ULI101: INTRODUCTION TO UNIX / LINUX AND THE INTERNET

WEEK 11 LESSON 1

THE SED UTILITY

LESSON 1 TOPICS

The sed Utility

- Purpose / Usage
- Examples

Using sed as a Filter

- Purpose / Usage
- Examples

Perform Week 11 Tutorial

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

SED UTILITY

Purpose

***sed** (**stream editor**) is a Unix utility that **parses** and **transforms** text, using a simple, compact programming language... **sed** was one of the earliest tools to support regular expressions, and remains in use for text processing, most notably with the substitution command.*

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

SED UTILITY

Sed Utility Usage

Syntax:

```
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 (contained in quotes).
- If the line matches the **address**, then it will perform the **instruction**.

Using Address:

- can use a line number, to select a specific line (for example: `5`)
- can specify a range of line numbers (for example: `5,7)`
- can specify a regular expression to select all lines that match
  (e.g `/^happy[0-9]/`)
- When using regular expressions, you must delimit them with a forward-slash (`/`)
- default address (if none is specified) will match every line

SED UTILITY

Sed Utility Usage

Syntax:

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

Instruction:

- **Action** to take for matched line(s)
- Refer to table below for list of some **common instructions** and their purpose

Instruction	Purpose
p	print line(s) that match the address (usually used with -n option)
d	delete line(s) that match the address
q	quit processing at the first line that matches the address
s	substitute text to replace a matched regular expression, similar to vi substitution

SED UTILITY

Example 1

The following 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

In this example, sed displays part of a file based on 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 selects and displays lines **3 through 6**.

SED UTILITY

Example 3

This command line uses the **quit** instruction to cause sed to display only the beginning of a file. In this case sed 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.
```

Note: sed prints **all lines**, beginning from the first line, by default. In this example, sed will terminate when the address (in this case, **line 5**) is matched.

SED UTILITY

Example 4

This example uses a regular expression as the pattern.

```
$ 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 slashes) contains a backslash escape sequence that represents a **TAB** character (`\t`) followed by an ampersand (`&`).

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

SED UTILITY

Example 5

This example uses a regular expression as the pattern again.

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

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

The instruction (`q`) instructs sed to stop processing lines once the regular expression is matched

The command will process the file, one-line at a time, beginning at the top, and (by default) outputs each line to standard output. Once the regular expression matches, it will display the matched line, and stop processing the file any further.

SED UTILITY

Instructor Demonstration

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

```
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:

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

SED UTILITY

Using sed Utility as a Filter

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

Examples

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

SED UTILITY

Getting Practice

To get practice to help perform **online assignment #3**, perform **Week 11 Tutorial:**

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

ULI101: INTRODUCTION TO UNIX / LINUX AND THE INTERNET

WEEK 11: LESSON 2

THE AWK UTILITY

LESSON 2 TOPICS

The awk Utility

- Purpose / Usage
- Examples

Using the awk Utility as a Filter

- Purpose / Usage
- Examples

Perform Week 11 Tutorial

- Investigation 2
- Review Questions (**Parts C and D**)

AWK UTILITY

Purpose

awk *is a utility that enables a programmer to write tiny but effective programs in the form of statements that define text patterns that are to be searched for in each line of a document and the action that is to be taken when a match is found within a line.*

Awk is mostly used for pattern scanning and processing. It searches one or more files to see if they contain lines that matches with the specified patterns and then performs the associated actions.

Reference: https://www.geeksforgeeks.org/awk-command-unixlinux-examples/

AWK UTILITY

Usage

```
awk options 'selection _criteria {action }' file-name
```

Notes:

- The **awk** command reads all lines in the input file and will be exposed to the expression (contained within quotes) for processing.
- Expression (contained in quotes) represents selection criteria, and action to execute (contained within braces) if selection criteria is matched
- If no pattern is specified, awk selects all lines in the input
- If no action is specified, awk copies the selected lines to standard output
- You can use parameters like `$1`, `$2` to represent first field, second field, etc.
- You can use the `-F` option with the awk command to specify the field delimiter.

AWK UTILITY

Patterns: Regular Expressions

You can use a regular expression, enclosed within slashes, as a pattern.

The ~ operator tests whether a field or variable matches a regular expression

The `!~` operator tests for no match.

You can perform both numeric and string comparisons using relational operators (`>` , `>=` , `<` , `<=` , `==` , `!=`)

You can combine any of the patterns using the Boolean operators `||` (OR) and `&&` (AND).

You can use **built-in variables** (like `NR` or "record number" representing line number) with comparison operators

AWK UTILITY

Patterns: Relational Operators

The following operators (in the table below) can be used with the **awk** utility to pattern searching. Since those symbols are used within the expression, they are NOT confused with redirection symbols.

Relational Operator	
<	Less than
<=	Less than or equal to
==	Equal to
!=	Not equal to
>	Greater than
>=	Greater than or equal to

AWK UTILITY

Example 1

```
cat data.txt
Saul Murray professor
David Ward retired
Fernades Mark professor

awk '{print}' data.txt
Saul Murray professor
David Ward retired
Fernades Mark professor
```

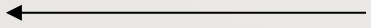
If no pattern is specified, awk selects **all lines** in the input

AWK UTILITY

Example 2

```
cat data.txt
Saul Murray professor
David Ward retired
Fernades Mark professor
```

```
awk '/^[F-Z]/ {print}' data.txt
Saul Murray professor
Fernades Mark professor
```



You can use a regular expression, enclosed within slashes, as a pattern.

In this case, the pattern is matched at the BEGINNING of each line (record) read from the input file.

AWK UTILITY

Example 3

```
cat data.txt
Saul Murray professor
David Ward retired
Fernades Mark professor

awk '/^[F-Z]/' data.txt
Saul Murray professor
Fernades Mark professor
```

If no action is specified, awk copies the selected lines to standard output

AWK UTILITY

Using Variables with awk Utility

You can use parameters which represent fields within records (lines) within the expression of the awk utility.

The parameter `$0` represents all of the fields contained in the record (line).

The parameters `$1`, `$2`, `$3` ... `$9` represent the first, second and third to the 9th fields contained within the record. Parameters greater than nine requires the value of the parameter to be placed within braces
(for example: `${10}`, `${11}`, `${12}`, etc)

Unless you separate items in a print command with commas,
awk **catenates** them.

AWK UTILITY

Example 4

```
cat data.txt
Saul Murray professor
David Ward retired
Fernades Mark professor
```

```
awk '$1 ~ /^[F-Z]/ {print}' data.txt
Saul Murray professor
Fernades Mark professor
```

The parameters **$1**, **$2**, **$3 ... $9** represent the first, second and third to the 9th fields contained within the record.

```
awk '$3 ~ /retired/ {print}' data.txt
David Ward retired
```

The ~ operator tests whether a field or variable matches a regular expression

AWK UTILITY

Example 5

```
cat data.txt
Saul Murray professor
David Ward retired
Fernades Mark professor
```

```
awk '$3 !~ /retired/ {print}' data.txt
Saul Murray professor
Fernades Mark professor
```

The **!~** operator tests for no match.

AWK UTILITY

Example 6

```
cat customer.dat
A100 Acme-Inc. 5400
R100 Rain-Ltd. 11224
T100 Toy-Inc. 3413
awk '$3 > 10000 {print}' customer.dat
R100 Rain-Ltd. 11224
awk '$3 <= 6000 {print}' customer.dat
A100 Acme-Inc. 5400
T100 Toy-Inc. 3413
```

Using relational operators with the awk command.

AWK UTILITY

Example 7

```
cat customer.dat
A100 Acme-Inc. 5400
R100 Rain-Ltd. 11224
T100 Toy-Inc. 3413
awk '$3 >= 5000 && $3 <= 10000  {print}' customer.dat
A100 Acme-Inc. 5400
awk '$3 <= 5000 || $3 >= 10000  {print}' customer.dat
R100 Rain-Ltd. 11224
T100 Toy-Inc. 3413
```


Using the **&&** and **||** conditional operators with the awk command.

AWK UTILITY

Example 8

```
cat customer.dat
A100 Acme-Inc. 5400
R100 Rain-Ltd. 11224
T100 Toy-Inc. 3413
awk '$3 > 10000 {print $1,$2}' customer.dat
R100 Rain-Ltd.
awk '$2 ~ /Acme-Inc./ {print $3}' customer.dat
5400
```


Using parameters to specify fields with print command to display output.

AWK UTILITY

Other Variables for awk Utility

The table below show other variables that can be used with the awk command.

Variable Name	
FILENAME	Name of the current input file
FS	Input field separator (default: SPACE or TAB)
NF	Number of fields in the current record
NR	Record number of the current record
OFS	Output field separator (default: SPACE)
ORS	Output record separator (default: NEWLINE)
RS	Input record separator (default: NEWLINE)

AWK UTILITY

Example

```
cat customer.dat
A100 Acme-Inc. 5400
R100 Rain-Ltd. 11224
T100 Toy-Inc. 3413
awk '{print NR,$0}' customer.dat
1 A100 Acme-Inc. 5400
2 R100 Rain-Ltd. 11224
3 T100 Toy-Inc. 3413
awk 'NR ==2 {print}' customer.dat
R100 Rain-Ltd. 11224
awk 'NR > 1 && NR < 5{print}' customer.dat
R100 Rain-Ltd. 11224
T100 Toy-Inc. 3413
```

Using **NR** (record number) variable with the awk utility

AWK UTILITY

Using awk Utility as a Filter

Although awk can be used as a streaming editor for text contained within a text file, awk can also be used as a filter using a pipeline command.

Examples

```
ls | awk '{print $1,$2}'
```

AWK UTILITY

Instructor Demonstration

Your instructor will demonstrate additional examples of using the **awk** utility.

AWK UTILITY

Getting Practice

To get practice to help perform **online assignment #3**, perform **Week 11 Tutorial:**

- INVESTIGATION 2: USING THE AWK UTILITY
- LINUX PRACTICE QUESTIONS (Parts C and D)