

# Tutorial10: Sed & Awk Utilities

 **DO NOT USE THIS VERSION OF THE LAB. This page will no longer be updated.**  
**New version here:** <https://seneca-ictoer.github.io/ULI101/A-Tutorials/tutorial10>  
**Andrew's students please go here:** [http://wiki.littlesvr.ca/wiki/OPS145\\_Lab\\_9](http://wiki.littlesvr.ca/wiki/OPS145_Lab_9)

## Contents

- 1 USING SED & AWK UTILITIES
  - 1.1 Main Objectives of this Practice Tutorial
  - 1.2 Tutorial Reference Material
- 2 KEY CONCEPTS
  - 2.1 Using the sed Utility
  - 2.2 Using the awk Utility
- 3 INVESTIGATION 1: USING THE SED UTILITY
- 4 INVESTIGATION 2: USING THE AWK UTILITY
- 5 LINUX PRACTICE QUESTIONS

## USING SED & AWK UTILITIES

### Main Objectives of this Practice Tutorial

- Use the **sed** command to **manipulate text** contained in a file.
- List and explain several **addresses** and **instructions** associated with the **sed** command.
- Use the **sed** command as a **filter** with Linux pipeline commands.
- Use the **awk** command to **manipulate text** contained in a file.
- List and explain **comparison operators**, **variables** and **actions** associated with the **awk** command.
- Use the **awk** command as a **filter** with Linux pipeline commands.

### Tutorial Reference Material

Course Notes	Linux Command/Shortcut Reference	YouTube Videos
<b>Slides:</b> <ul style="list-style-type: none"><li>▪ Week 10 Lecture 1 Notes: PDF (<a href="https://wiki.cdo.t.senecacollege.ca/uli101/slides/ULI101-1">https://wiki.cdo.t.senecacollege.ca/uli101/slides/ULI101-1</a>)</li></ul>	<b>Text Manipulation:</b> <ul style="list-style-type: none"><li>▪ Purpose of using the sed utility (<a href="https://www.digitalocean.com/community/tutorials/the-basics-of-using-the-sed-stream-ed">https://www.digitalocean.com/community/tutorials/the-basics-of-using-the-sed-stream-ed</a>)</li></ul>	<b>Commands:</b> <ul style="list-style-type: none"><li>▪ sed (<a href="https://man7.org/linux/man-pages/man1/sed.1p.html">https://man7.org/linux/man-pages/man1/sed.1p.html</a>)</li></ul> <b>Brauer Instructional Videos:</b> <ul style="list-style-type: none"><li>▪ Using the sed Utility (<a href="https://www.youtube.com/watch?v=npU6S61AIko&amp;list=PLU1b1f-2Oe90TuYfifnWulINjMv_Wr16N&amp;index=14">https://www.youtube.com/watch?v=npU6S61AIko&amp;list=PLU1b1f-2Oe90TuYfifnWulINjMv_Wr16N&amp;index=14</a>)</li></ul>

- 0.1.pdf) | PPTX (<https://wiki.cdot.senecacollege.ca/uli101/slides/ULI101-10.1.pptx>)
- Week 10 Lecture 2 Notes: PDF (<https://wiki.cdot.senecacollege.ca/uli101/slides/ULI101-10.2.pdf>) | PPTX (<https://wiki.cdot.senecacollege.ca/uli101/slides/ULI101-10.2.pptx>)
- awk (<http://man7.org/linux/man-pages/man1/awk.1p.html>)
- Using the awk Utility ([https://www.youtube.com/watch?v=OV3XzjDYgJo&list=PLU1b1f-2Oe90TuYfifnWulINjMv\\_Wr16N&index=13](https://www.youtube.com/watch?v=OV3XzjDYgJo&list=PLU1b1f-2Oe90TuYfifnWulINjMv_Wr16N&index=13))
- Purpose of using the awk utility (<https://www.digitalocean.com/community/tutorials/how-to-use-the-awk-language-to-manipulate-text-in-linux>)

# KEY CONCEPTS

## Using the 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 (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 by default unless the **-n** option is used to suppress default display

### 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**)
- 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

### Instruction:

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

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

## Using the awk Utility

### Usage:

```
awk [-F] 'selection-criteria {action}' file-name
```

### How It Works:

- The **awk** command reads all lines in the input file and will be exposed to the expression (contained within quotes) for processing.
- The **expression** (contained in quotes) represents **selection criteria**, and **action** to execute contained within braces {}
- if selection criteria is matched, then action (between braces) is executed.
- The **-F** option can be used to specify the default **field delimiter** (separator) character eg. **awk -F";"** (would indicate a semi-colon delimited input file).

### Selection Criteria

- You can use a regular expression, enclosed within slashes, as a pattern. For example: **/pattern/**
- The **~** operator tests whether a field or variable matches a regular expression. For example: **\$1 ~ /^[0-9]/**
- The **!~** operator tests for no match. For example: **\$2 !~ /line/**
- 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. For example: **NR >= 1 && NR <= 5**

### Action (execution):

- Action to be executed is contained within braces {}
- The **print** command can be used to display text (fields).
- 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.)
- You can use built-in **variables** (such as **NR** or "record number" representing line number) eg. **{print NR,\$0}** (will print record number, then entire record).

## INVESTIGATION 1: USING THE SED UTILITY

**ATTENTION:** This online tutorial will be required to be completed by **Friday in week 11 by midnight** to obtain a grade of **2%** towards this course

In this investigation, you will learn how to manipulate text using the **sed** utility.

### Perform the Following Steps:

1. **Login** to your matrix account and confirm you are located in your **home** directory.

2. Issue a Linux command to create a directory called **sed**

3. Issue a Linux command to change to the **sed** directory and confirm that you are located in the **sed** directory.

4. Issue the following Linux command to copy the data.txt file  
(**copy and paste** to save time):

```
cp -ulil01/tutorialfiles/data.txt ~/sed
```

5. Issue the **more** command to quickly view the contents of the **data.txt** file.

When finished, exit the more command by pressing the letter **q**

The **p** instruction with the **sed** command is used to  
**print** (i.e. *display*) the contents of a text file.

6. Issue the following Linux command:

```
sed 'p' data.txt
```

**NOTE: You should notice that each line appears twice.**

The reason why standard output appears twice is that the sed  
command  
(without the **-n option**) displays all lines regardless of an address  
used.

We will use **pipeline commands** to both display stdout to the screen and save to files  
for confirmation of running these pipeline commands when run a **checking-script** later in this investigation.

7. Issue the following Linux pipeline command:

```
sed -n 'p' data.txt | tee sed-1.txt
```

What do you notice? You should see only one line.

You can specify an **address** to display lines using the sed utility  
(eg. *line #*, **line #s** or range of **line #s**).

8. Issue the following Linux pipeline command:

```
sed -n '1 p' data.txt | tee sed-2.txt
```

You should see the first line of the text file displayed.

What other command is used to only display the first line in a file?

9. Issue the following Linux pipeline  
command:

```
sed -n '2,5 p' data.txt |  
tee sed-3.txt
```

What is displayed? How would you  
modify the sed command to display  
the line range 10 to 50?

The **s** instruction is used to **substitute** text  
(a similar to method was demonstrated in the vi editor in tutorial 9).

10. Issue the following Linux pipeline command:

```
sed '2,5 s/TUTORIAL/LESSON/g' data.txt | tee sed-4.txt | more
```

What do you notice? View the original contents of lines 2 to 5 in the **data.txt** file  
in another shell to confirm that the substitution occurred.

```
Display from file "lab8" the total number of lines  
that contain the actual word "word" (upper or lowercase):  
that contain the actual word "word" (upper or lowercase):  
Display from file "lab8" the sorted and uniq contents  
Display from file "lab8" the sorted and uniq contents  
that begin with the pattern "therefore"  
that begin with the pattern "therefore"  
You have completed the REGULAR EXPRESSIONS PRACTICAL APPLICATION  
Proceed to the REVIEW QUESTION  
Proceed to the REVIEW QUESTION  
Press <ENTER> to return to the REVIEW TUTORIAL MENU  
Press <ENTER> to return to the REVIEW TUTORIAL MENU  
Display the total number of lines from file "srep7"  
Display the total number of lines from file "srep7"  
contain the pattern "truck" (upper or lowercase):  
contain the pattern "truck" (upper or lowercase):
```

Issuing the **p** instruction without using the **-n** option (to  
suppress original output) will display lines twice.

```
[ murray.saul ] sed -n '2,5 p' data.txt  
### The author of this script is using this tutorial to send students' ###  
* These tutorials may be used for: *  
* ALL TUTORIALS HAVE THE FOLLOWING STRUCTURE: *  
* TIPS FOR GETTING THE MOST OUT OF TUTORIALS: *
```

Using the sed command to display a **range** of lines.

The **q** instruction terminates or **quits** the execution of the sed utility as soon as it is read in a particular line or matching pattern.

11. Issue the following Linux pipeline command:

```
sed '11 q' data.txt | tee sed-5.txt
```

What did you notice? How many lines were displayed before the sed command exited?

You can use **regular expressions** to select lines that match a pattern. In fact, the sed command was one of the first Linux commands that used regular expression.

The rules remain the same for using regular expressions as demonstrated in **tutorial 9** except the regular expression must be contained within **forward slashes** (eg. **/regexp/**).

12. Issue the following Linux pipeline command:

```
sed -n '/^The/ p' data.txt | tee sed-6.txt
```

What do you notice?

13. Issue the following Linux pipeline command:

```
sed -n '/d$/ p' data.txt | tee sed-7.txt
```

What do you notice?

The **sed** utility can also be used as a **filter** to manipulate text that was generated from Linux commands.

14. Issue the following Linux pipeline command:

```
who | sed -n '/^[a-m]/ p' | tee sed-8.txt | more
```

What did you notice?

15. Issue the following Linux pipeline command:

```
ls | sed -n '/txt$/ p' | tee sed-9.txt
```

What did you notice?

16. Issue the following to run a checking script:

```
~uli101/week10-check-1
```

If you encounter errors, make corrections and **re-run** the checking script until you receive a congratulations message, then you can proceed.

```
[ murray.saul ] sed '11 q' data.txt | tee sed-5.txt
### Introductory Linux (Unix) course OPS224. The structure of the
### The author of this script is using this tutorial to send students'
* These tutorials may be used for:
* ALL TUTORIALS HAVE THE FOLLOWING STRUCTURE:
* TIPS FOR GETTING THE MOST OUT OF TUTORIALS:
* in case you need to use them while performing the tutorial
* ? Complete the entire tutorial
* ? Repeat the tutorial on a regular basis for review
* ? Try to use skills you have developed to accomplish other tasks
* A FEEDBACK section has been added to the end of each tutorial to
* allow ALL USERS to provide comments to improve the quality of the
```

Using the sed command with the **-q** option to display up to a line number, then quit.

The only character that is not recognized as a The standard output of this command should differ The following standard output displayed The caret ^ symbol is used to force a match at The caret symbol also has a different meaning The character class symbol [] is used to The symbol that you use is backslash \ and These symbols can be used together to force There is one occurrence of word "feedbacking" There are two types of characters that CANNOT

Using the sed command using regular expressions with **anchors**.

```
[ murray.saul ] ls | sed -n '/txt$/ p' | tee sed-9.txt
data.txt
sed-1.txt
sed-2.txt
sed-3.txt
sed-4.txt
sed-5.txt
sed-6.txt
sed-7.txt
sed-8.txt
sed-9.txt
```

Using the sed command with **pipeline** commands.

In the next investigation, you will learn how to manipulate text using the **awk** utility.

# INVESTIGATION 2: USING THE AWK UTILITY

In this investigation, you will learn how to use the awk utility to manipulate text and generate reports.

## Perform the Following Steps:

1. Change to your **home** directory and issue a command to **confirm** you are located in your *home* directory.
2. Issue a Linux command to create a directory called **awk**
3. Issue a Linux command to change to the **awk** directory and confirm you are located in the **awk** directory.

Let's download a database file that contains information regarding classic cars.

4. Issue the following linux command:  
`cp -ulil01/tutorialfiles/cars.txt ~/awk`
5. Issue the **cat** command to quickly view the contents of the **cars.txt** file.

The "**print**" action (command) is the default action of awk to print all selected lines that match a **pattern**.

This **action** (contained in braces) can provide more options such as printing **specific fields** of selected lines (or records) from a database.

6. Issue the following linux command all to display all lines (i.e. records) in the **cars.txt** database that matches the pattern (or "make") called **ford**:

`awk '/ford/ {print}' cars.txt`

We will use **pipeline commands** to both display stdout to the screen and save to files for confirmation of running these pipeline commands when run a **checking-script** later in this investigation.

```
[ murray.saul ] awk '/ford/ {print}' cars.txt
ford    mustang 65      45      10000
ford    ltd     83      15      10500
ford    thundbd 84      10      17000
ford    bronco  83      25      9500
```

Using the awk command to display matches of the pattern **ford**.

7. Issue the following linux pipeline command all to display records in the **cars.txt** database that contain the pattern (i.e. make) **ford**:

`awk '/ford/' cars.txt | tee awk-1.txt`

What do you notice? You should notice ALL lines displayed without using **search criteria**.

You can use *builtin variables* with the **print** command for further processing. We will discuss the following variables in this tutorial:

**\$0** - Current record (entire line)  
**\$1** - First field in record  
**\$n** - nth field in record  
**NR** - Record Number (order in database)  
**NF** - Number of fields in current record

```
[ murray.saul ] awk '/chevy/ {print $2,$3,$4,$5}' cars.txt
nova 79 60 3000
nova 80 50 3500
impala 65 85 1550
```

Using the awk command to print search results by **field number**.

For a listing of more variables, please consult your course notes.



8. Issue the following linux pipeline command to display the **model**, **year**, **quantity** and price in the **cars.txt** database for makes of **chevy**:

```
awk '/chevy/ {print $2,$3,$4,$5}' cars.txt | tee awk-2.txt
```

Notice that a **space** is the delimiter for the fields that appear as standard output.

The **tilde character** ~ is used to search for a pattern or display standard output for a particular field.

9. Issue the following linux pipeline command to display all **plymouths** (**plym**) by **model name**, **price** and **quantity**:

```
awk '$1 ~ /plym/ {print $2,$3,$4,$5}' cars.txt | tee awk-3.txt
```

You can also use **comparison operators** to specify conditions for processing with matched patterns when using the awk command. Since they are used **WITHIN** the awk expression, they are not confused with redirection symbols

< Less than  
<= Less than or equal  
> Greater than  
>= Greater than or equal  
== Equal  
!= Not equal

```
[ murray.saul ] awk '$5 < 5000 {print $1,$2,$4,$5}' cars.txt
plym fury 73 2500
chevy nova 60 3000
chevy nova 50 3500
fiat 600 115 450
toyota tercel 180 750
chevy impala 85 1550
```

Using the awk command to display results based on **comparison operators**.

10. Issue the following linux pipeline command to display display the **car make**, **model**, **quantity** and **price** of all vehicles whose **prices are less than \$5,000**:

```
awk '$5 < 5000 {print $1,$2,$4,$5}' cars.txt | tee awk-4.txt
```

What do you notice?

11. Issue the following linux pipeline command to display display **price**, **quantity**, **model** and **car make** of vehicles whose **prices are less than \$5,000**:

```
awk '$5 < 5000 {print $5,$4,$2,$1}' cars.txt | tee awk-5.txt
```

12. Issue the following linux pipeline command to display the **car make**, **year** and **quantity** of cars that **begin** with the **letter 'f'**:

```
awk '$1 ~ /^f/ {print $1,$2,$4}' cars.txt | tee awk-6.txt
```

Combined pattern searches can be made by using **compound operator** symbols:

&& (and)  
|| (or)

```
[ murray.saul ] awk '$1 ~ /ford/ && $5 > 10000 {print $0}' cars.txt
ford ltd 83 15 10500
ford thundbd 84 10 17000
```

Using the awk command to display combined search results based on **compound operators**.

13. Issue the following linux pipeline command to list all **fords** whose **price is greater than \$10,000**:

```
awk '$1 ~ /ford/ && $5 > 10000 {print $0}' cars.txt | tee awk-7.txt
```

14. Issue the following linux command:

```
cp -ulil01/tutorialfiles/cars2.txt ~/awk
```

15. Issue the **cat** command to quickly view the contents of the **cars2.txt** file.

16. Issue the following linux pipeline command to display the **year** and **quantity** of cars that **begin** with the **letter 'f'** for the **cars2.txt** database:

```
awk '$1 ~ /^f/ {print $2,$4}' cars2.txt | tee awk-8.txt
```

What did you notice?

The problem is that the **cars2.txt** database separates each field by a semi-colon (;) instead of **TAB**. Therefore, it does not recognize the second and fourth fields.

You need to issue awk with the -F option to indicate that this file's fields are separated (delimited) by a semi-colon.

17. Issue the following linux pipeline command to display the **year** and **quantity** of cars that **begin** with the **letter 'f'** for the **cars2.txt** database:

```
awk -F";" ' $1 ~ /^f/ {print $2,$4}' cars2.txt | tee awk-9.txt
```

What did you notice this time?

18. Issue the following to run a checking script:

```
~uli101/week10-check-2
```

If you encounter errors, make corrections and **re-run** the checking script until you receive a congratulations message, then you can proceed.

# LINUX PRACTICE QUESTIONS

The purpose of this section is to obtain **extra practice** to help with **quizzes**, your **midterm**, and your **final exam**.

Here is a link to the MS Word Document of ALL of the questions displayed below but with extra room to answer on the document to simulate a quiz:

[https://wiki.cdote.senecacollege.ca/uli101/files/uli101\\_week11\\_practice.docx](https://wiki.cdote.senecacollege.ca/uli101/files/uli101_week11_practice.docx)

Your instructor may take-up these questions during class. It is up to the student to attend classes in order to obtain the answers to the following questions. Your instructor will NOT provide these answers in any other form (eg. e-mail, etc).

## Review Questions:

### Part A: Display Results from Using the sed Utility

Note the contents from the following tab-delimited file called **~murray.saul/uli101/stuff.txt**: (this file pathname exists for checking your work)

```
Line one.
This is the second line.
This is the third.
This is line four.
Five.
Line six follows
Followed by 7
Now line 8
and line nine
Finally, line 10
```

Write the results of each of the following Linux commands for the above-mentioned file:



1. **sed -n '3,6 p' ~murray.saul/uli101/stuff.txt**
2. **sed '4 q' ~murray.saul/uli101/stuff.txt**
3. **sed '/the/ d' ~murray.saul/uli101/stuff.txt**
4. **sed 's/line/NUMBER/g' ~murray.saul/uli101/stuff.txt**

## Part B: Writing Linux Commands Using the sed Utility

Write a single Linux command to perform the specified tasks for each of the following questions.

1. Write a Linux sed command to display only lines 5 to 9 for the file: **~murray.saul/uli101/stuff.txt**
2. Write a Linux sed command to display only lines the begin the pattern “and” for the file: **~murray.saul/uli101/stuff.txt**
3. Write a Linux sed command to display only lines that end with a digit for the file: **~murray.saul/uli101/stuff.txt**
4. Write a Linux sed command to save lines that match the pattern “line” (upper or lowercase) for the file: **~murray.saul/uli101/stuff.txt** and save results (overwriting previous contents) to: **~/results.txt**

## Part C: Writing Linux Commands Using the awk Utility

Note the contents from the following tab-delimited file called **~murray.saul/uli101/stuff.txt**: (this file pathname exists for checking your work)

```
Line one.
This is the second line.
This is the third.
This is line four.
Five.
Line six follows
Followed by 7
Now line 8
and line nine
Finally, line 10
```

Write the results of each of the following Linux commands for the above-mentioned file:

1. **awk 'NR == 3 {print}' ~murray.saul/uli101/stuff.txt**
2. **awk 'NR >= 2 && NR <= 5 {print}' ~murray.saul/uli101/stuff.txt**
3. **awk '\$1 ~ /This/ {print \$2}' ~murray.saul/uli101/stuff.txt**
4. **awk '\$1 ~ /This/ {print \$3,\$2}' ~murray.saul/uli101/stuff.txt**

## Part D: Writing Linux Commands Using the awk Utility

Write a single Linux command to perform the specified tasks for each of the following questions.

1. Write a Linux awk command to display all records for the file: `~/cars` whose fifth field is greater than 10000.
2. Write a Linux awk command to display the first and fourth fields for the file: `~/cars` whose fifth field begins with a number.
3. Write a Linux awk command to display the second and third fields for the file: `~/cars` for records that match the pattern “chevy”.
4. Write a Linux awk command to display the first and second fields for all the records contained in the file: `~/cars`

---

Author: Murray Saul

License: LGPL version 3 Link: <https://www.gnu.org/licenses/lgpl.html>

---

Retrieved from "[https://wiki.cdott.senecacollege.ca/w/index.php?title=Tutorial10:\\_Sed\\_%26\\_Awk\\_Uilities&oldid=160665](https://wiki.cdott.senecacollege.ca/w/index.php?title=Tutorial10:_Sed_%26_Awk_Uilities&oldid=160665)"

---

- This page was last edited on 4 September 2023, at 19:36.