Common Text Processing Commands in Linux Part II and Redirection Operators

This document provides the definition, usage/formula, and examples for `awk`, `sed`, `less`, and the redirection operators (`>`, `>>`, and `|`).

Commands

1. `awk`

Definition: `awk` is a versatile text-processing command used to search for patterns, extract fields, and perform operations on text files.

Usage/Formula:

```
awk 'PATTERN {ACTION}' FILE
```

Examples:

- Print the second column, a string, and the second to last field of a CSV file excluding the first 2 lines:

```
awk -F',' 'NR > 2 {print $2,"is equal to",$(NF-1)}' file.csv
```

- Print the second and last field in uppercase with a different field separator and line numbers:

```
awk -F: '{OFS=" = "}{print NR,toupper($2,$NF)}' /etc/passwd
```

- Print only a range of lines (e.g.after using grep -n on a man to find a flag):

```
man awk | awk 'NR>=870 && NR<900'
```

- Print the first field in lowercase, a header and a footer:

```
awk 'BEGIN {print "Processing File..."} {print tolower($1)} END {print
"Done!"}' /etc/passwd
```

- Print a table containing the first field and last field with a header(printf is print formatting)

```

```

```
awk -F';' 'BEGIN {printf "%s\t%s\n","Brand","Rating"}{print
$1,"\t",$NF}' ~/Documents/Csv/cereal.csv
```

- Print lines containing the word "error":

```
awk '/error/' file.txt
```

- Calculate the sum of values in the second column:

```
awk '{sum += $2} END {print sum}' file.txt
```

2. `sed`

Definition: `sed` (stream editor) is used to perform text transformations, such as find-and-replace operations, on files or input streams.

Usage/Formula:

```
sed [OPTION] 'COMMAND' FILE
```

Examples:

- Replace the first occurrence in a line, for every line:

```
sed 's/home/house/' file_report.txt
```

- Replace every occurrence in a line, for every line:

```
sed 's/home/house/g' file_report.txt
```

- Replace the third occurrence in the 9th line:

```
sed '9s/home/house/3' file_report.txt
```

- Replace every occurrence starting from the second occurrence, for the lines ranging from 3 to 9:

```
sed '3,9s/home/house/2g' file_report.txt
```

- Delete lines containing a pattern:

```
sed '/abc123/d' file.txt
```

- Delete the fifth line in a file:

```
sed '5d' file.txt
```

- Delete the last line in a file:

```
sed '$d' file.txt
```

- Delete the lines ranging from 5 to the last line in a file:

```
sed '5,$d' file.txt
```

- Insert one blank line after each line:

```
sed G file.txt
```

- Insert two blank lines:

```
sed 'G;G' file.txt
```

- Insert a line after every line containing "header":

```
sed '/header/a\This is a new line.' file.txt
```

- Insert a line above every line matching "header":

```
sed '/header/{x;p;x;}' file.txt
```

- Insert 5 spaces at the beginning of every line:

```
sed 's/^/     /' file.txt
```

NOTE:

- G is for adding blanklines: 6G specifies what line we want to ADD a blankline after (e.g. 6th line)
- g is for replacing with blanklines: 3g specifies which line we want to REPLACE(e.g.3rd line)

3. `less`

Definition: `less` is a pager command used to view large files one screen at a time, allowing navigation forward and backward.

Usage/Formula:

```
less [OPTIONS] FILE
```

Examples:

- View a file with `less`:

  ```
  less file.txt
  ```

- Search for a specific word while viewing: Press `/` followed by the search term (e.g., `/error`) and hit Enter.
- Quit: Press `q`.
- Go to the end: Press `G`.
- Go to the beginning: Press `g`.

Redirection Operators

1. Output Redirection (`>`)

Definition: The `>` operator redirects the output of a command to a file, overwriting the file if it exists.

Usage/Formula:

```
COMMAND > FILE
```

Examples:

- Redirect `ls` output to a file:

  ```
  ls > file_list.txt
  ```

- Overwrite a file with new content:

```
echo "New content" > file.txt
```

- Save the error to a file and the success to another:

```
ls -lA Downloads/Pictures > success.txt 2> error.txt
```

- Save the error and the success to the same file:

```
ls -lA Downloads/Pictures &> alloutput.txt
```

- Do not display errors:

```
ls -lA Downloads/ 2> /dev/null
```

2. Append Redirection (>>)

Definition: The >> operator appends the output of a command to a file without overwriting its content.

Usage/Formula:

```
COMMAND >> FILE
```

Examples:

- Append the date to a log file:

```
date >> log.txt
```

- Append the output of `ls` to an existing file:

```
ls >> file_list.txt
```

3. Pipe (|)

Definition: The | operator passes the output of one command as input to another.

Usage/Formula:

```
COMMAND1 | COMMAND2
```

Examples:

- Pass the output of `ls` to `grep` to filter results:

```
ls | grep "txt"
```

- Count the number of `.txt` files in a directory:

```
ls | grep "txt" | wc -l
```

- Search for "error" in a file and view the result with `less`:

```
grep "error" file.txt | less
```

- Display only the second line in a file:

```
head -2 file.lst | tail -1
```

- **IMPORTANT** Display only the ip addresses from the output of the ip command

```
ip addr | grep -Eo '[[:digit:]]{1,3}\.[[:digit:]]{1,3}\.[[:digit:]]
{1,3}\.[[:digit:]]{1,3}'
```