UNIX / Linux: awk command

Learning Objectives: Upon successful completion of this assignment the student will be able to:

- Explain and implement awk's unusual syntax
- Split a line into fields and format the output

Named after its authors, Aho, Weinberger, and Kernighan, the **awk** command is a powerful method for processing or analyzing text files, in particular data files that are organized by lines (rows) and columns. This command combines features of several filters. It can identify and manipulate individual fields in a line. It is one of the few UNIX filters that can perform computations. Further, awk also accepts extended regular expressions (EREs) for pattern matching, has C-type programming constructs, and has several built-in variables and functions.

Simple *awk* commands can be run from the command line. More complex tasks should be written as awk programs (so-called awk scripts) to a file.

The basic format of an awk command looks like this:

```
awk 'pattern {action}' input-file > output-file
```

This means: take each line of the input file; if the line contains the pattern apply the action to the line and write the resulting line to the output-file. If the pattern is omitted, the action is applied to all line.

Locate the /tmp folder on the server. Copy the file musicartists to your homework directory and rename it table1.txt Next, follow the exercises below. Save this file as .pdf, upload and submit the URL.

For example:

```
awk '{ print $5 }' table1.txt > output1.txt
```

This statement takes the element of the 5th column of each line and writes it as a line in the output file "output.txt". The variable '$4' refers to the second column. Similarly you can access the first, second, and third column, with $1, $2, $3, etc. By default columns are assumed to be separated by spaces or tabs (so called white space). So, if the input file "table1.txt" contains these lines:

```
1, Justin Timberlake, Title 545, Price $7.30
2, Taylor Swift, Title 723, Price $7.90
3, Mick Jagger, Title 610, Price $7.90
4, Lady Gaga, Title 118, Price $7.30
5, Johnny Cash, Title 482, Price $6.50
```

```
6, Elvis Presley, Title 335, Price $7.30
7, John Lennon, Title 271, Price $7.90
8, Michael Jackson, Title 373, Price $5.50
```

Then the command would write the following lines to the output file "output1.txt":

```
545,
723,
610,
118,
482,
335,
271,
373,
```

If the column separator is something other than spaces or tabs, such as a comma, you can specify that in the awk statement as follows:

```
awk -F, '{ print $3 }' table1.txt > output1.txt
```

This will select the element from column 3 of each line if the columns are considered to be separated by a comma. Therefore the output, in this case, would be:

```
 Title 545
 Title 723
 Title 610
 Title 118
 Title 482
 Title 335
 Title 271
 Title 373
```

The list of statements inside the curly brackets ('{','}') is called a block. If you put a conditional expression in front of a block, the statement inside the block will be executed only if the condition is true.

```
awk '$7=="\$7.30" { print $3 }' table1.txt
```

In this case the condition is $7=="\$7.30", which means that the element at column 7 is equal to $7.30. The backslash in front of the dollar sign is used to prevent the system from interpreting $7 as a variable and instead take the dollar sign literally.

So this awk statement prints out the element at the 3rd column of each line that has a "$7.30" at column 7.

You can also use regular expressions as condition. For example:

```
awk '/30/ { print $3 }' table1.txt
```

The string between the two slashes ('/') is the regular expression. In this case it is just the string "30". This means, if a line contains the string "30", the system prints out the element at the 3rd column of that line. The output in the above example would be:

```
Timberlake,
Gaga,
Presley,
```

If the table elements are numbers awk can run calculations on them as in this example:

```
awk '{ print ($2 * $3) + $7 }'
```

Besides the variables that access elements of the current row ($1, $2, etc.) there is the variable $0 which refers to the complete row (line), and the variable NF which holds to the number of fields.

You can also define new variables as in this example:

```
awk '{ sum=0; for (col=1; col<=NF; col++) sum += $col; print sum; }'
```

This computes and prints the sum of all the elements of each row.

Awk statements are frequently combined with sed commands.

Research and find 2 awk commands used for ethical hacking and list them here.

The following commands would give you a list of all users whose home directories could be modified by anyone.

```
ls -l /home > usr_list
awk '{ print $3, $1 }' usr_list > usr_ls_permissions
awk '/w.$/ { print $1 }' usr_ls_permissions > writable_usr_ls
cat writable_usr_ls
```

These commands would return a list of hidden executable files in a given directory.

```
ls -al > file_list
awk ‘/^-..x.*\ \./ { print $3, $9 }' file_list > hidden_exe_list
cat hidden_exe_list
```