**Awk Command**

**What is Awk?**

* Awk is both a command and programming language on Linux/Unix based systems. Awk can be used to search and manipulate text files. Like grep and sed, awk can search for a specific pattern defined by an expression. Fun fact – the ‘awk’ name comes from the last names of the developers Alfred Aho, Peter Weinberger, and Brian Kernighan.

To start editing the ‘AwkLab.data’ file for this Lab, power on the Ubuntu Server in VirtualBox, and transfer this file to the home directory of the server. If needed, refer to the previous File Share Documentation for assistance.

1. **Command – “awk ‘{print $1}’ AwkLab.data”**

A screenshot of a computer

Description automatically generated

* Awk has its own programming language available for use when the command is stated. Awk along with the action ‘**{print $1}**’ tells the system to print the first field, or column, in the specified file AwkLab.data.

1. **Command – “awk -F’:’ ‘/Tom|Frodo/ {print $2}’ AwkLab.data”**

A screenshot of a computer

Description automatically generated

* Using the **“-F”** flag with the awk command, is a way of changing the field separator, or delimiter. Awk on its own will use one space on a line as the default field separator. Depending on how a file is formatted, this might need to be changed to get the results we want. For this instance, we want to print the phone numbers for both Frodo and Tom after their names. To do this, I will need to change the default delimiter to a colon **(:)** because of the format. EX: Frodo Baggins:(206) 548-1278:250:80:75. Using awk **-F’:’** tells awk to now separate the fields by colons rather than spaces. Start the pattern with **‘Tom’ *and ( | )* Frodo** to search for both names, **{print $1 $2}** then specifies to print text between the first and second colons that also match the names given.

1. **Command – “awk ‘/Peregrin/ {print $1 $2}’ AwkLab.data”**

A screen shot of a computer

Description automatically generated

* This time pattern defined will tell awk to search for the name “**Peregrin**”, and then print the first two fields that are separated by a space on the line that matches the name.

1. **Command – “awk -F’:’ ‘/123/ {print $1 $2}’ AwkLab.data”**

A screenshot of a computer

Description automatically generated

* To find the names on this list who have an area code of “123”, use a semi colon **(-F’:’**) as the delimiter to separate the columns. The pattern then states to match the number “123”, which is the area code we’re searching for, and then print columns 1 and 2 for all the area codes that match the search.

1. **Command – “awk -F’[ :]’ ‘{ if ($2 ~ /^[DT]/) {print $2} if ($3 ~ /^[DT]/) {print $3}}’ AwkLab.data”**

A screenshot of a computer

Description automatically generated

* This took a long time to figure out. Eventually I found a way to write an “If then print” statement using awk. In short, the statement tells the system to print the string only if the condition specified is met. Using the option **-F’:’** again to use colons to separate the fields, I can then begin the if statement to print all Last Names that start with a D or T. “if ($2 ~ /^[DT] calls out to look in the second field. ($2), for strings that are equal to, **(~),** the letters D or T [DT] at the beginning of the field using **(^).** To account for those on the list that have middle names included, use another if statement for column 3 by inserting a $3 where the $2 was in the previous one. This way awk can print last names, that start with a D or T from fields 2 and 3.

1. **Command – “awk ‘4>=length($1) {print$1}’ AwkLab.data”**

A screen shot of a computer

Description automatically generated

* Another example of how awk can be used to print certain patterns, the goal of this command is to print all strings in the first field that are less than 4 characters in length. To search for a pattern or condition like this, use the **‘4>=length’** to tell the system to find any string less than 4 characters. Then we specify the first field **($1)** because I want to look for first names in this file. Finish the command with **{print $1}** to print the specified pattern.

1. **Command – “awk ‘/(916)/ {print $1}’ AwkLab.data”**

A black screen with white text

Description automatically generated

* Awk will use this pattern to match the desired area code (916) and then print the first field / first names ($1). This command only prints the first names that match the string, in this case the area code (916), specified in the pattern.

1. **Command – “awk -F: ‘/Sacharissa/ {print $1 “\t” “$”$3, “$”$4, “$”$5}’ AwkLab.data”**

A screenshot of a computer

Description automatically generated

* Using the -F: option, this separates the fields again by a colon. It is a good habit to run “cat *filename*” to view the file contents first, and then determine which delimiter to use for each scenario. In this case I want to add the dollar sign to each campaign donation from Sacharissa. By default, the donation amounts are separated by semi colons per each donation, this is why I chose semi colons as the delimiter for this pattern. Then to add the $ symbol prior to each donation amount, use **{print “$”$3, “$”$4. “$”$5 }.** “$” is what is used to insert the symbol before the specified field. **$1** prints the name field and I added the tab spacer **“\t”** which adds a bigger space between the name and contribution fields.

1. **Command – “awk -F: ‘match($0,/\w+:/{print substr($0,RSTART,RLENGTH-1)”,” $2}’ AwkLab.data”**

A screenshot of a computer

Description automatically generated

* Use the **-F:** option to separate the fields by a colon *OR* space. The match function in this pattern, Match($0 states to search the entire line, not just one field. Then adding **/\w:+/** is what tells awk to search for any string of alphabetical characters. “Print substr” is used to select the last name (substring) of the first field, which is all we want in this case. ‘RSTART’ is what identifies where to start the substring, and **‘RLENGTH-1’** identifies the length of the substring, and subtracts 1 from it to eliminate the colon at the end of each last name. Add in the syntax to print a comma **(“,”)** before printing the second field **($2)** that is the phone number tied to the person’s name.

1. **Command – “awk -F’:’ ‘$4>110 {print $1 “\t” “$”$5}’ AwkLab.data”**

A screenshot of a computer

Description automatically generated

* Because the campaign donation amounts are separated by colons at the end of the line, I am again using the **-F:** option to separate the fields by a colon. The pattern then states that IF the fifth field **($5)** is greater than 110, THEN print their name **[Field 1, $1]** and the amount of the last donation **[Field 5, $5].** I used the tab option **(“\t) to** separate the data for formatting purposes.

1. **Command – “awk -F’:’ ‘$3<150 {split($1, field1, “ “); print field1 [2] $2 “$”$3}’ AwkLab.data”**

A black screen with white text

Description automatically generated

* Introducing the **“Split”** function of awk for this solution. Split is another way to separate specific pieces of fields that we want to pull. In this case with -F’:’ separating all fields by a colon, the first and last name are considered one field. To pull just the last name, we can tell the system to split the first field ‘{split($1,’ then give it a friendly name field1, and tell awk what to use as the delimiter, in this case we are using a space “ “. Then use print ‘field1 [2]’ to identify that we want print only the second part of it. The rest of the pattern defines how we print the phone number and donation amount *IF* it is less than 150.

1. **Command – “awk** -F**: ‘$3>10 && $3<200 {split($1, field1, “ “); print field1 [1] , “$”$3}’ AwkLab.data”**

A screenshot of a computer

Description automatically generated

* Again, using our new friend the “Split” function, I can carve up the first field containing the names into two pieces. To pull just the first name this time, print ‘field1 [1]’ to identify that we want print only the first piece of this field. $3>10 && $3<200 says if Field 3 is greater than 10, but less than 200, then print ‘Print field1 [1] , “$”$3}’ which means to print the first name (*first part of first field)* along with their respective donation amount (*third field*) for the first month.

1. **Command – “awk -F: ‘{x=$3+$4+$5} x<700 {print $1, “$”x}’ AwkLab.data”**

A screenshot of a computer

Description automatically generated

* We can set variables using awk, such as x in this equation. The goal is to determine which donation amounts over the three-month period are less than 700. First, I set X to equal the three-month donation amount total for each individual person. The rest of the statement says IF X is less than 700, then print the first field (persons names) and print their total donation amount (X). I inserted a dollar sign again using quotes “$” just for formatting purposes, I think it makes the results easier to read.

1. **Command – “awk -F: ‘match($0,/\w+:/) avg{$3+$4+$5)/3; if(avg>100) {print substr($0,0,RSTART), “$”avg}}’ AwkLab.data”**

A screenshot of a computer

Description automatically generated

* Use the **-F:** option to separate the fields by a colon *OR* space. Similar to the match function in Example 9**, Match($0** states to search the entire line, not just one field. Then using **/\w:+/** is what tells awk to search for any string of alphabetical characters. To find the average donation amount, we need to create another variable that stores the average value. I’m using **‘avg’** as the name for this variable. Add the three donation fields together and divide by 3 to get the average - avg**{$3+$4+$5)/3.** The second half of the pattern tells awk to print the first name and last initial **{print substr($0,0,RSTART),** if their donation average is greater than 100. **‘RSTART’** helps awk to find the last initial of the person. Last, print the average amount of their donations “**$”avg.**

1. **Command – “awk -F: ‘!/(916)/{match($0,/\w+:/); print substr($1, RSTART) , substr($2,1,5)}’ AwkLab.data”**

A screenshot of a computer

Description automatically generated

* Awk can be told to avoid matching specified patterns as seen in this example, **(!/(916/** will tell awk to avoid the number / area code (916). Then using our reliable match function that matches any string consisting of alphabetical characters, we can tell awk to match print the last names of those who do not have a (916) area code. In the first sub string, I used ‘RSTART’ again to identify the beginning of the last match with this sub string. To single out the area code, I needed to print only the first 5 characters of the second substring, note I did 5 instead of 3 to include the parentheses. So **substr($2,1,5)** prints from the second field **($2)**, starting at the first character of the string (1), and the first five characters (5).

1. **Command – “awk ‘{print NR , $0}’ AwkLab.data”**

A screenshot of a computer

Description automatically generated

* Think of the ‘NR’ option in awk as a translation for “Record Number”. This useful tool can be used to identify how many total records, or lines, are in each file. **Print NR** simply tells awk to print the Record Number and the entire record or line **($0)**.

1. **Command – “awk -F: ‘{total=$3+$4+$5}; {print $1 , “$”total}’ AwkLab.data”**

A screenshot of a computer

Description automatically generated

* I started by creating a variable **“total”** that awk can use to hold the total donation value for everyone. We know that using a colon as the delimiter, means fields 3, 4, and 5 are the donation values we want to add together – **{total=$3+$4+$5}**. Then use field one ($1) to print the name of the person which is followed by their total donation amount **{print $1 , “$”total}**

1. **Command – “awk -F: ‘/Tiffany Aching/ {print $1 , “$”$3+10}’ AwkLab.data”**

A screen shot of a computer

Description automatically generated

* Isolate the fields again using a colon, (**awk -F:)** then match Tiffany’s full name **(/Tiffany Aching/).** This command will now only match lines with her name. Now because we want to add 10 to her first donation amount, I need to add 10 to the third field **($3+10).** The updated amount along with her name are then printed.

1. **Command – “awk ‘{gsub(/Samwise Gamgee/,”Sean Astin”); print $0}’ AwkLab.data”**

A screenshot of a computer

Description automatically generated

* **Gsub** is another useful command, short for Global Substitution. It is very similar to how we did substitutions with the sed command. This command tells awk to find the name Samwise Gamgee and change it to Sean Astin. Because this is global substitution, any instances of this name in this file will be changed to Sean Astin. Sub is another command for substitutions but will only change the first occurrence of the pattern specified.

1. **Command – “awk -f AwkScript.awk AwkLab.data > AwkLab”**

A screenshot of a computer

Description automatically generated

* Use a text editor (I prefer Notepad++ which can be installed [here](https://notepad-plus-plus.org/)) use a new text file for writing the awk script. When finished, save the text file with the extension **“.awk”.** Then upload it to the Ubuntu server, refer to the previous file share document if needed. When running the script, be sure to write the results to a file using ‘> *filename’*. If you want to append to an existing file, use ‘>> *filename’*. I have commented additional notes into the script (see below).

**Script**

#!/usr/bin/awk -f

#The top line is what states to execute the script using awk.

BEGIN { FS=":"}

#Tells the script where to begin, and sets the delimiter to a colon (:)

/Took/ {split($1, field1," "); print field1 [1] , "$" $3+$4+$5};

#Searches for the name Took, and splits the first field by first name / last name so we can print ONLY the first name of those who have last name 'Took'

$5>10 && $5<200 {print $1 , "$"$5};

#If field 5 is less than 10, AND if field 5 is less than 200, print the full first field (First Name Last Name) and their donation amount (Field 5).

{avg=($3+$4+$5)/3; if(avg<300) print $1 , "$"avg};

#Store the average donation amount per person into a variable called 'avg'. Then if the average (avg) is less than 300, print their full name and average donation amount.

**Sources**

<https://www.aholdengouveia.name/LinuxAdmin/Awk.html>

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

<https://flylib.com/books/en/4.356.1.52/1/>

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

<https://stackoverflow.com/questions/32481877/what-are-nr-and-fnr-and-what-does-nr-fnr-imply>

<https://stackoverflow.com/questions/26113399/awk-sub-function-syntax>