**Documentation for AWK in Ubuntu**

**Brayan Rosario**

**CIS-245**

**LINUX ADMINISTRATION**

**AWK: is a scripting language used for manipulating data and generating reports. Allow users to use variables, numeric, functions, string functions, and logical operations. You can do some other things with AWK such as: Scan a file line by line, splits each input line into fields, compares input line/fields to pattern and Performs action.**

**In this course we are asked to resolve an amount of 20 questions using the help of** [**https://flylib.com/books/en/4.356.1.63/1/**](https://flylib.com/books/en/4.356.1.63/1/) **and some youtube videos and some other resources.**

**1 Print all the phone numbers.**

**For this question This is how I solved it.**

**Awk: Programming language to manipulating data.**

**-F: is a way to changes the delimiter**

**‘{print $2}’: inside the quotes you will open the brackets and inside of the brackets you will use (Print) to print whatever you are required to print in this case the phone numbers, as the phone numbers are in the second spot in our file you will use $ dollar sing to matches the end of the string and then the number 2 followed of file name.**

**Awk -F: ‘{print $2}’ lab3october.txt**

**Text

Description automatically generated**

**2 Print Dan’s phone number.**

**Awk: programing language to manipulating data.**

**-F: is a way to change the delimiter.**

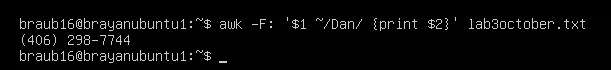
**‘’: the quotes to insert the syntax of your commands.**

**$1: to matches the end of the first line of your file.**

**~/Dan/: the tilde is use to match an expression within a record or a file, the name of the person inside of the slashes to indicate what’s the name phone number you want to show on the screen**

**{print $2}: and then I will add print $2 inside of the brackets, to print the second line of the file which is where the numbers are located because the dollar sign will match the end of the string and in this part of the code will match the second line which is where the numbers are and then the file of your preference.**

**Awk -F: ‘$1 ~/Dan/ {print $2}’ lab3october.txt**

****

**3 Print Susan’s name and phone number.**

**For this one it wants the name to append to the phone number this is what I used.**

**Awk: programming language to manipulating data**

**-F: is a way to change the delimiter.**

**‘’: the quotes to insert the syntax of your commands.**

**/ /: the slashes to insert the string name that we want to print for our question in this case is Susan.**

**{print $1,$2}: print inside of those brackets followed of $1 and $2 which indicate the last thing of the spot 1 and the spot 2 of you file and will be printed. In this case Susan is the name we want to print and for that we apply the $1 for the last thing of the file in the number 1 spot and $2 which print the second thing in the file that respond to phone number.**

**A picture containing graphical user interface

Description automatically generated**

**4 Print all last names beginning with S.**

**For this question we are looking for just for the last names starting with S.**

**Awk: for the programming language to manipulating data.**

**-F”[ :]”: this option is followed by a regular expression enclosed in brackets. If a space, colon, or tab is encountered and AWK will use that character as a field separator. The expression is using double quotes so that the shell will not pounce on the metacharacters for its own.**

**‘’: singles quote to insert the rest of your code and complete your syntax.**

**$2 ~/^S/: Dollar sing with the number 2 to indicate the second spot of the file, followed the tilde (~) to match an expression within a record or a file then, I used the (/^S/) to display all lines in the file that start with the regular expression S in this case the last name starting with the letters S.**

**{print $2}: inside of those brackets I selected print followed of the dollar sign and the number 2 to just print the second spot of the file after the names to just print the last names starting with s.**

**Awk -F”[ :]” ‘$2 ~/^S/ {print $2}’ lab3ocotober.txt**

**A picture containing text

Description automatically generated**

**5 Print all first names beginning with either a C or J.**

**Awk: for the programming language to manipulating data.**

**$1: to indicate the end of the file and matches at end of the string, using the number 1 in front of the dollar sign will display just the first line or spot of the file.**

**~/^(C|J)/ : the tilde to match an expression within a record or a file. The /^ to display all the lines in the file that start with a regular expression (C|J)/ inside the brackets I added the E and the J followed with the (|) which means a pipe that can combine both letters and will display the letters you put on and then I added the {print $1} this will print the first line of the file because the $1 will matches the end of the string, followed of the file name of your preference.**

**Awk ‘$1 ~/^(C|J)/ {print $1}’ lab3ocotober.txt**

**Text

Description automatically generated**

**6 Print all the lines containing only four characters.**

**Awk: for the programming language to manipulating data.**

**Length($1) ==4: it will return the number of characters of the given string and the bracket with the dollar sing and the number 1 will just print the end of the first line of the actual file you are working on the (==4) will just display the four characters of the string you want to be shown on the screen.**

**{print $1}: and print followed with the dollar sing with the number 1 in the front will just print the first line of the file, in this case just will print the lines containing only four characters.**

**Awk ‘length($1) ==4 {print $1}’ lab3october3.txt**

**Text

Description automatically generated**

**7 Print the first names of all those in the 916 are code .**

**Awk: for the programming language to manipulating data.**

**-F”[ :]”: this option is followed by a regular expression enclosed in brackets. If a space, colon, or tab is encountered and AWK will use that character as a field separator. The expression is using double quotes so that the shell will not pounce on the metacharacters for its own.**

**‘’: singles quote to insert the rest of your code and complete your syntax.**

**$3 ~/(916)/: the dollar sign with the number 3 in front will match the end of the line 3 in the file where the 916 is located. The (~) the tilde to match an expression within a record or a file. The /(916)/ the slashes and the ( ) inside with the 916 to indicate the exact number we are looking for.**

**{print $1}: this will take the 916 and will print or display it in the first spot of the file.**

**Awk: -F”[ :]” ‘$3~/(916)/ {print $1}’ lab3ocotber.txt**

**A picture containing meter, clock

Description automatically generated**

**8 Print mike’s campaign contribution. Each value should be printed with a leading dollar sign $250 $100 $175**

**Awk: for the programming language to manipulating data.**

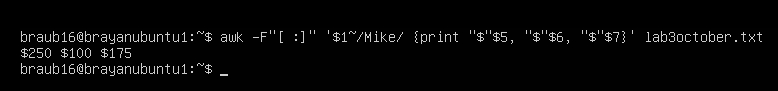
**-F”[ :]” this option is followed by a regular expression enclosed in brackets. If a space, colon, or tab is encountered and AWK will use that character as a field separator. The expression is using double quotes so that the shell will not pounce on the metacharacters for its own.**

**‘’: the quotes to insert the syntax of your commands.**

**$1~/Mike/: the dollar sign with the number one in the front will tell the command to match the end of the first line and display this on the screen. The tilde to match an expression within a record or a file. And the mike pattern inside of the slashes will hold the name we are looking for our syntax.**

**{print “$”$3, “$”$4, “$”$5}: I select the print to display all that we want for our syntax, as the question what us to print the values with the dollar sign in the front we are going to enclose the dollar sign with double quotes as soon as the contribution are in three different number I specify a sequence followed from where they are in the file and separated with a comma between the numbers.**

**Awk -F”[ :]” ‘$1~/Mike/ {print “$”$5, “$”$6, “$”$7}’ lab3ocotober.txt**

****

**9 Print last names followed by a comma and the first name.**

**Awk: for the programming language to manipulating data.**

**-F”[ :]” this option is followed by a regular expression enclosed in brackets. If a space, colon, or tab is encountered and AWK will use that character as a field separator. The expression is using double quotes so that the shell will not pounce on the metacharacters for its own.**

**‘’: the quotes to insert the syntax of your commands.**

**{print $2 “,” $1}: the dollars sign with the number 2 in the front will match the end of the second spot where the last names are located, the comma with the double quotes on will hold the comma and will be printed in the middle of between the last name and the name, the dollar sign with the number 1 in the front will end the first spot where the names are located and with the print inside with those brackets will be printing the last name and the actual name separate by a comma.**

**Awk -F”[ :]” ‘{print $2 “,” $1}’ lab3ocotober.txt**

**Text

Description automatically generated**

**10 write an awk script called facts that**

**A: prints full names and phone numbers for the savages.**

**B: print chet contributions.**

**C: prints all those who contributed $250 the first month.**

**This question has 3 question inside.**

**>>> THE FIRST SCRIPT CODE, PRINT FULL NAMES AND PHONE NUMBERS FOR THE SAVAGE<<<**

**Inside of my VI script, using the #!/bin/bash I create this following codes.**

**Awk: : for the programming language to manipulating data.**

**-F”[ :]” this option is followed by a regular expression enclosed in brackets. If a space, colon, or tab is encountered and AWK will use that character as a field separator. The expression is using double quotes so that the shell will not pounce on the metacharacters for its own.**

**‘’: the quotes to insert the syntax of your commands.**

**$1 ~/Savage/: as we need to have the names for savages I used the dollar sign and the number 1 to match the end of the first line where the names are, the tilde (~/Savage/) to match an expression within a record or a file and the slashes enclose with the pattern name Savages to indicate what is the name we are looking for.**

**{print $1, $2, $3, $4}: as we want to print the full names of it I and the phone numbers used {print $1, $2, $3, $4} to select the first, second, third and fourth of the string in my syntax and this will print the first name and the last name which is basically the full names of Savage, and the are code with the rest of the phone numbers and it will be include showing on the screen.**

**Awk -F”[ ;]” ‘$2 ~/Savage/ {print $1, $2, $3, $4}’ lab3october.txt**

**>>> THE SECOND SCRIPT CODE, PRINT CHET CONTRIBUTION<<<**

**Awk: : for the programming language to manipulating data.**

**-F”[ :]” this option is followed by a regular expression enclosed in brackets. If a space, colon, or tab is encountered and AWK will use that character as a field separator. The expression is using double quotes so that the shell will not pounce on the metacharacters for its own.**

**‘’: the quotes to insert the syntax of your commands.**

**$1 ~/Chet/ {print $3, $4, $5}: to print the contribution I select the dollar sign with the number 1 in the front to indicate only to print the first line or the first spot of that line, the (~) to match an expression within a record or a file, with the name inside of the slashes to indicate the exact name we are looking for in our files. For the rest of the contribution and we want to print the value of it I specified the amount with a dollar sign and that will be included and will be display on the screen.**

**>>> THE THIRD SCRIPT CODE, PRINT ALL THOSE WHO CONTRIBUTED $50 THE FIRST MONTH<<<**

**Awk: : for the programming language to manipulating data.**

**-F”[ :]” this option is followed by a regular expression enclosed in brackets. If a space, colon, or tab is encountered and AWK will use that character as a field separator. The expression is using double quotes so that the shell will not pounce on the metacharacters for its own.**

**‘’: the quotes to insert the syntax of your commands.**

**$3 ~/50/: the dollar sign with the number 3 in the front will match the end of the line in the file, the (~) to match an expression within a record or a file and the 50 insert in the slash will only find those who contributed only 50.**

**{print $0}: as we want only to print those ones who contributed I select print and the dollar sign and the value of 0 to just print those who contribute with the 50 amounts.**

**Text

Description automatically generated**

**Text

Description automatically generated**