ULI101: INTRODUCTION TO UNIX / LINUX AND THE INTERNET

WEEK 10 LESSON 1

THE SED UTILITY

LESSON 1 TOPICS

The `sed` Utility

- Definition / Purpose
- Usage
- Using **`sed`** as a Filter with Pipeline Commands
- Demonstration

Perform Week 10 Tutorial

- Investigation 1
- Review Questions (**Parts A** and **B**)

SED UTILITY

Purpose

The **sed** command stands for **Streaming Editor**.

The sed command is used to **manipulate text** that is contained in a **text file** or via a **pipeline command**.

Although the sed command does NOT change content inside a text file, this command acts like a *"on-the-fly"* text editor to display modified text on the **screen**, **redirect** to a file or act as a **filter** within a pipeline command.

SED UTILITY

Usage:

```
sed [-n] 'address instruction' filename
```

How it Works:

- The sed command reads **all lines in the input file** and will be exposed to the expression (i.e. area contained within **quotes**) one line at a time.
- The expression can be within **single** quotes or **double** quotes.
- The **expression** contains an **address** (match condition) and an **instruction** (operation).
- If the line matches the **address**, then it will perform the **instruction**.
- Lines will display be default unless the **–n** option is used to suppress default display

SED UTILITY

Usage:

```
sed [-n] 'address instruction' filename
```

Addresses:

- Can use a **line number**, to select a specific line (for example: `5`)
- Can specify a **range of line numbers** (for example: `5,7`)
- Regular expressions are contained within **forward slashes** (e.g. **`/regular-expression/`**)
- Can specify a **regular expression** to select all lines that match a pattern (e.g `/^[0-9].*[0-9]$/`)
- If **NO** address is present, the **instruction** will apply to **ALL** lines

SED UTILITY

Usage:

```
sed [-n] 'address instruction' filename
```

Common Instructions:

- **p** **Print** lines that match the address (commonly used with **-n** option)
- **d** Omit (**delete**) display of lines that match the address
- **q** Print lines including line that matches address and then **quit** processing
- **s** **Substitute** text to replace a matched regular expression (similar *search and replace*)

SED UTILITY

Example 1

The following sed command line displays all lines in the readme file that contain the word **line** (all lowercase).

In addition, because there is no **–n** option, sed displays all the lines of input.

As a result, sed displays the lines that contain the word line **twice**.

```
sed '/line/ p' readme
Line one.
The second line.
The second line.
The third.
This is line four.
This is line four.
Five.
This is the sixth sentence.
This is line 7.
This is line 7.
Eight and last.
```

Unless you instruct it not to, sed sends **all lines**, selected or not to standard output.

When you use the `-n` option on the command line, sed sends only those lines to stdout that you specify with the print `p` command

SED UTILITY

Example 2

The following sed command displays contents of a file from a **range** of line numbers.

```
sed -n '3,6 p' readme
The third.
This is line four.
Five.
This is the sixth sentence.
```

The print `p` instruction using the `-n` option only displays lines **3** through **6**.

SED UTILITY

Example 3

The following sed command displays the first **five** lines of text just as **a head -5** lines command would.

```
sed '5 q' readme
Line one.
The second line.
The third.
This is line four.
Five.
```

The sed command prints **all lines**, beginning from the first line, In this example, sed will **terminate** when **line 5** is matched.

SED UTILITY

Example 4

The following sed command displays a **TAB** character for lines contained in a file.

```
$ sed 's/^./\t&/' readme
    Line one.
    The second line.
    The third.
    etc...
```

The regular expression in the following instruction (`^.`) matches one character at the beginning of every line that is not empty.

The replacement string (between the second and third forward slashes) contains a backslash escape sequence that represents a **TAB** character (`\t`) followed by an ampersand (`&`).

The **ampersand** character (`&`) takes on the value of what the regular expression matched.

SED UTILITY

Example 5

The following sed command uses a **regular expression** and the **quit** instruction.

```
sed '/[0-9][0-9][0-9]$/ q' myfile
sfun 11
cool 12
Super 12a
Happy112
```

The regular expression in the following expression `[0-9][0-9][0-9]$` matches **three digits** at the end of a line.

The command will process the file, one-line at a time, beginning at the top and (by default) outputting each line to standard output.

Once the regular expression is matched, it will display the matched line and stop processing the *sed* command.

SED UTILITY

Using sed Utility as a Filter with Pipeline Commands

Although sed can be used as a streaming editor for text contained within a text file, the sed command can also be used as a **filter** within a **pipeline command**.

Examples

```
ls | sed 's/^[0-9]/x/g'
echo "I like Linux" | sed 's/ /,/g'
```

SED UTILITY

Instructor Demonstration

Your professor will demonstrate additional examples using the **sed** utility.

Pathname of cars database:
~murray.saul/uli101/cars

Commands

```
sed -n '3,6 p' cars
sed '5 d' cars
sed '5,8 d' cars
sed '5 q' cars
sed -n '/chevy/ p' cars
sed '/chevy/ d' cars
sed '/chevy/ q' cars
sed 's/[0-9]/*/' cars
sed 's/[0-9]/*/g' cars
sed '5,8 s/[0-9]/*/' cars
sed 's/[0-9][0-9]*/*** & ***/' cars
```

Contents of **cars** database file:

```
plym fury 77 73 2500
chevy nova 79 60 3000
ford mustang 65 45 17000
volvo gl 78 102 9850
ford ltd 83 15 10500
Chevy nova 80 50 3500
fiat 600 65 115 450
honda accord 81 30 6000
ford thundbd 84 10 17000
toyota tercel 82 180 750
chevy impala 65 85 1550
ford bronco 83 25 9525
```

HOMEWORK

Getting Practice

Perform **Week 10 Tutorial:**
(Due: Friday Week 11 @ midnight for a 2% grade):

- INVESTIGATION 1: USING THE SED UTILITY
- LINUX PRACTICE QUESTIONS (Parts A and B)