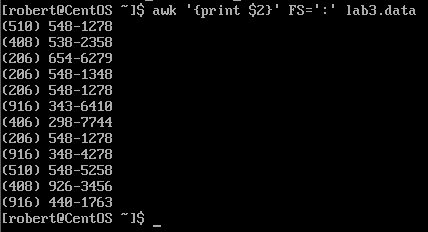
# Awk

1. Print all the phone numbers.

 **awk ‘{print $2}’ FS=’:’ filename**

**FS** : Field Separator

The Field Separator are the colons **FS= ‘:’**. **‘{print $2}** This mean to print the second field. (The phone numbers are located in the second field).

Example: Mike Harrington:(510) 548-1278:250:100:175

The Name is the first field. The Phone Number is the second field. 250 is the third field. 100 the fourth field and 175 is the fifth field.

1. Print Dan’s phone number.



**awk ‘/Dan/{print$2}’ FS=’:’ filename**

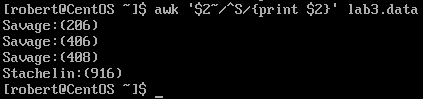
Comments: Same concept as the example above. The only difference is that we need to write the person name **‘/Dan/**’ enclosed in backslash.

1. Print Susan’s name and phone number

 **awk ‘/Susan/{print $1, $2}’ FS=’:’ filename**

Comments: Once again, the concept as the previous examples. **{Print $1, S2}** This means to print field one and two according to the field separator FS=’:’.

1. Print all last names beginning with S.

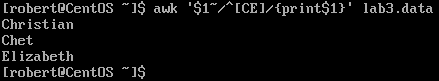
 **awk ‘$2 ~ /^S/{print2}’ filename**

Comments: “**$2”** In the field two, if the word “**match ~** “ with a “**^S**”, which means a starting S, print the field two **/{print2}**. The reason we should field two is because the last names is located in the field 2.

**^**: Indicates the beginning of the line

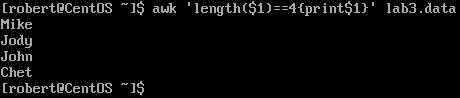
**~** : Means match

1. Print all first names beginning with either a C or E.

 **awk ‘$1 ~/^[CE]/{print$1}’ filename**

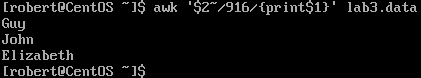
Comments: Same concept as the above example. The only difference is the **[CE]**. In this expression, awk will take the first letter inside the bracket to look for names that start with the letter C in the first field **$1**. Then it will do the same for the second letter inside the bracket [E].

1. Print all first names containing only four characters.

 **awk ‘length($1)==4{print$1}’ filename**

Comments: Awk has some similarities as the Python programing language. **Length** is self-explanatory. If we write **length($1)**, it means the length of the first field. Putting everything together **length($1)==4**. This expression means, to look for words that has a length of four characters in the first field. Remember to write an equal sign in programing language, we must use **“==”**.

1. Print the first names of all those in the 916 area code.

 **awk ‘$2 ~/916/{print$1}’ filename**

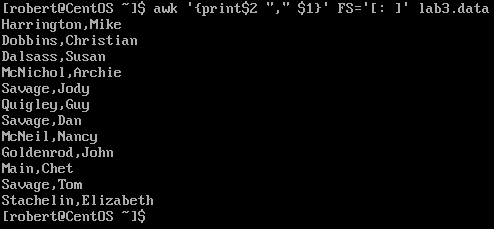
Comments: Same concept as example number 4. If field two ‘**$2**’ match ‘**~**’ with 916, print field 1 **{print$1}**. The first name is located in field number 1 and the phone number in field number 2.

1. Print Mike 's campaign contributions. Each value should be printed with a leading dollar sign; e.g., $250 $100 $175.

**awk** **‘$1 ~ /Mike****/{print “$”$3, “$”$4, “$”$5’} FS=’:’ filename**

Comments: We start by searching the name Mike in the field 1**‘$1 ~ /Mike/’** , if it matches, we want awk to print dollars sign for each field. To add write a dollar sign as a string we just need to write it inside an apostrophe **“$“**. To print each field with a dollar we also need to add a field separator **FS=’:‘**. In this example we use the colons as the field separator. If we do not add the field separator expression, it will only print a dollar sign for only one field. **/{print “$”$3, “$”$4, “$”$5’}**. This expression means, to print a string $ for the field 3, print a string $ for the field 4, print a string $ for field 5.

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

 **awk** **‘{print$2 “,” $1}’ FS=’[: ]’ lab3.data**

Comments: Same concept as the previous examples. However, this exercise took me a while to figured out. The trick was in the field separator **FS=’ ‘**. To write a “space” as a field separator, we must use **FS= ‘[: ]’**. Why do we have to use a space as a field separator? The name is the first field follow by a space then the last name. If we make the field separator as the space, the name will be the first field and last name the second. Therefore, **‘{print$2 “,” $1}’** will print the second field, a comma as a string and the first field.

1. Write an awk script called facts that
2. Prints full names and phone numbers for the Savages .
3. Prints Chet 's contributions.
4. Prints all those who contributed $50 the first month.

**vi filename.awk** to create an awk script

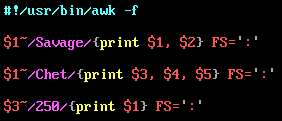
within the editor we write **#!/usr/bin/awk -f** at the top

to run the script : **awk -f filename.awk lab3.data**

Note: We do not need to use “awk” at the begging of an expression.

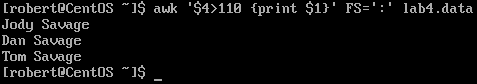
1. **$1~/Savage/{print $1, $2} FS=’:’** : If there is match in the word “Savage” in the field number 1, It will print field 1 that correspond to the first name and last name, and field 2 correspond to the phone number.
2. **$1~/Chet/{print $3, $4, $5} FS=’:’** : Same concept as the above example. Field 3,4 and 5 correspond to the contributions.
3. **$3~/250/{print $1} FS =’:’** : Field 3 correspond to the first month. If there is a match of 250 in field 3, print field 1 which correspond to the full name.

Note : In the Keyboard press **Shift + Z Z** to exit and save the file



# AWK Part II

1. Print the first and last names of those who contributed more than $110 in the second month.

 **awk ‘$4>110 {print $1}’ FS=’:’ lab4.data**

**$4** represents the field 4 or the second month. If there is a number **“>”**greater than 110 in the field 4, print field **$1** which correspond to the person full names.

“ **>** “ : Greater.

1. Print the names and phone numbers of those who contributed less than $75 in the last month.

 **awk ‘$5<85{print $1,$2}’ FS=’:’ lab4.data**

**$5** represents the last month. If there is a number less **“<”** than 75 in the field 5, print field **$1** which represents the names and field **$2** represents the last name.

“ < “ : Less than.

1. Print the names of those who contributed between $75 and $150 in the first month.

**awk ‘$3>74 && $3<151{print $1}’ FS=’:’ lab4.data**

No records were found. This one took me a bit of research to find the expression for “and” **&&**.

**&&** : And

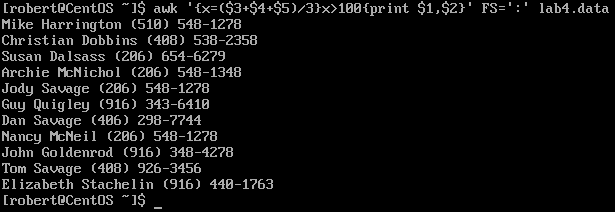
### Algebra Formulas in Awk ?

1. Print the names of those who contributed less than $700 over the three-month period.

 **awk ‘{**x**=$3+$4+$5}**x**<700{print $1}’ FS=’:’ lab4.data**

Comments: For this method we need to add all the months . But first, we must create a variable. I created variable and named “x”. Within the variable, I added field $3,$4, $5 which correspond to the three months. Then in the expression, I said if x is less than **<** 700, print field $1 which correspond to the person names.

1. Print the names and addresses of those with an average monthly contribution greater than $100

**awk ‘{x=$3+$4+$5)/3}x>100{print $1, $2}’ FS=’:’ lab4.data**

Comments: Same method as the previous example. To find the average, we must add all the months divided by how many months. If the variable is less than 100 print field $1 which represent the names and $2 the phone number.

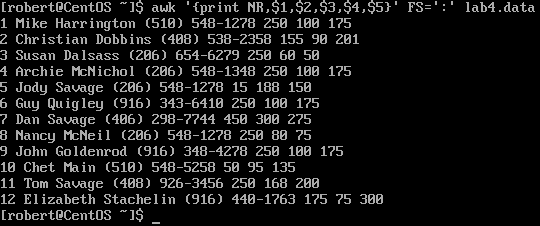
Note: Lab4 does not contain addresses.

1. Print the first name of those not in the 916 area code.

 **awk ‘$2 !~/916/{print $1}’ lab4.data**

Comments: Similar to question 7 from part 1. 916 must do not match anything in field 2. Then print field 1 avoiding 916.

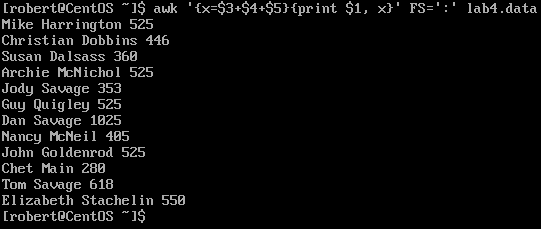
1. Print each record preceded by the number of the record.



After a bit of research, and I found that the expression for the number of records.

**NR** : The number of records.

1. Print the name and total contribution of each person.

 **awk {x=$3+$4+$5}{print $1, x}’ FS=’:’ lab4.data**

Comments: Same method as previous examples. We have to make a variable and then add all the contributions fields to find the total.

1. Add $10 to Chet 's second contribution.



Comments: **$4** represents the second field. Print the second field which is **$4** plus 10.

1. Change Nancy McNeil 's name to Doris Shutt.

 **awk ‘{gsub(/old-word/, “new-word”);}{print $1, $2}’ lab4.data**

**gsub**: Global substitution