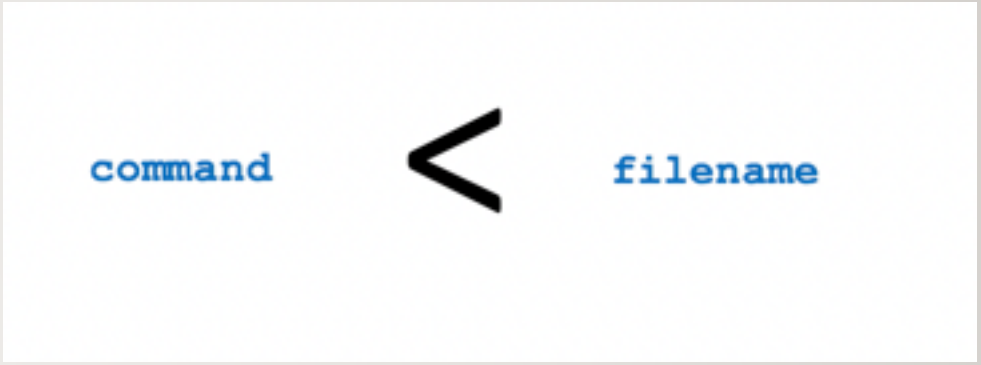


*When a command is executed via an interactive shell, the streams are typically connected to the text terminal on which the shell is running, but can be changed with **redirection** or a **pipeline**.*

Reference: https://en.wikipedia.org/wiki/Standard_streams

REDIRECTION

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



This would apply only to Unix/Linux commands that accept stdin input such as **cat**, **more**, **less**, **sort**, **grep**, **uniq**, ***head***, ***tail***, **tr**, **cut**, **wc**, etc.

Examples:

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

REDIRECTION

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

One greater sign will create a new file with the command's output or overwrite a file's previous content.

Two greater signs will add to the bottom of a file's existing content.

Examples:

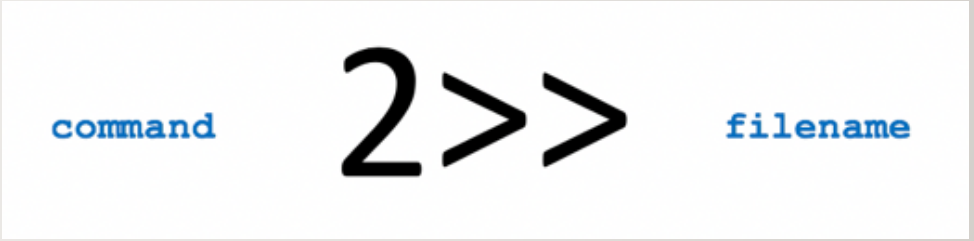
```
ls -l
ls -l > detailed-listing.txt
ls /bin >> output.txt
```

REDIRECTION

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

A "2" immediately followed by one greater sign will create a new file with the command's output or overwrite a file's previous content.

A "2" immediately followed by two greater signs will add to the bottom of a file's existing content.



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** all data written into it.

This is useful to discard unwanted command output.

Examples:

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

REDIRECTION

The **Here Document** is a section of a source code file that is treated as if it were a separate file.

Reference: https://en.wikipedia.org/wiki/Here_document

In Linux, it allows a user to redirect stdin from within the command itself.

Example:

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

REDIRECTION

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**

REDIRECTION

Getting Practice

To get practice to help perform **assignment #2**, perform **Week 5 Tutorial:**

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

ULI101: INTRODUCTION TO UNIX / LINUX AND THE INTERNET

WEEK 5: LESSON 2

REDIRECTION – PART 2
PIPELINE COMMANDS
MULTIPLE / MULTILINE COMMANDS

LESSON 2 TOPICS

Redirection – Part 2

- Purpose of Pipeline Commands
- Linux Pipeline Command Syntax
- **tee** Command

Multiple / Multi-Line Commands

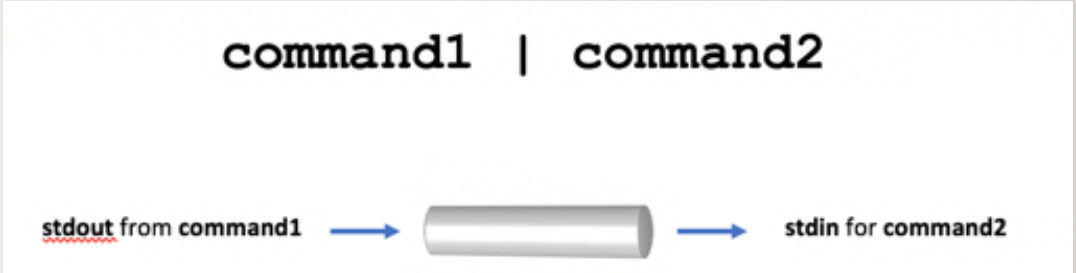
- Multiple Linux Commands using Semicolon ";" and Grouping ( )
- Issuing Large Linux Commands over Multiple Lines

Perform Week 5 Tutorial

- Investigations 2 and 3
- Review Questions (Questions 5 – 12)
- **Work on Assignment 2 (Section 3: Redirection & Pipes)**

PIPELINE COMMANDS

Pipeline Commands use a symbol (called a pipe) to allow a command's **standard output** to be redirected into the **standard input** of other commands WITHOUT having to use **temporary** files.



Pipes that are used in a **pipeline command** are represented by the **pipe** "|" symbol.

Therefore, a few simple commands can be **combined** to form a more powerful pipeline command.

PIPELINE COMMANDS

Filters

Commands to the **right** of the pipe symbol are referred to as **filters**.

They are referred to as *filters* since those commands are used to **modify** the stdout of the previous command.

Many commands can be "piped" together, but these commands (filters) must be chained in a specific order, depending on what you wish to accomplish

PIPELINE COMMANDS

Examples:

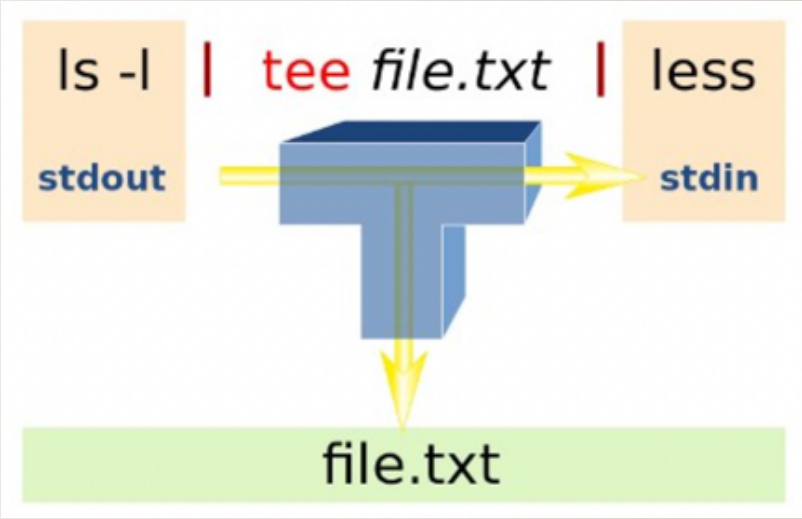

```
ls -al | more
ls | sort -r
ls | sort | more
ls -l | cut -d" " -f2 | tr 'a-z' 'A-Z"
ls | grep Linux | head -5
head -7 filename | tail -2
```

PIPELINE COMMANDS

The **tee** utility can be used to split the flow of information.

The tee option **-a** can be used to add content to the bottom of an existing file as opposed to overwriting the file's previous contents.

The reason for the name "**tee**" is that the splitting of the flow of information resembles a capital T.



Examples:

```
ls | tee unsorted.txt | sort
ls | grep Linux | tee matched.txt | more
ls | head -5 | tee -a listing.txt
```

MULTIPLE / MULTI-LINE COMMANDS

Besides piping, there are other ways that multiple commands may be placed in one line:
commands may be separated by **semi-colons**.

```
command1 ; command2 ; command3
```

Example:

```
sleep 5; ls
```

Multiple commands can also be **grouped** by using parentheses.

Example:

```
(echo "Who is on:"; who) > whoson
```

(***Note:*** *<u>all</u> command output is sent to a file*)

MULTIPLE / MULTI-LINE COMMANDS

Commands may also be **spread-out over multiple lines**, making it easier (for humans) to interpret a long command. You can add a **backslash** symbol "\" at the end of a line, to get rid of the special meaning of newline (to end a command line)

```
command1 |   \
command2 |   \
command3
```

Example:

```
echo "This will be split over multiple \
lines. Note that the shell will realize \
that a pipe requires another command, so \
it will automatically go to the next line" \
| tr '[a-z]' '[A-Z]'
```

PIPELINE COMMANDS
MULTIPLE / MULTI-LINE COMMANDS

Instructor Demonstration

Your instructor will now demonstrate how to issue Pipeline / Multiple / Multi-Line Linux Commands:

- Linux Pipeline Commands
- Linux Pipeline Commands using the **tee** Command
- Multiple Linux Commands using semicolon ";"
- Multiple Linux Commands using Grouping ( )
- Mult-Line Linux Commands using Backslash \

PIPELINE COMMANDS
MULTIPLE / MULTI-LINE COMMANDS

Getting Practice

To get practice to help perform assignment #1, perform the online tutorial **Tutorial5** (**ctrl-click** to open link):

- INVESTIGATION 2: REDIRECTION USING PIPES
- INVESTIGATION 3: ISSUING MULTIPLE UNIX/LINUX COMMANDS
- LINUX PRACTICE QUESTIONS (Questions 6 – 12)
- Perform Online Assignment 2 (**Section 3: Redirection & Pipes**)